INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Desenvolvimento Software Têxtil

VASCO JORGE FORTUNA LUSITANO
Outubro de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Desenvolvimento de Software de Gestão de Laboratório Têxtil

Vasco Jorge Fortuna Lusitano

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Paulo Oliveira

Coorientador: Nelson Freire

Porto, outubro de 2021

Dedico a todas as pessoas que contribuíram para a minha formação ao longo destes anos,

em especial para a minha família, amigos e docentes.

Vasco Lusitano

Agradecimentos

Em primeiro lugar, presto o meu agradecimento a todos os professores que nestes anos de mestrado me proporcionaram uma formação dinâmica e profissional, bem como aos meus colegas de curso com quem trabalhei e cresci durantes estes anos.

Quero destacar os meus orientadores Paulo Oliveira e Nelson Freire pelo apoio fundamental e orientação decisiva que me proporcionou ao longo de todo o processo. Aproveito também para deixar uma palavra de elogio pelo profissionalismo, total disponibilidade e paciência de ambos.

O meu agradecimento ao meu amigo Pedro Torres pela sua disponibilidade ao longo de todo este período, assim como a todos os colaboradores da Empresa Têxtil da Maganha que me acolheram e me ajudaram durante o todo o desenrolar do projeto.

Por último, mas não menos importante, uma referência para a minha família, que foi um pilar importantíssimo durante todo o meu percurso académico.

Abstract

The Fourth Industrial Revolution is a new phase of the textile industry. This uprising focuses on the connectivity and automation of manual processes, forcing textile companies to look for goods that make them go beyond the archaic level of outdated methods.

After contacting an engineer in the textile area, with several years of experience, the need for companies in the sector to use a technology capable of handling and storing the data inherent to their methodologies was evident.

Thus, the objective of idealizing and developing a solution oriented to the needs of the company in which the textile professional worked was aligned, namely Empresa Têxtil da Maganha.

This document is divided into 7 chapters, each of which has its responsibilities, in more detail, the introduction, the state of the art, the value analysis, the requirements and design analysis, the implementation, the experimentation and evaluation and the conclusion.

The initial part presents the context, approach, problems and objectives, in a perspective that invites the reader to understand the theme and seek a solution. Next, the planning and technical specification are presented, showing the state of the art decisions made. The technical description in the report highlights the requirements analysis and architectural design, followed by the details of the system implementation.

With the development of the textile software disappearing, a quality solution was achieved, following the standards and models described.

Palavras-chave: BMS, *serverless*, laboratory management, JavaScript

Resumo

A Quarta Revolução Industrial trata-se de uma nova fase da indústria têxtil. Esta insurreição dá foco à conectividade e automação dos processos manuais, obrigando as empresas têxteis a procurar por bens que as façam ultrapassar o patamar arcaico de métodos desatualizados.

Após o contacto com um engenheiro da área têxtil, com vários anos de experiência, foi evidente a necessidade de as empresas do ramo utilizarem uma tecnologia capaz de manipular e guardar os dados inerentes às suas metodologias.

Assim, alinhou-se o objetivo de idealizar e desenvolver uma solução orientada às necessidades da empresa na qual o profissional da área têxtil trabalhou, a Empresa Têxtil da Maganha.

O presente documento divide-se em 7 capítulos, cada um dos quais com as suas responsabilidades, mais detalhadamente, a introdução, o estado de arte, a análise de valor, a análise de requisitos e conceção, a implementação, a experimentação e avaliação e a conclusão.

A parte inicial apresenta o contexto, abordagem, problemas e objetivos, numa perspetiva que convida o leitor a compreender o tema e a visar uma solução. De seguida são expostos o planeamento e a especificação técnica, evidenciando-se no estado da arte as decisões tomadas. A descrição técnica no relatório destaca a análise dos requisitos e desenho arquitetural, seguidos pelo detalhamento da implementação do sistema.

Sumindo a elaboração do software têxtil, foi alcançada uma solução com qualidade, seguindo os padrões e modelos descritos.

Palavras-chave: BMS, *serverless*, gestão de laboratório, JavaScript

Índice

1 Introdução 1

1.1 Contexto 1

1.2 Problema 2

1.3 Objetivos 2

1.4 Abordagem 3

1.5 Estrutura do documento 4

2 Estado da Arte 5

2.1 Domínio do problema 6
2.1.1 Conceitos gerais da indústria têxtil vestuário 6
2.1.2 Indústria tinturaria e acabamentos 7

2.2 Tecnologias de Gestão de Organizações 12
2.2.1 Soluções Genéricas 13
2.2.2 Soluções de Mercado Para a Área Têxtil 15

2.3 Desenvolvimento de Solução 20
2.3.1 Arquitetura de Software 21

2.4 Tecnologias de Desenvolvimento 27
2.4.1 No-Code 27
2.4.2 Low-Code 28
2.4.3 Comparação Plataformas No-Code e Low-Code 29
2.4.4 Developer-Code 30
2.4.5 Interpretadores de Código 36
2.4.6 Base de dados 39

3 Análise de Valor 51

3.1 *Front-end* no Desenvolvimento de Novos Produtos 51

3.2 New Concept Development Model 52
3.2.1 Identificação da Oportunidade 54
3.2.2 Análise da oportunidade 54
3.2.3 Criação da ideia e enriquecimento 55
3.2.4 Escolha da ideia 55
3.2.5 Definição de conceitos 55
3.2.6 Sistema Stage-gate 56

3.3 Valor, Valor para o Cliente e Valor Percecionado 57

3.4 Proposta de Valor 60

3.5 Modelo de Negócio Canvas 61
3.5.1 Propostas de Valor 62
3.5.2 Segmentos de Clientes 62
3.5.3 Canais 62
3.5.4 Relacionamentos com Clientes 63

3.5.5 Fontes de Receita 63
3.5.6 Recursos Chave 63
3.5.7 Atividades Chave 64
3.5.8 Parceiros Chave 64
3.5.9 Estrutura de Custos 64
3.6 QFD 65
4 Análise de Requisitos e Conceção 70
4.1 Engenharia de Requisitos 70
4.1.1 Modelo FURPS+ 71
4.1.2 Requisitos funcionais 71
4.1.3 Requisitos não funcionais 81
4.2 Análise e Conceção da Solução 83
4.2.1 Modelo de Domínio 83
4.2.2 Diagrama de Camadas 84
4.2.3 Diagrama de Implementação 87
4.2.4 Diagrama de Implantação 88
5 Implementação 89
5.1 *Stack* Tecnológico 89
5.1.1 Cliente 91
5.1.2 Servidor 102
5.1.3 Base de Dados 105
5.2 Testes 107
5.2.1 Testes Unitários 108
5.2.2 Testes Integração 109
5.2.3 Testes de Sistema 110
5.2.4 Testes de Aceitação 111
6 Experimentação e Avaliação 112
6.1 Hipóteses 112
6.2 Identificação dos Indicadores e Fontes de Informação 113
6.2.1 Identificação dos Indicadores 113
6.2.2 Fontes de Informação 113
6.3 Descrição da Metodologia de Avaliação 114
6.3.1 Testes de Qualidade de Software 114
6.3.2 Inquéritos de Avaliação 115
7 Conclusão 117
7.1 Objetivos Concretizados 117
7.2 Limitações Encontradas 118
7.3 Trabalho Futuro 118

Lista de Figuras

Figura 1 – Processo produtivo da transformação na indústria têxtil 6
Figura 2 – Conceitos inerentes à indústria de tinturaria e acabamentos e respetivas interações 7
Figura 3 - Fases do processo produtivo no tingimento e acabamento 8
Figura 4 - Estrutura de um cartaz de cores enviado pelo cliente 9
Figura 5 – Constituição de um *lab-dip* 10
Figura 6 – Estados de um ensaio de laboratório 10
Figura 7 – Classificação das fibras quanto à sua origem 11
Figura 8 – Fluxograma do processo de abertura de nova cor num laboratório têxtil 12
Figura 9 – Subdivisão dos módulos do *software Texplus* 16
Figura 10 – Subdivisão do *software ERP* da empresa *MacWinFonte:* (MacWin, 2021) 18
Figura 11 – Representação gráfica da lógica de Fowler na qualidade interna do *software* 22
Figura 12 – Benefícios obtidos com arquitetura *Serverless* 26
Figura 13 – Gráfico comparativo de despesas associadas à escalabilidade 26
Figura 14 – Exemplo demonstrativo de *workflow* de desenvolvimento *low-code* 28
Figura 15 – Correlação entre grau de personalização e velocidade de desenvolvimento 30
Figura 16 - Popularidade das bases de dados relacionais 43
Figura 17 – Popularidade das bases de dados não relacionais 47
Figura 18 – Sintaxe CRUD de MySQL vs MongoDB 49
Figura 19 – Fluxo de atividades do desenvolvimento de novos produtos 52
Figura 20 – *New Concept Development* (NCD) *Model* desenvolvido por Koen 53
Figura 21 – Representação do sistema *Stage-gate* de Robert Cooper 56
Figura 22 – Fórmula para criação de valor para o cliente 58
Figura 23 – *Canvas* da Proposta de Valor 60
Figura 24 – *Canvas* do Modelo de Negócio 61
Figura 25 – *House of Quality* 66
Figura 26 – *House of Quality*, Projeto Têxtil 68
Figura 27 – Diagrama de Casos de Uso 72
Figura 28 – Casos de uso da gestão de cor 73
Figura 29 – Diagrama de sequência do sistema UC2.1 75
Figura 30 – Diagrama de sequência do UC2 76
Figura 31 – Diagrama de sequência referente à obtenção de todos os clientes 76
Figura 32 – Diagrama de sequência do sistema UC11 77
Figura 33 – Diagrama de sequência do UC11 78
Figura 34 – Diagrama de sequência referente à obtenção de todos os cartazes 79
Figura 35 – Diagrama de sequência do sistema UC15 80
Figura 36 – Diagrama de sequência do UC15 80
Figura 37 – Modelo domínio do sistema 84
Figura 38 – Diagrama de camadas 85
Figura 39 – Diagrama de Implementação 87

Figura 40 – Diagrama de Implantação 88
Figura 41 – *Stack* tecnológica do sistema 90
Figura 42 – Estruturação dos diretórios de código-fonte do projeto 91
Figura 43 – Dashboard da página *home* 93
Figura 44 – Responsividade do *dashboard* inicial da aplicação 94
Figura 45 – Interface na autenticação do utilizador 95
Figura 46 – Fragmento código da autenticação do utilizador 95
Figura 47 – Listagem de entidades do sistema 96
Figura 48 – Criar nova entidade no sistema 96
Figura 49 – Editar dados de entidade 96
Figura 50 – Eliminar entidade registada 97
Figura 51 – Componente reutilizável para a criação de tabela CRUD 97
Figura 52 – Implementação da lista de entidades 98
Figura 53 – Formulário de criação de referência 99
Figura 54 – Implementação do componente de seleção de dados 99
Figura 55 – Notificações de alerta 99
Figura 56 – Implementação da notificação de alerta 100
Figura 57 – Estrutura de Interface em TypeScript 100
Figura 58 – *Controller* da gestão de entidades 101
Figura 59 – Implementação chamada *post* através do *Axios* 102
Figura 60 – Distribuição de responsabilidade da API *Gateway* pelas funções Lambda 103
Figura 61 – Validação da autenticação numa função Lambda 104
Figura 62 - Modelo de dados relacional do sistema 105
Figura 63 – Fragmento de código que desencripta as variáveis de acesso 106
Figura 64 - Modelo de dados relacional 107
Figura 65 – Diagrama de estados do processo da abordagem TDD 108
Figura 66 – Modelo de testes *V-Model* 108
Figura 67 – Fragmento de código de teste unitário 109
Figura 68 – Modelo de testes V-Model 110
Figura 69 – Fragmento de código de teste unitário 110
Figura 70 – Opções resposta da escala *Likert-type* 115
Figura 71 – Diagrama de sequência do sistema do UC1.1 147
Figura 72 – Diagrama de sequência do UC1.1 147
Figura 73 – Diagrama da dependência getClientes do UC1.1 148

Lista de Tabelas

Tabela 1 – Tipo de tingimento de acordo com a fibra 11
Tabela 2 – Síntese comparativa entre BMS e ERP 14
Tabela 3 – Síntese das características dos *softwares* apresentados..... 19
Tabela 4 – Síntese comparativa entre arquiteturas Monolítica, Microsserviços e *Serverless*... 25
Tabela 5 – Plataformas *No-code* vs *Low-code*..... 29
Tabela 6 – Análise comparativa das tecnologias de *front-end* 33
Tabela 7 – Comparação *AWS Lambda* vs *Azure Functions* vs *Google Cloud Functions* 35
Tabela 8 – Síntese dos interpretadores de código..... 38
Tabela 9 – Arquiteturas *No-SQL*..... 40
Tabela 10 – Síntese comparativa SQL e NoSQL..... 40
Tabela 11 - Síntese comparativa das bases de dados estruturadas analisadas..... 43
Tabela 12 – Síntese comparativa das tecnologias de bases de dados não relacionais analisadas 47
Tabela 13 – Definição de conceitos no valor entregue ao cliente e a aplicabilidade no projeto 58
Tabela 14 – Requisitos Técnicos e de Cliente 68
Tabela 15 – Análise integral ao UC2.1..... 74
Tabela 16 – Análise integral ao UC11..... 77
Tabela 17 – Análise integral ao UC15..... 79
Tabela 18 – Tecnologias utilizadas e respetiva área 90
Tabela 19 – Ficheiros contidos nos diretórios do projeto..... 91
Tabela 20 – Métodos de solicitação HTTP 104
Tabela 21 – Testes de Qualidade de Software..... 114
Tabela 22 – Descrição individual dos Casos de Uso..... 144

Acrónimos e Símbolos

ACID	*Atomicity Consistency Isolation* Durability
API	*Application Programming Interface*
B2B	*Business to Business*
BJSON	*Binary JavaScript Object Notation*
BMC	*Business Model Canvas*
BMS	*Business Management Software*
BSD	*Berkeley Software Distribution*
CAP	*Consistency Availability Fault-Tolerant Partition*
CQL	*Cassandra Query Language*
CRUD	*Create Read Update Delete*
CSS	*Cascading Style Sheets*
DDB	*Distributed Database*
ERP	*Enterprise Resource Planning*
ETM	Empresa Têxtil da Maganha
FEI	*Front End of Innovation*
FFE	*Fuzzy Front End*
GPL	*General Public License*
HOQ	*House of Quality*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
iOS	*Internet Operating System*
JSON	*JavaScript Object Notation*
JSX	*JavaScript XML*
JWT	*JSON Web Token*
MVC	*Model View Controller*
MVCC	*Multi-Version Concurrency Control*
MVP	*Minimum Viable Product*
NCD	*New Concept Development*
NPD	*New Product Development*
OCDE	Organização para a Cooperação e Desenvolvimento Económico
PHP	*Personal Home Page*
Q&A	*Question and Answer*
QFD	*Quality Function Development*
ORM	*Object Relational Maping*
RDS	*Relation Database Service*
SAP	*Systems Applications and Products*
SOA	*Service Oriented Architecture*
SQL	*Structured Query Language*
TDD	*Test Driven Development*
TDS	*Tabular Data Stream*
UI	*User Interface*
VOC	*Voice of the Costumer*

Glossário

A

Acabamento	Processo químico da modificação das características de um tecido através do uso de corantes
Apache	Servidor web que disponibiliza conteúdo web pela internet

B

Back-end	Camada de uma aplicação de desenvolvimento web que corresponde à manipulação de dados, sendo esta invisível ao utilizador
Backup	Cópia de segurança dos seus dados (informações) de um dispositivo de armazenamento
Big data	Descreve uma grande quantidade de dados sendo estes estruturados ou não estruturados
Body	Corpo de uma estrutura
Brainstorm	Metodologia criadora de ideias que costuma ser aplicada em grupos
Browser	Navegador que possibilita aos utilizadores interagirem com documentos HTML hospedados em um servidor da rede
Bug	Erro ou falha numa aplicação que faz com que o seu fluxo tome um caminho inesperado ou errado
Business case	Caso de negócio que inicia o raciocínio para a criação de um projeto ou tarefa

C

Cartaz	Documento que dispõe as novas cores a serem criadas
Cloud	Fornecimento de vários tipos de sistemas pela internet, nomeadamente bases de dados, servidores, entre outros
Confeção	Processo de preparação de uma peça pronta a usar
Container	Espaços para execução de código que permitem configurações específicas para um único objeto
Cor	Formulação química que resulta numa determinada cor

D

Dashboard	Painéis que mostram métricas e indicadores importantes para alcançar objetivos e metas traçadas
Debug	Processo de análise de código com a finalidade de se validar o seu fluxo ou descobrir um bug
Delete	Operação que elimina um ou mais registos da base de dados
Desktop	Hardware do qual é constituído um computador fixo
Developer	Indivíduo com capacidade para desenvolver uma aplicação, por outras palavras, que tenha conhecimento sobre código

Developer-code	Modelo de desenvolvimento de aplicações em que envolve a escrita de código de ponta a ponta
Docker	Contentor que permite correr uma aplicação fora de um hardware local
Drag & Drop	Nomenclatura de interfaces do utilizador em que um dado elemento é arrastado para umas determinadas coordenadas à escolha do usuário

E

Encomenda	Encargo solicitado por um cliente à empresa
Endpoint	Centro de comunicação de uma rede por onde entra e sai informação
End-user	Pessoa que realmente usa um determinado produto
Ensaios	Tentativa de obtenção de uma determinada cor através de uma fórmula
Escalabilidade	Capacidade de potencial ou crescimento de um sistema
Estamparia	Processo de impressão de letras, imagens ou padrões no tecido
Event handler	Eventos disparados dentro da janela do navegador que notificam sobre mudanças no navegador

F

Feedback	Resposta de retorno após a avaliação de um produto ou conteúdo, com o propósito de obter informação útil para a sua evolução
Fiação	Processo de fabricação de fio têxtil
Framework	Plataforma de desenvolvimento de *software* da qual resulta uma aplicação estruturada
Front-end	Camada de uma aplicação de desenvolvimento web que corresponde à interface do utilizador
Full stack	Ferramenta de código ou desenvolvedor com competências para construir uma aplicação de ponta a ponta (*front-end* e *back-end*)
Functional Decomposition	Descrevem os passos necessários para o funcionamento de um processo

G

Github	Plataforma que guarda e partilha projetos de código além de controlar o seu versionamento

H

Header	Cabeçalho de uma estrutura

I

Insert	Operação que insere um novo registo da base de dados
ITMA	Congresso de exposição de máquinas têxteis e de vestuário

J

Join	Combinação de várias colunas ou mesmo tabelas numa dada operação

K

Know-how	Conhecimento e capacidade de saber ou conseguir fazer algo de forma correta

L

Lab-dip	Amostra de tecido tingido feita para atender aos padrões de cores.
Lavandaria	Conjunto de máquinas que tratam da lavagem e limpeza a seco de um tecido
Layout	Engloba elementos como texto, gráficos, imagens e a forma como eles se encontram em um determinado espaço
Logs	Ficheiro de texto, que recolhe cronologicamente todos os eventos que afetaram um sistema
Low-code	Modelo de desenvolvimento de aplicações no qual apenas é necessário a escrita de código esporadicamente e utiliza-se frequentemente um mecanismo de *drag & drop*

M

Microsserviço	Arquitetura de *software* em que os componentes são separados por módulos, tendo cada um a sua própria plataforma
Mock	Criação de uma versão falsa de um serviço externo ou interno que pode substituir o real
Monolítico	Arquitetura de *software* em que todos os componentes de uma aplicação se encontram numa única plataforma
Multi-thread	Processo que contém as instruções de execução de múltiplas sequências

N

No-code	Modelo de desenvolvimento de aplicações no qual não envolve escrita de código, mas sim um mecanismo de *drag & drop*

O

Open-source	*Software* de desenvolvimento disponibilizado à sociedade sem qualquer tipo de custos

P

Package	Conjunto de ficheiros que constituem um modulo de um projeto
Pantone	Escala de cor, usada por várias indústrias, em que cada cor se encontra identificada por uma combinação de números com letras
Pop-ups	Janelas que aparecem repentinamente no primeiro plano de uma interface visual

Q

Query	Solicitação à base de dados de consulta ou manipulação da informação

R

Renderizar	Processo pelo qual se obtém o produto final de um através de processamento digital
Request	Pedido de serviço
Response	Resposta de serviço
Route handler	Mapeamento do URL de uma solicitação com as rotas registadas.
Run time	Tempo de execução durante o ciclo de vida de um programa a decorrer

S

Select	Operação que retorna dados de uma base de dados
Sequential programming	Sequência de instruções realizadas sempre pela mesma ordem e que apresenta sempre os mesmos resultados
Serverless	Arquitetura de *software* em que os componentes são separados em funções e estas hospedadas num sistema cloud externo
Single-thread	Processo que contém as instruções de execução de uma única sequência
Software	Conjunto de serviços, programas, ou procedimentos que em conjunto constituem uma aplicação estruturada
String	Sequência de caracteres

T

Tecelagem	Processo de entrelaçamento dos fios têxteis de forma a criar o tecido
Technology Stack	Lista de todas as tecnologias necessárias para atingir um determinado produto
Template	Ficheiro ou conjunto de ficheiros que servem como ponto inicial para a construção de um documento ou projeto
Timers	Mecanismo automático para ativar um dispositivo em um momento predefinido
Tinturaria	Processo químico da modificação da cor de um tecido através do uso de corantes
Token	Permitem a criação e armazenamento de chaves privadas que permitem assinaturas digitais seguras
Trigger	Um evento que é despoletado automaticamente segundo uma regra

U

UNIREG	*Software* de gestão de base de dados
Update	Operação que altera um ou mais registos da base de dados
Use cases	Identificam as funcionalidades de uma aplicação com base nos requisitos do sistema

User-friendly	Característica de um *software* que incita que é de fácil uso ou compreensão
User stories	Idênticos aos casos de uso, no entanto são baseadas na interação do utilizador

V

Visual Studio Code	Ferramenta de edição de código
Value Proposition Canvas	Ferramenta de desenvolvimento que explora mais profundamente o seguimento de cliente e proposta de valor

W

Webinar	Apresentação organizada por uma empresa com o objetivo de apresentar o seu produto
Workflow	Sequência de tarefas que analisam e processam uma dada estrutura de dados

X

XHP	Extensão de PHP

1 Introdução

Este capítulo introdutório descreve o enquadramento do projeto no contexto em que se insere, descrevendo o problema, os objetivos a atingir, a abordagem idealizada, e por último, é apresentada a estrutura do documento.

1.1 Contexto

O surgir do conceito Indústria 4.0 neste setor exige por parte das empresas a procura de novas ideias, particularmente focadas em políticas de qualidade e sustentabilidade, e, por isso, obriga a que as tecnologias disponibilizadas no mercado acompanhem este nível de inovação, e correspondam aos requisitos propostos pelas organizações.

Nos últimos anos, a indústria têxtil em Portugal tem sido alvo de destaque pelo aumento contínuo das exportações. Desta forma, por consequência da pandemia, a economia encontra-se em reestruturação, e o têxtil vestuário ganha cada vez mais visibilidade pela habilidade de se reinventar, resistir às dificuldades, e apostar em novos métodos para se diferenciar.

Os fatores diferenciais entre as diversas organizações concorrentes, são essencialmente o preço, a qualidade e o prazo de entrega. Por isso, as empresas para sobreviverem recorrem ao mercado tecnológico para serem capazes de aumentar a sua capacidade de resposta.

"Na verdade, 'tecnologia' foi quase sempre sinónimo de 'avanço', e foi aplicada em quase todos os domínios da existência humana" (Venturi, 2014). A tecnologia tem um impacto profundo em diversos aspetos globais dos negócios e da indústria, e continua a fazê-lo, com novos métodos que elevam a fasquia da inovação e eficiência.

1.2 Problema

As empresas do setor da Indústria Têxtil e Vestuário (ITV), atravessam atualmente a Quarta Revolução Industrial, representando o aparecimento de uma onda de tecnologia que impulsiona a eficiência em todas as operações.

Desta forma, organizações resistentes à mudança, e que se oponham a adotar novos conceitos tecnológicos, colocam em causa o seu lugar no mercado, e sobrevivência a longo prazo.

No entanto, para as empresas que se dedicam ao tingimento e acabamento de têxteis, torna-se vital a digitalização da informação associada aos diversos processos produtivos, permitindo a rastreabilidade e acessibilidade aos dados.

Atualmente, as soluções direcionadas a este nicho de mercado não preenchem os requisitos necessários para estas empresas. Os produtos disponíveis foram desenvolvidos sobre estruturas monolíticas, altamente limitativas quanto à sua escalabilidade, e capacidade de manutenção.

Adicionalmente, os níveis de usabilidade são francamente baixos, devido às interfaces desenvolvidas em *frameworks* descontinuadas, e, por isso, dispõem-se de forma pouca organizada e intuitiva. Igualmente crítico, é a ausência do tratamento de dados, tendo em conta a complexidade e o número de variáveis inerentes ao processo produtivo deste tipo de empresas.

Assim, o desenvolvimento de um produto baseado em tecnologias atuais, que centralize os dados, e os converta em informação para o utilizador, auxilia a tomada de decisão. Consequentemente, aspetos como a eficiência de custos, a monotorização de produção, e a retenção de clientes, serão influenciados de forma positiva, indo de encontro ao que realmente estas organizações esperam de um *software* de gestão.

1.3 Objetivos

No âmbito deste projeto, pretende-se colmatar as vulnerabilidades das organizações de tinturaria e acabamento, através do desenvolvimento de um produto que centralize os dados das organizações, descartando limitações em integrações com *software* pertencente a maquinaria externa.

Desta forma, a arquitetura da solução deve ser altamente escalável, de forma a permanecer estável enquanto se adapta a mudanças, expansões do sistema e alterações de recursos. A nível de base de dados pretende-se relações com reduzido nível de redundância, possibilitando a manipulação da informação sem causar inconsistências de dados.

Adicionalmente, a usabilidade da aplicação, é outro aspeto a ter em conta, pelo que a ferramenta deve ser iminentemente gráfica e de fácil utilização. A disponibilização da criação e

personalização de *dashboards* é fulcral, no sentido de apresentar ao utilizador informação significativa num único espaço.

Em suma, pretende-se que os objetivos no desenvolvimento deste projeto sejam:

- Pesquisa bibliográfica e análise tecnológica de forma a idealizar uma solução orientada às necessidades de uma empresa têxtil, de modo a colmatar as vulnerabilidades dos setores de entrega e tratamento da encomenda proveniente de um cliente;

- Consumação da arquitetura das funcionalidades correspondentes aos módulos de gestão de encomendas e gestão de laboratório, desenhando uma solução que ultrapasse todas as preocupações transversais do seu desenvolvimento;

- Realização do desenvolvimento aplicacional de acordo com os módulos desenhados no ponto da realização da arquitetura;

- Deificação de um possível plano de manutenção da solução;

- Idealização de um possível plano de desenvolvimento dos restantes módulos da solução.

1.4 Abordagem

O desenvolvimento do presente projeto iniciou-se com um diagnóstico no terreno, presenciando a atividade diária das empresas do setor do têxtil vestuário, reconhecendo os problemas que afetavam o desempenho das mesmas. Este acompanhamento possibilitou identificar necessidades não atendidas e emergentes, e estipular requisitos necessários e comuns no espectro de organizações analisadas.

Deste modo, procedeu-se a uma análise da oportunidade mercado, analisando a oferta no mercado em Portugal na área do *software* de gestão, e posteriormente, procedeu-se com uma pesquisa bibliográfica, no sentido de idealizar um produto capaz de corresponder às necessidades apresentadas pelo setor.

"Um caso de estudo bem escrito acompanhará um cliente enquanto ele define um problema, determina uma solução, implementa-a e colhe os benefícios" (Cherry, 2019). No decorrer do projeto, foi aplicada a metodologia do caso de estudo, um método comum de análise quantitativa e qualitativa, que pressupõe na observação detalhada de um certo fenómeno, sendo constituída por cinco etapas:

- Reconhecimento do fenómeno a ser estudado e análise do seu estado atual;

- Análise do historial e recolha de associados para a análise do fenómeno;

- Identificar fatores que originem o problema, de forma a obter um ponto de partida para desenvolver soluções, e colmatar o problema;

- Aplicar as medidas corretivas desenvolvidas e analisar a recetividade do setor envolvido;

- Testar e determinar a eficiência das medidas aplicadas.

1.5 Estrutura do documento

O documento encontra-se dividido em 4 capítulos principais e uma secção direcionada a anexos. Os capítulos dispõem-se pela seguinte ordem:

- O primeiro capítulo, a introdução, inclui a descrição do enquadramento do projeto e a descrição do problema. Adicionalmente, são expostos os objetivos, a abordagem a seguir, e, por último, a estrutura do documento.

- O segundo capítulo, o estado da arte, consta a pesquisa bibliográfica e análise crítica às soluções presentes no mercado e tecnologias existentes no desenvolvimento de *software*.

- O terceiro capítulo, a análise de valor, detalha o processo de transformação de uma ideia em um produto, apresentando a proposta de valor do produto e modelo de negócio associado ao projeto

- O quarto capítulo, análise de requisitos e conceção, apresenta os requisitos funcionais e não funcionais da aplicação. Seguidamente, demonstra-se a análise e conceção da solução, evidenciando as relações entre conceitos e as camadas do sistema.

- O quinto capítulo, implementação, apresenta todo o processo de desenvolvimento do sistema, e as tecnologias utilizadas em cada uma das camadas do sistema. Adicionalmente, aborda-se os diferentes tipos de testes realizados à aplicação.

- O sexto capítulo, a experimentação e avaliação, contém os testes de hipóteses possíveis no contexto do sistema de gestão têxtil e especifica as metodologias de avaliação a utilizar.

- O sétimo capítulo, a conclusão do documento, identifica os objetivos concretizados ao longo do projeto, bem como as limitações encontradas. Para finalizar, aborda-se o trabalho a realizar no futuro.

2 Estado da Arte

O presente capítulo apresenta a pesquisa bibliográfica realizada, apresentando inicialmente o domínio do problema, seguido de uma análise aos produtos disponibilizados no mercado, assim como as suas tecnologias em projetos com objetivos semelhantes. Posteriormente, são abordados temas intrínsecos ao desenvolvimento de uma solução de raiz.

Começando pela análise das diferentes categorias de *software* orientados à gestão de organizações, com o objetivo de demonstrar as diferenças e clarificar as funcionalidades inerentes ao presente projeto.

Seguidamente, são apresentadas as soluções existentes no mercado, descrevendo as características individuais e tecnologias associadas ao desenvolvimento, finalizando com uma abordagem superficial à oportunidade identificada para o desenvolvimento de um novo produto.

Deste modo, dados os requisitos arquiteturais do projeto, introduzem-se conceitos que representam diferentes tipos de estrutura de sistemas, nomeadamente Monolítico, Microsserviço e *Serverless* e, baseado no enquadramento do projeto, resulta a análise da solução que apresenta maior conformidade com os objetivos.

Por conseguinte, é realizada uma análise minuciosa às tecnologias disponíveis para o desenvolvimento de *software*, desde ambientes de desenvolvimento *no-code* e *low-code* e, posteriormente, *frameworks* dedicadas à produção de código, dispondo as diferenças entre cada, com recurso a tabelas de síntese.

Assim, segue-se a análise comparativa das diferentes estruturas de base de dados, tanto relacionais como não relacionais e, por último, são apresentadas diferentes plataformas de armazenamento *cloud*.

Nesta pesquisa realizada a informação apresentada é acompanhada das respetivas referências bibliográficas, assim como uma síntese comparativa de cada tema abordado.

2.1 Domínio do problema

Esta secção apresenta o enquadramento teórico do setor têxtil vestuário, mencionando todos os aspetos necessários para proceder com a análise de requisitos, e assim compreender totalmente os conceitos envolvidos, os dados de entrada e saída dos seus processos, a fim de cumprir com os objetivos estipulados para o produto a desenvolver.

Inicialmente, serão demonstrados conceitos gerais da indústria do têxtil vestuário, com o propósito de detalhar o subsetor dedicado ao tingimento e acabamento. Seguidamente, a secção termina com uma menção ao laboratório, e a sua relevância para as organizações pertencentes ao subsetor abordado.

2.1.1 Conceitos gerais da indústria têxtil vestuário

A indústria do têxtil, responsabiliza-se pela transformação de fibras em um produto acabado, podendo ser este destinado a peças de vestuário, têxteis-lar, como tapetes, cortinas e roupa de cama, e a tecidos técnicos, como por exemplo, uma farda de bombeiro.

A complexidade do processo produtivo, subdivide-se em fases distintas, em que para cada uma delas requer um grau de especialização elevado.

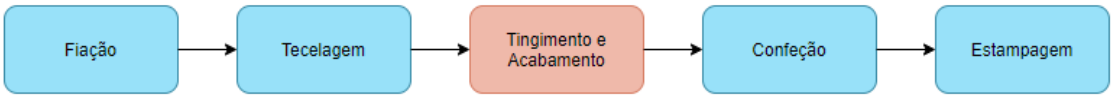


Figura 1 – Processo produtivo da transformação na indústria têxtil

Na Figura 1, detalha o fabrico de fibras têxteis, utilizadas como matéria-prima, até a obtenção do produto final. Cada elemento dispõe da sua função:

- Fiação: Consiste na transformação mecânica de fibras têxteis no estado em rama, com determinada espessura e resistência (Silva, 2018).

- Tecelagem: Compõe a estrutura têxtil obtida a partir do cruzamento de fios horizontais e verticais, de maneira a construir um tecido ou malha. Estes diferem-se consoante a técnica utilizada no entrelaçamento de fios.

- Tingimento e Acabamento: O tingimento trata-se do processo químico de modificação da cor do tecido / malha através da utilização de corantes. O acabamento caracteriza-se pela aplicação de processos físicos e químicos sobre o tecido / malha, nos quais o emprego de água e vários produtos químicos geram efeitos de limpeza, branqueamento, coloração, modificação de toque e funcionalidades ao produto final (Peixoto & Mendes, 2012).

- Confeção: Focada no *design*, define a forma, as linhas e os acessórios das peças de cada coleção.

- Estampagem: "Impressão" de desenhos a uma ou mais cores sobre um substrato têxtil. Além da valorização estética, e efeitos de brilho e texturas, a estamparia possibilita ainda a incorporação de propriedades funcionais (retardante de chama, antiderrapante, refletor, …).

Atualmente, Portugal tem cerca de seis mil empresas a trabalhar em todos os subsetores da indústria têxtil, onde segundo a Associação Portuguesa do Têxtil Vestuário, destaca-se pelo *deal time*, isto é, menor tempo de entrega do processo produtivo da Europa. Adicionalmente dispõe de tecnologia para produzir vestuário de qualidade, e mão de obra permite uma produção a um custo inferior ao da maioria dos seus congéneres europeus (Dugal, 2021).

No entanto, o destaque na Figura 1 no processo de "Tinturaria e Acabamentos" contextualizam o mercado alvo do produto idealizado no presente projeto.

2.1.2 Indústria tinturaria e acabamentos

O fluxo apresentado na secção 2.1.1 evidencia os subsetores inerentes ao processo de transformação da matéria-prima na indústria do têxtil vestuário. Em cada um dos subsetores verificam-se diferentes níveis de complexidade, onde as organizações dedicadas à tinturaria e acabamentos têxteis estão expostas a um elevado número de variáveis no decorrer do seu processo produtivo, daí a procura por soluções personalizadas.

Adicionalmente, o tingimento e o acabamento traduzem-se nos processos com maior peso na cadeia de valor do produto.

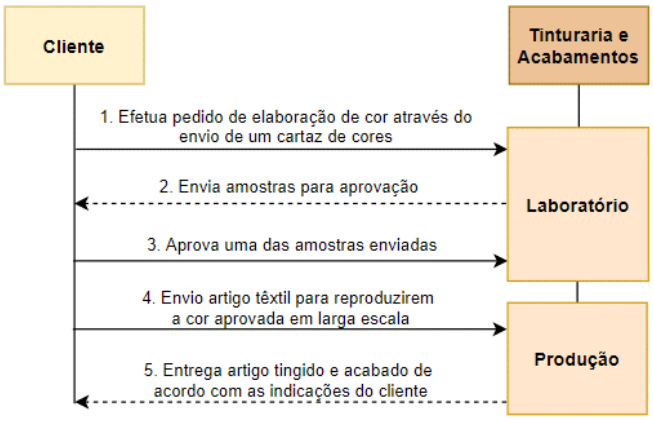


Figura 2 – Conceitos inerentes à indústria de tinturaria e acabamentos e respetivas interações

A Figura 2 apresenta resumidamente o fluxo associado a uma encomenda. O processo inicia-se por um pedido de abertura de cor por parte do cliente, ou seja, partindo do pressuposto que a empresa de tinturaria e acabamentos se trata de uma prestadora de serviços, o cliente encarrega-se de indicar a cor do produto que pretende, e a empresa limita-se a reproduzi-la.

Deste modo, o laboratório executa múltiplas tentativas de replicação da cor indicada pelo cliente. Posteriormente, são enviadas pelo menos duas amostras resultantes das experiências realizadas, com a finalidade de obter uma aprovação do cliente.

De seguida, o processo passa pela aprovação de uma única amostra por parte do cliente. Caso este considere que uma delas compatibilize com as suas necessidades, terá as condições reunidas para proceder com uma encomenda, que procederá com a reprodução da cor da amostra aprovada em larga escala.

Assim, realizado o pedido de encomenda pelo cliente, a organização procede com o tingimento e acabamento, e posteriormente a entrega do produto.

2.1.2.1 Intervenção do laboratório

O crescimento progressivo da industrialização, elevou os níveis de complexidade dos produtos de vestuário, obrigando as organizações a um estudo prévio aprofundado. O laboratório representa o ponto de partida para todas as encomendas recebidas por uma organização.

No que toca à intervenção das empresas do subsetor de tinturaria e acabamentos na manufatura do produto final, o laboratório tem capacidade para reproduzir duas fases do processo produtivo:

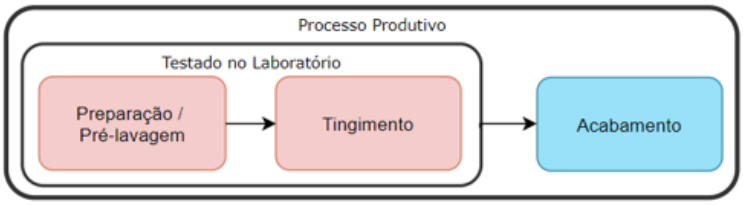


Figura 3 - Fases do processo produtivo no tingimento e acabamento

Ademais, o tingimento representa a fase de maior complexidade no fluxo apresentado na Figura 3, devido ao elevado número de variáveis envolvidos no processo. Desta forma, a testagem e elaboração da fórmula previamente em laboratório, antecipa possíveis problemas que podem ocorrer aquando realizada a produção em larga escala, evitando assim reprocessamentos, que resultam em custos para a organização.

Deste modo, a secção 2.1.2 evidenciou que todo o processo se inicia por um pedido do cliente ao laboratório. A informação é enviada sob o formato de um cartaz.

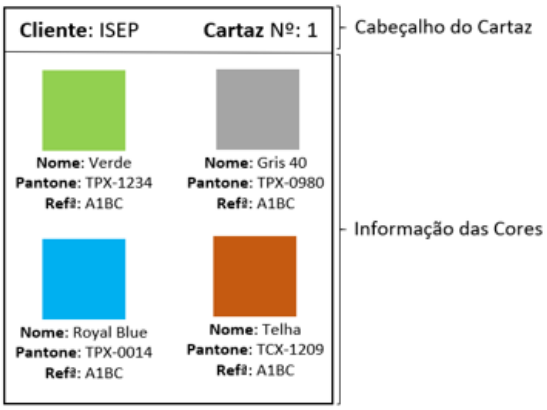


Figura 4 - Estrutura de um cartaz de cores enviado pelo cliente

A Figura 4 apresenta o cabeçalho do cartaz, identificado pelo nome da empresa que faz a solicitação, e por um número que identifica o cartaz de forma única para o cliente. Da mesma forma que as marcas disponibilizam nas suas lojas os mesmos produtos com cores distintas, o cliente usualmente envia múltiplas cores para serem reproduzidas.

Adicionalmente, para as cada uma das cores dispostas no cartaz, o cliente fornece as seguintes informações:

- Nome: Designação atribuída pelo cliente àquela cor.
- *Pantone*: Codificação universal que representa uma tonalidade específica (Basiel, 2021).
- Referência (Refª): Identifica inequivocamente a matéria-prima (tecido ou malha) que vai ser fornecida pelo cliente, sendo esta caracterizada pela sua estrutura e fibras que a compõe.

Apesar de todas as cores partilharem a mesma referência, o laboratório independentemente cada cor, a fim de criar a sua própria fórmula, originando assim o *lab-dip*, que se define pelos atributos mencionados anteriormente (nome, *pantone* e referência), mas adicionalmente estabelece:

- Tratamento Prévio / Preparação: Conjunto de operações que a matéria-prima será submetida de forma a estar apta a ser tingida. Trata-se essencialmente da eliminação de impurezas para melhorar a qualidade do substrato têxtil.
- Tipo de Tingimento: Gama de produtos utilizados consoante a(s) fibra(s) a tingir.
- Relação de Banho: Define a relação entre a massa do banho de tingimento e a quantidade de matéria a tingir (exemplo: RB = 1:10, significa que 1 kg tecido / malha para 10 litros de banho).

A relação de banho define o principal motivo da evolução das máquinas de tingir, uma vez que assim se diminui o consumo energético, o consumo de alguns produtos auxiliares, e o consumo de água.

Num contexto de laboratório, as máquinas que realizam o tingimento em pequena escala, e por isso, habitualmente dispõem de capacidade para 50 ml de banho, sendo que a relação de banho se traduz da seguinte forma:

$$Relação\ de\ Banho = \frac{Peso\ da\ Amostra\ (g)}{Quantidade\ de\ Banho\ (ml)}$$

Seguidamente, estando definido o *lab-dip*, associam-se as diversas fórmulas elaboradas com o propósito de reproduzir a cor solicitada pelo cliente. No ambiente de laboratório, estas tentativas são denominadas de ensaio, onde *lab-dip* agrega múltiplos ensaios

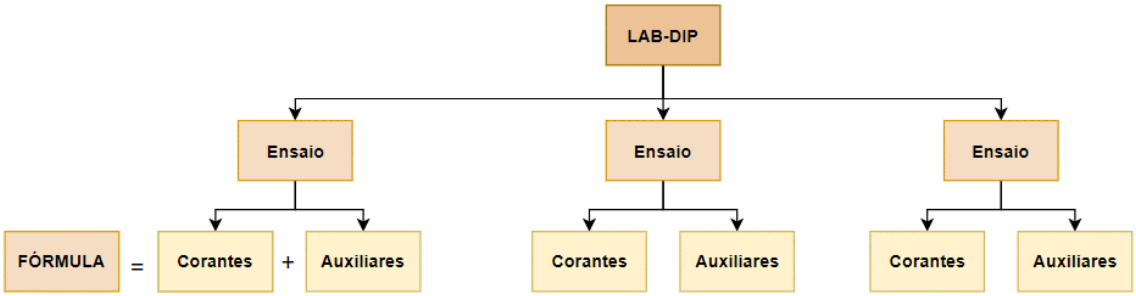


Figura 5 – Constituição de um *lab-dip*

Na Figura 5 especifica que a fórmula de um ensaio é constituída por dois tipos de produtos: corantes e auxiliares, que desempenham papeis distintos no processo de tingimento.

Adicionalmente, um *lab-dip* tem em média 5 a 6 ensaios associados, por isso, trata-se cada um deles de forma independente, em que individualmente detêm o seu estado.

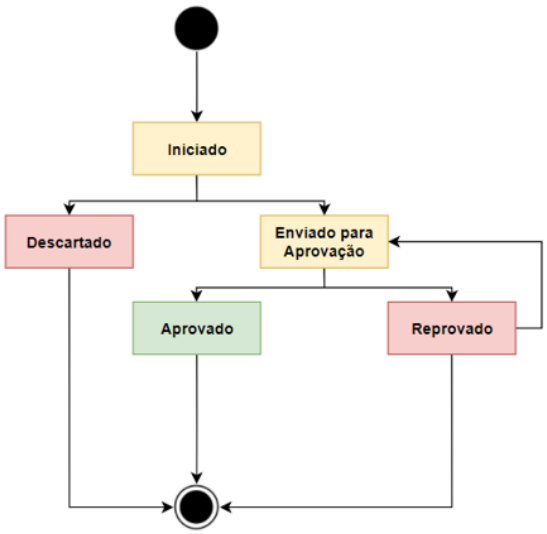


Figura 6 – Estados de um ensaio de laboratório

O estado “Iniciado” como demonstrado na Figura 6 aplica-se a todos os ensaios criados no sistema. Posteriormente, dependendo do resultado da fórmula testada, o mesmo pode ser enviado ao cliente, ou descartado. Por fim, caso o *feedback* do cliente seja negativo, procede-se com ajustes na fórmula. Caso seja positivo, a combinação química utilizada nesse ensaio refletirá a fórmula utiliza em produção de larga escala.

Na informação mencionada aquando da criação do *lab-dip*, define-se o tipo de tingimento, onde este varia consoante o substrato têxtil que se pretende tingir, e consequentemente determina a gama de corantes a ser utilizada. As várias fibras provêm de origens distintas:

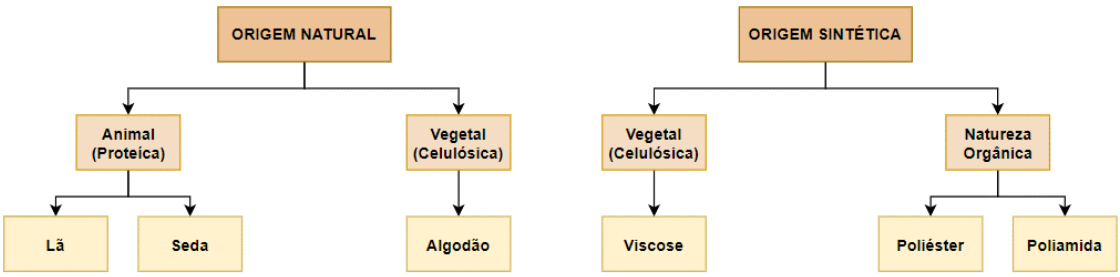


Figura 7 – Classificação das fibras quanto à sua origem

A Figura 7 apresenta algumas das fibras que podem constituir um artigo têxtil. A referência que se define no *lab-dip* identifica inequivocamente uma determinada composição, devido à possibilidade de ocorrerem junções de fibras no mesmo artigo (Durães, 2020).

As misturas de fibras dentro do mesmo artigo, conferem uma melhoria de qualidade do substrato têxtil, combatendo desta forma algumas desvantagens de cada fibra quando utilizada a sós.

Deste modo, possibilita corrigir imperfeições fazendo as combinações necessárias, adicionando percentagens especificas de determinadas fibras com um objetivo pretendido, nomeadamente, a criação de efeitos de cor, a melhoraria da eficiência da fiação, reduzir custos, mistura, conferir propriedades absorventes, entre outras.

A escolha entre as fibras naturais e as sintéticas depende do objetivo do produto final. As fibras naturais adicionam conforto, são biodegradáveis e renováveis. No entanto, aumenta o teor de degradação da peça. Por outro lado, as fibras sintéticas foram criadas com o intuito de facultar aos consumidores uma secagem rápida, maior resistência e durabilidade.

Desta forma, com diferentes características e níveis de absorção em cada fibra, os tipos de tingimento variam entre elas:

Tabela 1 – Tipo de tingimento de acordo com a fibra

		Tipo de Tingimento					
		Ácidos	Diretos	Cuba	Dispersos	Reativos	Pigmento
Fib	Algodão	✗	✗	✓✓	✓✓	✓✓	✗
	Viscose	✓✓	✓✓	✗	✗	✓	✗

	Lã	✔✔	✔✔	✔	✔	✘	✔
	Seda	✘	✘	✘	✘	✔✔	✔✔
	Poliamida	✔✔	✔✔	✘	✘	✔	✘
	Poliéster	✔	✔	✔	✔	✔	✔

Legenda: ✔✔ Muito adequado; ✔ Adequado em contextos específicos; ✘ Incompatível

O conceito de tipo de tingimento, traduz-se numa gama de corantes, ou seja, para ocorrer o tingimento do algodão e do poliéster são utilizados produtos distintos, tanto corantes como químicos auxiliares. Deste forma, permite que as fibras apesar de poderem estar misturadas, sejam tingidas de forma praticamente independente.

A razão da existência de um grande número de corantes comerciais, está relacionada com os inúmeros fins do produto final, onde aspetos como desgaste da cor na lavagem, na presença de luz ou na fricção, são influenciados pela qualidade dos corantes utilizados (Sweeney, 2018).

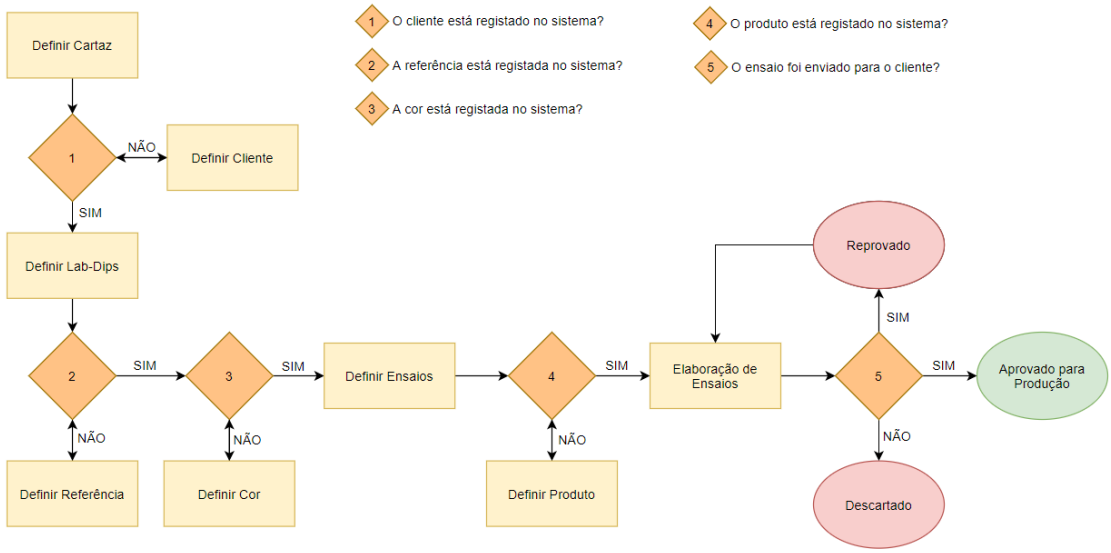


Figura 8 – Fluxograma do processo de abertura de nova cor num laboratório têxtil

Em suma, partindo do momento em o laboratório recebe um cartaz proveniente do cliente, a interação com o sistema procede-se da forma indicada na Figura 8, onde agrega toda a informação detalhada na subsecção, mencionando os conceitos inerentes ao procedimento habitual de reprodução de uma nova cor a pedido do cliente. Desta forma, permite obter uma visão geral da hierarquia e relação entre cada um dos termos abordados.

2.2 Tecnologias de Gestão de Organizações

"A tecnologia é um tipo de conhecimento ou habilidade utilizado para empregar e controlar fatores de produção que, por sua vez, podem levar à saída de produtos e serviços." (Sharma K. , 2015). A tecnologia revolucionou a maneira como as empresas conduzem os seus negócios, onde métodos de gestão básicos são alterados e inseridas em paralelo novas habilidades para

a coordenação da organização, onde a presença da tecnologia torna-se vital no momento da tomada de decisões (Huang, 2018).

O mercado atual disponibiliza um vasto número de sistemas de gestão, onde cada um procura posicionar-se como a solução ideal para as organizações, oferecendo benefícios tangíveis e intangíveis, que resultam no aumento da eficácia na produção e dos seus custos associados, permitindo que as pequenas empresas se igualem a organizações maiores (Perkins, 2020).

2.2.1 Soluções Genéricas

Atualmente, atendendo à variedade de *softwares* existentes, os mesmos podem ser divididos em duas grandes categorias: *Business Management Softwares* (BMS) e *Enterprise Resource Planning* (ERP). Ambos funcionam como sistemas baseados em transações que reúnem dados, com a finalidade de transmitir informação ao utilizador e auxiliar na tomada de decisão através do conhecimento armazenado no sistema (Perkins B. , 2020).

2.2.1.1 Business Management Softwares

Os *Softwares* de Gestão de Negócios (BMS), focam-se em auxiliar o acompanhamento e a automatizar os processos de produção através da sua versatilidade de integrar com outros sistemas, permitindo, assim, centralizar a informação. Deste modo, promove a clareza na comunicação entre setores e na exposição da informação. Adicionalmente, intervém em áreas inerentes ao processo produtivo, nomeadamente a gestão da cadeia de suprimentos e de subcontratados (Lebeaux, 2014).

2.2.1.2 Enterprise Resource Planning

Por outro lado, os Sistemas de Planeamento de Recursos (ERP), atendem a outras vertentes de uma organização, sendo projetados para serem uma plataforma única de gestão de recursos de uma organização. Por consequência, alarga-se a áreas como a contabilidade, recursos humanos e distribuição, cobrindo processos operacionais de ponta a ponta (Radley, 2017).

Em suma, considera-se que um produto ERP tem condições para ser usado como um BMS, mas nem todos os produtos BMS têm funcionalidades suficientes de forma a desempenhar-se como um ERP competente. Apesar de as diferenças tenderem a ser cada vez menos percetíveis, ou seja, a categorizar um *software* de um dos tipos apresentados, pode ser falacioso, tendo em conta que se trata de termos ambíguos e de elevada abrangência. Portanto, é frequente que um sistema não se enquadre na perfeição com nenhuma das categorias, isto é, o mesmo pode possuir características de ambos os tipos (Bernshteyn, 2018).

2.2.1.3 Comparação Business Management Software vs Enterprise Resource Planning

Na Tabela 2 verificam-se as comparações entre um BMS e ERP, que visa as diferenças nas áreas de trabalho que incorporam, onde se evidencia um grau de complexidade superior do ERP em relação ao BMS.

Tabela 2 – Síntese comparativa entre BMS e ERP

	BSM	ERP
Monotorização de Encomendas Monitorizar fluxo e estado atual de encomendas	✔	✔
Gestão de Suprimentos Lista de stocks de produtos, criação de ordens de compra, …	✔	✔
Gestão de Distribuição Lista de stocks, arquivo de orçamentos, criação de ordens de compra, …	✔	✔
Gestão de Maquinaria Monitorizar estado e informação de máquinas	✔	✔
Manutenção Preventiva Calendarização de manutenções e reparações	✘	✔
Monotorização de Energia Medidores de leitura em tempo real do consumo de energia	✘	✔
Relatórios e Análises métricas (KPI's) Criação de relatórios de elevado nível de especificidade	✔	✔
Gestão de Recursos Humanos Medidores de leitura em tempo real do consumo de energia	✘	✔
Contabilidade / Faturação Medidores de leitura em tempo real do consumo de energia	✘	✔
Personalização da Configuração Adequação do sistema às necessidades individuais da empresa	✔	✔

Fonte: (Burnson, 2016)

A escolha de um determinado sistema reflete as necessidades sentidas por uma organização, mas neste contexto as escolhas centram-se nos requisitos e imprescindibilidade para as empresas de um setor específico.

No contexto do trabalho em curso, considera-se que o desenvolvimento de um produto abrangente como um *ERP* não colmata as necessidades específicas do nicho de empresas de tinturaria e acabamentos. Na verdade, a oferta de *softwares* existe, mas o que proporciona a oportunidade de entrada no mercado é a ausência de flexibilidade e especialização nas variáveis específicas do processo produtivo (2.2.2).

Portanto, é usual que as empresas integrem sistemas diferentes para a produção, faturação e gestão de recursos humanos, ao invés de disporem de um único sistema. Este fator evidencia a procura por parte das organizações por uma solução que seja capaz de planear, adaptar e ser altamente eficaz e interventiva no processo produtivo, sendo a abrangência do *software* a outras áreas um aspeto secundário.

2.2.2 Soluções de Mercado Para a Área Têxtil

Nesta secção são apresentadas três soluções de *software* distintas, que atualmente abrangem um elevado número de empresas de tinturaria e acabamentos na região Norte do país, detalhando as características, funcionalidades, benefícios e vulnerabilidades de cada uma. A informação foi reunida através da presença em *webinars* de propaganda de produtos e *feedbacks* obtidos nas deslocações presenciais às organizações.

Deste modo, através de uma síntese comparativa, é adotada uma postura crítica entre as análises realizadas a cada um dos produtos. Posteriormente, apresenta-se um caso de estudo para cada um dos *softwares* mencionados, aprofundando temas como o processo de implementação, a intervenção no processo produtivo e os aspetos positivos e negativos.

2.2.2.1 Texplus

O *software Texplus* detido pela empresa espanhola *Datamonplus*, é um produto desenvolvido unicamente direcionado a empresas do têxtil vestuário, em que numa fase embrionária dedicava-se exclusivamente a empresas de tinturaria e acabamentos e, posteriormente expandiu a sua abrangência à produção vertical, incluindo fiação, tecelagem, tinturaria, acabamentos, estamparia, confeção e lavandarias (Datamonplus, 2020).

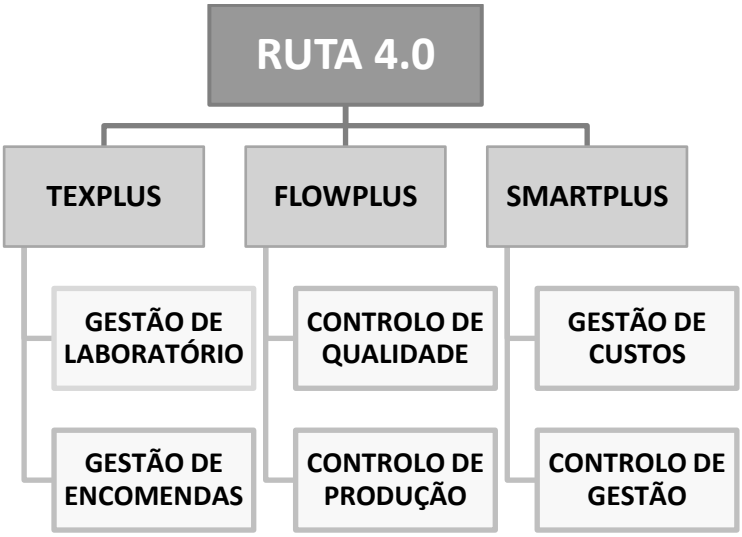
O produto dispõe de uma interface obsoleta (Anexo A: Interface Utilizador Texplus

), desenvolvida em Visual FoxPro que se trata de uma linguagem de programação orientada ao objeto, que oferece um ambiente para o desenvolvimento de aplicações desktop para o Windows. No entanto, foi descontinuada em 2007 e, atualmente, a *Microsoft* já não oferece suporte (Wood, 2020).

A nível de armazenamento, adotam uma base de dados relacional, mais concretamente, adotam o SQL c0k0 linguagem de definição e manipulação de dados. A organização que adquire o *software* está na obrigação de assumir o custo de compra e manutenção do servidor, que obrigatoriamente está unicamente dedicado para a base de dados e informação do *Texplus*.

No ano de 2020, a *Datamonplus* apresentou o seu último projeto "*Ruta 4.0*", onde, através da participação na propaganda do produto, a estrutura deste é apresentada na Fonte:

Figura 9, onde são detalhados os diversos módulos pelos quais se divide.



Fonte: (Datamonplus, 2020)

Figura 9 – Subdivisão dos módulos do *software Texplus*

O projeto constitui-se por vários módulos, em que cada um deles se complementam, mas que são adquiridos separadamente. Ainda oferece a migração da interface para um ambiente web, através do desenvolvimento da aplicação numa plataforma *low-code* denominada *Genexus 16*.

2.2.2.2 Cenário da Adoção do Texplus pela ETM

A Empresa Têxtil da Maganha (ETM), sedeada na Trofa, dedica-se à prestação de serviços de tingimento e acabamento e foi das primeiras empresas em Portugal a implementar o *Texplus*, mais precisamente no ano de 2000, e que acompanha a empresa até à atualidade. O processo de implementação prolongou-se por mais de um ano, apesar de o *software* nesse momento dispor apenas de funcionalidades básicas (ETM, 2021).

Atualmente, o produto intervém no fluxo de todas as encomendas que entram na empresa, onde o acompanhamento entre fases produtivas é igualmente realizado, até ao momento da sua saída da empresa, criação de documentos de transporte e faturação inclusive.

Adicionalmente, o *Texplus* não abrange mais de metade dos setores da empresa, devido a limitações na sua integração com *softwares* externos, limitando esses mesmos departamentos a realizarem os seus registos de dados diariamente em folhas de cálculo Excel.

Deste modo, a ETM mencionou o interesse na procura de uma nova solução para a gestão da empresa, tendo em conta que o suporte da *Datamonplus* não tem correspondido às expectativas, não disponibilizando apoio em Portugal. As alterações solicitadas no *software* resultam frequentemente em novos problemas e, consequentemente, perdeu a confiança na empresa, não avançando com a aquisição do novo projeto mencionado anteriormente "*Rota 4.0*".

2.2.2.3 4Tex

O *4Tex* iniciou a sua entrada no mercado do setor têxtil, focando-se numa divisão fulcral para as organizações de tinturaria e acabamentos, o laboratório de cores, estabelecendo parceria com a empresa *Colormetrix*. Esta organização dedica-se à criação de equipamento direcionado a este departamento, proporcionando, assim, um produto que, além de armazenar as formulações criadas em laboratório, integra com os restantes departamentos produtivos (Colormetrix, 2015).

O *software* foi desenvolvido em Visual Basic 6, que possui um ambiente de desenvolvimento integrado (IDE) gráfico, que facilita a construção de interface das aplicações (Anexo B: Interface Utilizador 4Tex

). A linguagem de programação é totalmente orientada ao objeto, implementada em .NET (Kiong, 2008). A nível de base de dados, aplica-se a mesma lógica que no Texplus, ou seja, estrutura relacional SQL, onde a informação armazena-se num servidor local dedicado ao sistema.

Considerando a antiguidade do ambiente de desenvolvimento, a *Microsoft,* por várias, vezes já manifestou a intenção em descontinuar o suporte, que permanece até à atualidade, sendo igualmente um fator limitativo, tendo em conta que há dependência por parte do *4Tex* de várias *framework* descontinuadas, que obriga a que novos desenvolvimentos sejam realizados unicamente em *Windows XP*.

2.2.2.4 Cenário da Adoção do 4Tex pela Carvitin

A Carvitin, empresa de tinturaria e acabamentos, localizada em Barcelos, no distrito de Braga, representa a nível nacional umas das áreas de maior densidade de indústria têxtil. A organização, desde a sua criação em 2005, dispunha do *4Tex* como sistema para a gestão de produção, onde era definido o planeamento e ordenação de prioridades em cada máquina.

No ano de 2019, decidiu abdicar do seu *software* para substituir pelo *Texplus*, justificando a troca com diversos fatores, nomeadamente, a ausência da componente de faturação e a falta de listagens e tratamento de dados, funcionando apenas como um produto que armazena dados, mas que não converte em informação para o utilizador. Por último, a descontinuidade na evolução do *4Tex*, tendo em conta que é um produto gerido por uma única pessoa, incita que a mesma tem intenção de não dar continuidade ao projeto.

Assim, esta recente mudança de *software* por parte da Carvitin, foi um processo que durou mais de um ano, onde foi sentida muita resistência à mudança por parte dos colaboradores da empresa. A organização desvalorizou as dificuldades de adaptação à interface e usabilidade do *software*, tendo em conta que a aquisição do Texplus ocorreu com o principal objetivo de, posteriormente, avançar com a implementação do projeto da Datamonplus "*Ruta 4.0*", caso contrário a empresa estaria a fazer a troca de um *software* considerado ultrapassado, por outro equivalente.

2.2.2.5 MacWin

A *MacWin* dedica-se ao desenvolvimento de soluções de *software ERP*, bem como à prestação de serviços informáticos especializados, que abrangem diversas áreas de atividade. No entanto, a organização, desde a sua criação, especializou-se na Indústria do Têxtil e do Vestuário (ITV).

Deste modo, 80% da carteira de clientes pertence ao setor ITV e, por isso, os projetos implementados e a multiplicidade de soluções oferecidas, focam-se neste setor de maior relevância (MacWin, 2021).

Atualmente, a *MacWin* dispõe do seu produto GM Têxtil, que se subdivide em diferentes módulos, sendo cada um especializado num determinado tipo de produção, dentro do setor do têxtil vestuário.

Fonte: (MacWin, 2021)

Figura 10 – Subdivisão do *software ERP* da empresa *MacWin**Fonte:*

Figura 10A Figura 10, apresenta os vários módulos que compõem o GM Têxtil, em que cada um deles pode ser adquirido e implementado de forma independente. No caso de se tratar de uma empresa de produção vertical, ou seja, que engloba o processo produtivo desde a tecelagem, passando pela tinturaria e terminando na confeção, podem ser integrados e operar juntos.

2.2.2.6 Cenário da Adoção do MacWin pela Pocargil

O grupo Pocargil, integra um projeto industrial do setor têxtil com mais de 30 anos de experiência. Ao contrário dos casos de estudo anteriormente apresentados, trata-se de uma empresa em que a sua produção abrange dois setores distintos: tinturaria e confeção.

Na última década de 2010, o grupo dispunha de *softwares* de gestão de produção distintos em cada um dos setores, sendo *MacWin* para a área de confeção, e 4Tex na produção de tinturaria.

Recentemente, a organização procedeu com uma transformação tecnológica no setor da tinturaria, procedendo com *upgrade* de máquinas, integração de grupos de máquinas automatizadas e alteração no *software* de gestão.

Primeiramente, a Pocargil tencionava uniformizar ambos os setores, integrando o MacWin. Este cenário, apesar de aparentemente ser a solução mais lógica, foi descartado devido à organização considerar que o módulo de "Gestão de Tinturaria" se encontrava prematuro, não contendo ainda uma componente de gestão associada ao laboratório.

Adicionalmente, a empresa teve em conta o tempo de implementação e os custos associados à instalação do MacWin na sua área da confeção que, segundo indicou, as despesas pós-compra em serviços de personalização não indenizavam, correspondendo ao triplo do valor pago pelo produto inicialmente. Assim, a decisão final recaiu sob o Texplus que, entre as diversas opções de mercado, aos olhos da organização, atualmente apresenta:

- Valor comprovado no mercado;
- Integra os vários departamentos que constituem o setor da tinturaria;
- Inclusão da faturação (Fator complementar).

2.2.2.7 Síntese do estudo de softwares de gestão de tinturarias e acabamentos

A Tabela 3 sintetiza as diferenças entre os diferentes *softwares* apresentados, destacando os principais fatores diferenciais, que correspondem a funcionalidades características deste tipo de produtos.

Tabela 3 – Síntese das características dos *softwares* apresentados

	TEXPLUC	**4TEX**	**MACWIN**
FUNCIONALIDADES			
Monotorização de Encomenda	✔	✔	✔
Gestão de Suprimentos	✔	✔	✔
Gestão de Distribuição	✘	✔	✔
Gestão de Maquinaria	✔	✘	✔
Manutenção Preventiva	✘	✘	✘
Monotorização de Energia	✘	✘	✘
Relatórios e Análises Métricas (KPI's)	✘	✘	✘
Gestão de Recursos Humanos	✘	✘	✔
Contabilidade / Faturação	✔	✘	✔
Personalização da Configuração	✔	✔	✔
Módulo Gestão Laboratório	✔	✔	✔
ARQUITETURA / TECNOLOGIAS			
Arquitetura do *Software*	Monolítica	Monolítica	Monolítica
Linguagem de Base de Dados	SQL	SQL	SQL
Tecnologias Desenvolvimento	Genexus 16	VisualBasic6	?
PLATAFORMAS / IDIOMAS			
Windows	✔	✔	✔
Aplicativo Mobile	✘	✘	✔
Português	✘	✔	✔
SUPORTE DE UTILIZADORES / INFRAESTRUTURA			
Capacidade de Utilizadores	P M G (*1)	P M G (*1)	P M G (*1)

	TEXPLUC	4TEX	MACWIN
Suporte Cliente	✔	✔	✔
INFORMAÇÃO DE PREÇOS			
Preço de Aquisição do *Software*	55.000€ (*2)	30.000€ (*2)	65.000€ (*2)
Preço por Terminal	1.200€ (*2)	NA	NA

(*1) Capacidade: **P** – Pequena (Inferior a 50 utilizadores); **M** – Média (50 a 1000 utilizadores); **G** – Grande (Superior a 1000 utilizadores)

(*2) Preço médio com base na informação fornecida por empresas distintas, incluindo apenas o módulo base no caso do Texplus e MacWin

Os valores associados à compra apresentados para cada um dos *softwares* não incluem custos relacionados com a compra do servidor necessário para a base de dados que, segundo os requisitos exigidos pelos produtos, varia em média entre os 7.000€ e os 10.000€, somando ainda a mensalidade paga para ser providenciado o suporte da aplicação.

Destaca-se a superioridade do *software Texplus* no seu certame com 4Tex, não pelo produto que apresenta na atualidade, mas sim pelo potencial prometido pela empresa com a migração para o ambiente web no novo projeto em desenvolvimento, onde pode contar com o fator benéfico para o crescimento da empresa, a decadência do *software* concorrente.

MacWin, a nível de qualidade de *software*, supera o *Texplus*, pelo facto de se tratar de uma ferramenta de desenvolvimento tradicional, não apresentando dependências de ferramentas descontinuadas, nem de recurso a plataformas *low-code* (Anexo C: Interface Utilizador MacWin

). Esta característica obsequia proveitos, como uma maior capacidade de escalabilidade e de enriquecimento da interface do utilizador. Todavia, não são apenas as aptidões da plataforma que se pode deter em conta, tendo a MacWin duas grandes inconveniências em relação a *Texplus*:

- Trata-se de uma aplicação de produção vertical, ou seja, não disponibiliza as mesmas potencialidades no setor de tinturaria e acabamento;

- O preço de aquisição de *software* e custos associados à sua instalação e configuração são mais elevados.

No entanto, com base na informação obtida acerca da *Datamonplus*, com base em *feedbacks* de empresas que utilizam o *Texplus* há vários anos e através do acesso à apresentação do projeto "*Ruta 4.0*", e respetiva estrutura de equipa, considera-se uma forte probabilidade de a empresa fracassar com os objetivos prometidos, baseando-se nas deficiências organizacionais apresentadas e pela dependência total de uma ferramenta de *low-code*.

2.3 Desenvolvimento de Solução

"O nosso sucesso futuro é diretamente proporcional à nossa capacidade de compreender, adotar e integrar novas tecnologias no nosso trabalho" (Ratnakar, 2011). A escolha das

tecnologias de desenvolvimento é essencial na construção de um produto capaz de satisfazer os requisitos propostos por um cliente, de modo a tornar viável o futuro de uma empresa.

A secção inicia-se com a definição dos diferentes tipos de estrutura arquitetónicas de um sistema, onde são abordados os conceitos Monolítico, Microsserviços e Serverless, mencionando as suas características, vantagens e desvantagens associadas e o seu enquadramento com o projeto em contexto.

Posteriormente, são analisadas as opções tecnológicas possíveis para o desenvolvimento da aplicação, em que numa primeira instância são abordadas as plataformas *low-code* e *no-code.* Após, segue-se a exposição das diferentes ferramentas de programação focadas em *front-end* e *back-end* e, ademais, procede-se o estudo das diferentes estruturas de base de dados, especificando as particularidades de cada modelo de dados.

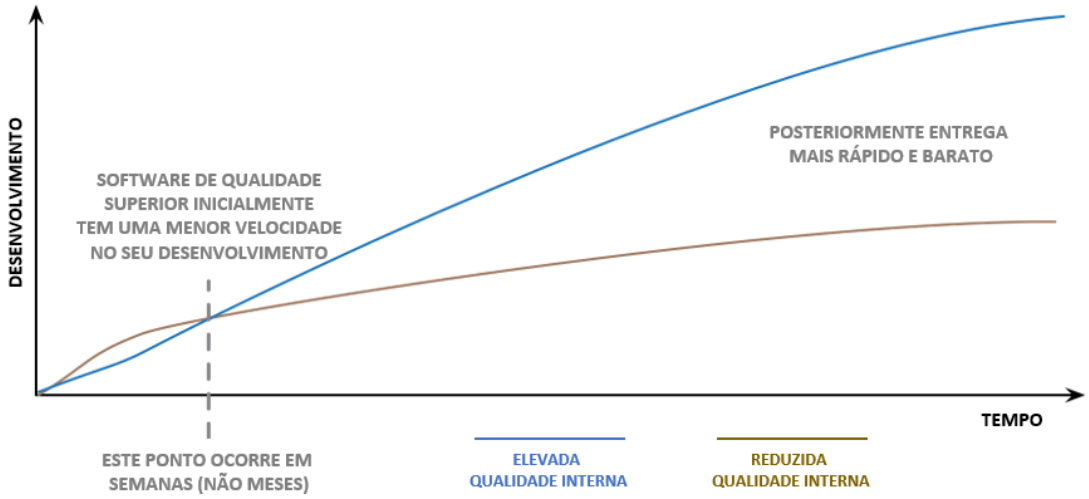
Finalmente, apresenta-se uma análise crítica sobre as tecnologias apresentadas, expondo as propriedades individuais, com o intuito de avaliar o seu nível de culminância perante um produto desta magnitude.

2.3.1 Arquitetura de Software

A arquitetura de *software* estabelece o comportamento de um sistema a nível estrutural, onde cada um dos seus componentes foi projetado para realizar uma tarefa ou conjunto de tarefas específicas.

"Sem olhar para a arquitetura de *software* regularmente, uma empresa está a abrir as portas a consequências e problemas a longo prazo que podem colocar os seus sistemas em risco de colapso, *hacking*, ou baixo performance." (Cast, 2020). A escolha da arquitetura deve ser abordada numa primeira instância da criação de um novo produto, que pressupõe num conjunto de decisões que estão diretamente correlacionadas com o desenvolvimento do *software.* Cada uma das medidas podem ter um impacto considerável na qualidade, capacidade de manutenção, desempenho e sucesso geral do produto final.

Deste modo, Fowler refere "O papel fundamental da qualidade interna é reduzir o custo de mudanças futuras." (Fowler, 2019), frisando o facto do papel da qualidade interna do produto espelhar os custos em mudanças futuras e, por isso, traduz esse pensamento de forma gráfica.



Fonte: (Fowler, 2019)

Figura 11 – Representação gráfica da lógica de Fowler na qualidade interna do *software*

Assim, no sentido de procurar o desenvolvimento de um produto com elevada qualidade interna, a seguinte análise define três tipos de arquitetura completamente distintos, no sentido de reconhecer quais as particularidades características de cada uma, as vantagens e desvantagens associadas. Finaliza-se com uma síntese comparativa no contexto do projeto e a probabilidade de resultar em problemas sérios, caso não sejam controlados. (Eeles, 2006)

2.3.1.1 Arquitetura Monolítica

Na engenharia de *software*, um padrão monolítico, refere-se a uma única unidade indivisível, ou seja, todas as partes do *software* são unificadas, e todas as suas funções são geridas num único lugar (Gnatyk, 2018).

Deste modo, esta arquitetura visa a simplicidade, onde o desenvolvimento e a resolução de problemas é facilitada pelo facto de todo o código estar concentrado num único lugar, não obrigando a implementar alterações ou atualizações separadamente. O processo de *debug* torna-se igualmente beneficiado, pela clareza no rastreamento de um determinado processo no código (Anastasia, 2019).

Adicionalmente, a performance geralmente destaca-se em comparação com as restantes arquiteturas, justificada pela ausência de dependências de serviços, ou de chamadas de diferentes *Application Programming Interface* (API).

No entanto, este tipo de estrutura declara-se como altamente limitativa no que toca a escalabilidade do produto, consequência da elevada concentração do código, que provoca um alto acoplamento. “Na arquitetura monolítica a maior vantagem também pode ser a maior desvantagem.” (Vega, 2019).

A inclusão de uma tecnologia é um processo extremamente problemático porque todo o aplicativo deve ser reescrito, tendo em conta que qualquer mudança pode afetar transversalmente o sistema.

2.3.1.2 Arquitetura Microsserviços

Os sistemas construídos com uma arquitetura orientada a serviços (SOA) consistem numa aplicação descentralizada e dividida por uma série de módulos menores, denominadas de serviços, que são fracamente acoplados e comunicam entre si por meio de uma rede. A arquitetura de microsserviço é uma variante de SOA, constituído por pequenos serviços independentes, com propriedades específicas, base de dados dedicada e que utilizam protocolos leves (Monus, 2018).

Além dos próprios serviços, existem outros componentes que surgem numa arquitetura típica de microsserviços, nomeadamente a *API Gateway*, que consiste no ponto de entrada onde, em vez de invocar os serviços diretamente, é chamada a API Gateway, que encaminha os serviços apropriados no *back-end*.

A API Gateway trata os pedidos de duas formas: redireciona-os para o microsserviço apropriado ou distribui os mesmos pelos microsserviços necessários para satisfazer o pedido. Desta forma, a *API Gateway* consegue disponibilizar uma API distinta para cada tipo de cliente, consoante os recursos necessários. Adicionalmente, consegue tratar questões como a autenticação e restrições de utilização, removendo a necessidade de as aplicar em componentes de negócio (Higginbotham, 2016).

Os benefícios obtidos com esta arquitetura têm como base a simplicidade e facilidade de compreensão das funcionalidades associadas a cada serviço, tendo em conta que são implementações mais curtas e de um grau de abrangência ínfimo. Adicionalmente, possibilita a existência de uma base de dados descentralizada, sendo assim possível escolher entre ter sistemas de gestão de base de dados diferentes para cada componente ou um conjunto dos mesmos (Nemer, 2019).

Deste modo, oferece igualmente agilidade na gestão de correção de *bugs*, nomeadamente pelo isolamento da falha, que não impactará a totalidade do funcionamento do sistema. O mesmo se aplica ao lançamento de novos recursos e inserção de novas tecnologias, ou seja, a tecnologia ou combinações de tecnologias utilizadas podem variar entre microsserviços, adaptando as escolhas ao contexto da funcionalidade do serviço (Microsoft, 2019).

"A complexidade de uma aplicação baseada em microsserviços é diretamente correlacionada com o número de serviços envolvidos." (Wittmer, 2019). Embora grande parte do processo de desenvolvimento seja simplificado com microsserviços, a complexidade da aplicação aumenta proporcionalmente com o número de serviços envolvidos.

Nesse sentido, existem algumas áreas onde os mesmos podem causar maiores transtornos, como por exemplo, a incompatibilidade de versões, isto é, apesar da possibilidade de fazer alterações em um microsserviço sem afetar os sistemas externos que interagem com ele, uma alteração na API, pode afetar vários utilizadores do sistema.

2.3.1.3 Serverless

Uma arquitetura *Serverless* dispõe de múltiplas e variadas definições para um sistema que, traduzido à letra, aparenta ser "Sem Servidor", mas, na verdade, funciona apenas como uma abstração da programação. Assim, permite ao desenvolvedor focar-se na produção de código, descartando preocupações associadas à escala e capacidade de armazenamento do sistema, em que os custos são diretamente proporcionais ao uso da aplicação.

Este conceito está associado ao conceito *FaaS*, mais concretamente *Function as a Service*, ou seja, trata-se de um modelo de execução de serviços em *cloud*, em que a responsabilidade da manutenção dos serviços pertence a terceiros, e as funções desenvolvidas são armazenadas pelo fornecedor, que aloca dinamicamente com base nas necessidades do cliente.

Assim sendo, quando ocorre a chamada de uma função, o fornecedor de serviços disponibiliza os recursos, executa a função e, logo após, desativa esses mesmos recursos. Contrariamente, nos outros modelos o sistema executa num servidor dedicado em permanente funcionamento, independentemente do uso (Carey, 2019).

A combinação de fatores mencionados, resultam numa maior produtividade do desenvolvedor, aumenta a capacidade de resposta na resolução de problemas e adicionalmente, disponibiliza um largo leque de linguagens de programação no desenvolvimento de funções. Por fim, descarta quaisquer responsabilidades relacionadas com a escalabilidade do sistema, traduzindo assim os diversos benefícios obtidos através de uma arquitetura *Serverless*.

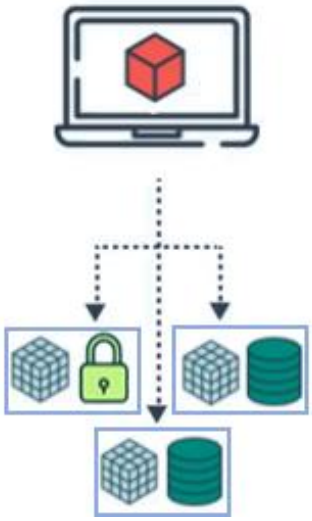
Por outro lado, abdicar da gestão do servidor e alocação de recursos, possui desvantagens, nomeadamente, a subordinação de terceiros que como referem Rohit Akiwatkar e Gary Arora "Você está preso a um fornecedor" (Akiwatkar & Arora, 2017), em que destaca as consequências de optar por uma arquitetura Serverless, onde embora existam opções de *open-source* disponíveis. O mercado Serverless é dominado pelas *big techs* de serviços *cloud* (*Amazon, Microsoft* e *Google*). Portanto, os *developers* geralmente optam pelas ferramentas disponibilizadas pelo próprio fornecedor do serviço, o que dificulta o processo de mudança para outra plataforma / fornecedor no caso de alguma insatisfação.

No seguimento das desvantagens da arquitetura Serverless, são os denominados "*cold start*", que ocorrem quando se sucede uma chamada a uma nova função desenvolvida, ou quando a mesma não é solicitada após um determinado período, passa automaticamente para o estado de inativa, que pode ser contornado através de um mecanismo que efetue chamadas periódicas aos diversos serviços (Fee, 2020).

2.3.1.4 Síntese Comparativa das Arquiteturas de Software

Na sequência dos diferentes tipos de arquitetura apresentados, a Tabela 4 visa os aspetos diferenciais e relevantes de cada uma das diferentes estruturas:

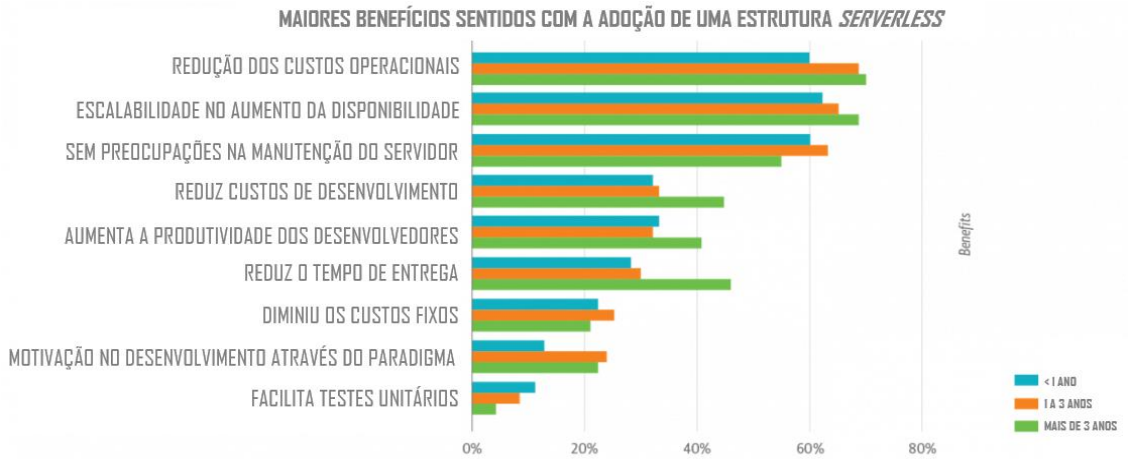
Tabela 4 – Síntese comparativa entre arquiteturas Monolítica, Microsserviços e *Serverless*

MONOLÍTICO	**MICROSSERVIÇOS**	***SERVERLESS***
DESCRIÇÃO		
Concentra os vários componentes e lógica num único bloco ideal para sistemas simples	Distribui a informação em diferentes partes / serviços orientados a uma finalidade	Semelhante aos microsserviços, mas descarta a responsabilidade da gestão de recursos
		
VANTAGENS		
Facilita o processo de *debug* caso se trate de um sistema curto, e desenvolvimento rápido e de custos reduzidos	Escalabilidade elevada, isolamento de falhas e simplicidade na interpretação do código	Escalabilidade elevada, isolamento de falhas e inibe responsabilidade de manutenção de infraestruturas
DESVANTAGENS		
Escalabilidade limitada, erros impactam de forma transversal e dificuldade na manutenção proporcional à complexidade	Complexidade no planeamento e configuração de serviços, segurança e manutenção do servidor	Dependência do fornecedor de serviços e custos difíceis de estimar

A Tabela 4 evidencia o facto de a arquitetura monolítica ser cada vez menos adotada no desenvolvimento de novos produtos, tendo em conta as suas limitações associadas ao *design*, escalabilidade e agilidade na adição de novas funcionalidades e, por isso, no contexto do projeto não deve ser considerada uma possibilidade (Shet, 2016).

Desta forma, centrando na arquitetura de microsserviços e *serverless*, as diferenças são diminutas, mas, normalmente, um microsserviço é mais abrangente e pode fazer mais do que uma função. Uma função caracteriza-se por um pedaço de código relativamente pequeno que executa apenas uma ação em resposta a um evento, mas logicamente trata-se de uma generalização e pode variar consoante o *developer* (Cloudflare, 2021).

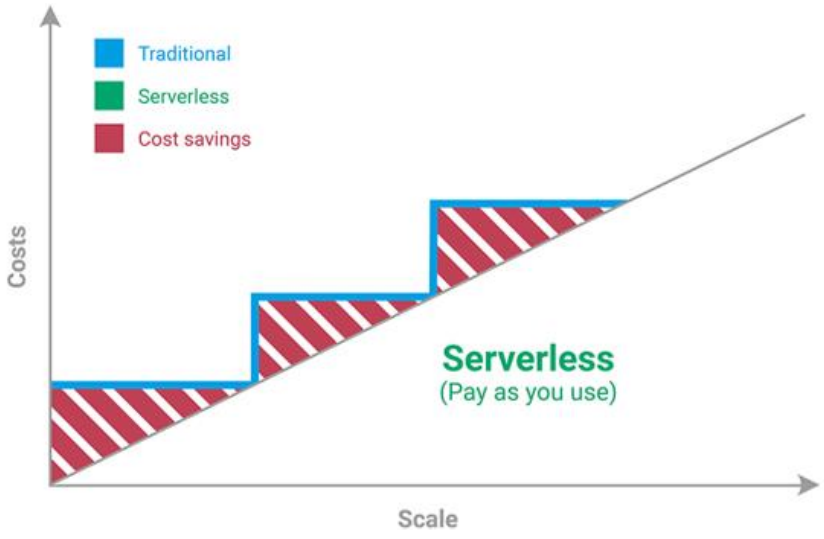
No contexto do projeto, o Serverless parte em vantagem pela flexibilidade na escalabilidade, tendo em conta a agilidade e capacidade de resposta. Sendo estas características fulcrais para corresponder às necessidades do cliente, a *O'Reilly* elaborou um estudo onde foram abrangidas organizações de diferentes setores, com o objetivo de verificar a adesão das empresas à tecnologia Serverless e os benefícios obtidos através da mudança.



Fonte: (Magoulas & Guzikowski, 2019)

Figura 12 – Benefícios obtidos com arquitetura *Serverless*

Complementarmente, os custos associados à escalabilidade diferem entre os diferentes tipos de estrutura, que de um determinado ponto de vista pode-se considerar uma desvantagem, pela imprevisibilidade dos custos com o servidor, nomeadamente, pelo facto de se pagar unicamente consoante a utilização. Por outro lado, evita um investimento contínuo na infraestrutura e, por vezes, o receio de não disponibilizar os meios necessários, obriga a um investimento em recursos superior ao realmente necessário, tornando este processo altamente dispendioso:



Fonte: (Edelhoff, 2019)

Figura 13 – Gráfico comparativo de despesas associadas à escalabilidade

Assim, a minimização de custos e a automatização da escalabilidade são os fatores que destacam o *Serverless* de outras soluções baseadas em nuvem, já que a sua política de pagamento consoante o uso, previne despesas desnecessárias de desenvolvimento e implementação de outras aplicações e, por isso, considera-se a solução arquitetónica ideal para o projeto em contexto

2.4 Tecnologias de Desenvolvimento

A transformação e a evolução ininterrupta na era digital atual ocorrem praticamente em todos os setores, onde as tecnologias de desenvolvimento estão incessantemente a adaptar-se e, por isso, como referiu Alison DeNisco, "O desenvolvimento *software* trata-se de um campo dinâmico, onde as linguagens de programação, ferramentas e tecnologias podem viver e morrer dentro de alguns anos, e as necessidades do mercado de trabalho estão em constante mudança." (Rayome, 2019) . É fundamental explorar e ser minucioso nas análises às tendências atuais de tecnologias de desenvolvimento, porque o rigor e cautela nas decisões tomadas num ponto inicial e embrionário do projeto, podem resultar numa partida em vantagem perante a concorrência.

Desta forma, tratando-se do desenvolvimento de um novo produto, antes de avançar com qualquer desenvolvimento de código, definir-se-á a stack tecnológica, que diz respeito ao conjunto de tecnologias que constituem um sistema (Suschevich, 2020).

Assim, na vasta variedade de tecnologias disponíveis para desenvolvimento capazes de construírem uma aplicação, a presente secção dispõe as mesmas separadas em três tipos de gamas, ferramentas *no-code* que prometem a inexistência de código na criação de uma aplicação*,* plataformas *low-code* que se complementam com pequenas porções de código e o método tradicional *developer-code*.

2.4.1 No-Code

As plataformas de desenvolvimento de *software no-code* possibilitam a construção de aplicações sem recurso a qualquer conhecimento de código e que abriu portas a um público que não dispõe de habilidades ou *know-how* para produzir *software*. (Dreverman, 2019)

O recurso a ambientes de desenvolvimento *no-code*, abstrai muitas das complexidades na construção da interface de utilizador no *front-end*, pelo facto de fornecer modelos previamente construídos. Posteriormente, só necessitam de ser minimamente ajustados ao contexto da aplicação em construção e que através do tempo economizado no desenvolvimento, possibilitou a entrada destas plataformas no mercado, por representarem uma fonte de poupança a nível de custos.

Deste modo, personalidades do mundo da programação, nomeadamente Chris Wanstrath, ex-CEO e cofundador do *Github*, referiu que "Eventualmente haverá zero código. Não para tudo,

mas para alguns projetos não vai ser preciso codificar. Não temos de apenas automatizar armazéns e consultórios médicos, mas também o desenvolvimento de *software*." (Swinhoe, 2017). Atualmente, a presença do *no-code* é vista por muitos como uma força disruptiva na indústria de desenvolvimento de *software*.

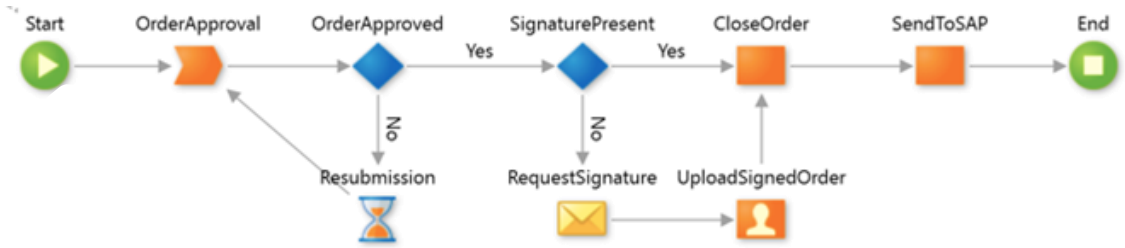
Por outro lado, o recurso ao *no-code* funciona como uma alternativa excelente para empresas direcionadas ao desenvolvimento de aplicações simples, com prazos de entrega curtos, e sem fundo monetário suficiente para sustentar um informático ou uma outra empresa prestadora de serviços. (Healey, 2020).

À semelhança de qualquer ferramenta tecnológica, o *no-code* possui igualmente as suas limitações. "Ao nível de limitação de uma aplicação define-se pelo quão escalável é, e se pode ou não ser integrada com soluções externas. Definindo *no-code* como uma solução normalmente não escalável o suficiente para ter impacto nos objetivos de um negócio." (Meer, 2018). O autor defende a imaturidade das plataformas e os problemas de escalabilidade, destacando ainda o facto de as plataformas estarem direcionadas a soluções genéricas e não serem ideais para desenvolvimentos de grau de complexidade e especificidade altos.

2.4.2 Low-Code

No desenvolvimento de *software* convencional, os desenvolvedores escrevem linhas de código para criar uma determinada a funcionalidade e os respetivos requisitos desejados, contrariamente às plataformas *low-code*. Estas encapsulam tudo o que funciona nos bastidores e fornecem aos utilizadores da plataforma ferramentas visuais de manipulação intuitiva no sentido de otimizar o processo de criação de novas aplicações (Alexander, 2020).

As plataformas desta gama, geralmente apresentam componentes reutilizáveis e ferramentas de *drag and drop*, ou seja, simplesmente arrastar e soltar de maneira a construir as diferentes etapas e recursos específicos desejados pelo utilizador, oferecendo igualmente recursos de teste para o fluxo desenvolvido (Looper, 2019).



Fonte: (Josephson, 2019)

Figura 14 – Exemplo demonstrativo de *workflow* de desenvolvimento *low-code*

Por outro lado, a verdade é que nem todas as chamadas plataformas de *low-code* funcionam da mesma forma, ou seja, as soluções variam amplamente, especialmente em termos da quantidade de código necessária para criar ou modificar aplicações dependendo, portanto, da plataforma, o termo *low-code* pode ser extremamente relativo (Ismail, 2017).

No que toca a vantagens, este tipo de plataformas, não diferem muito das anteriormente mencionadas no *no-code*, nomeadamente, aceleram o processo de desenvolvimento e encurta prazos de entrega e permite a acessibilidade e agrega indivíduos não formados tecnicamente na área da informática (Spendel, 2020).

No entanto, destaca-se a incapacidade e limitação na integração com ferramentas externas, assim como a impossibilidade de integrar ferramentas que controlam as versões, a qualidade e a resiliência da implementação, problema esse que não acontece no desenvolvimento tradicional (Pratt, 2020).

2.4.3 Comparação Plataformas *No-Code* e *Low-Code*

Ambientes de desenvolvimento low-code e no-code são tipicamente confundidos, sobretudo pelo facto de ambos os tipos de plataformas proporcionarem funcionalidades já previamente implementadas, que podem ser manuseadas por pessoas sem conhecimento de implementação de código.

Tabela 5 – Plataformas *No-code* vs *Low-code*

	No-Code	***Low-Code***
Principais Utilizadores	Startups	Desenvolvedores
Objetivo Primário	Facilidade no Uso	Velocidade de Construção
Requer Codificação	✗	✓
Facilidade de Uso	★★★	★☆☆
Personalização	Ajuste de Templates	Total
Desenvolvimento Ponta-a-Ponta	Limitado	Disponibilizado

Analisando a Tabela 5, esta contrasta ambas as categorias. verifica-se que a principal característica que as difere é o facto de *low-code* necessitar da intervenção de um programador a um certo ponto (Alexander, 2020).

Assim sendo, o *no-code* oferece benefícios como a rapidez e responsividade dos projetos. No entanto, dificilmente são integrados com serviços exteriores, ou seja, o seu uso é maioritariamente como *front-end*, ao contrário do *low-code*, que apresenta competências quanto ao ramo de integrações (Khan, 2020).

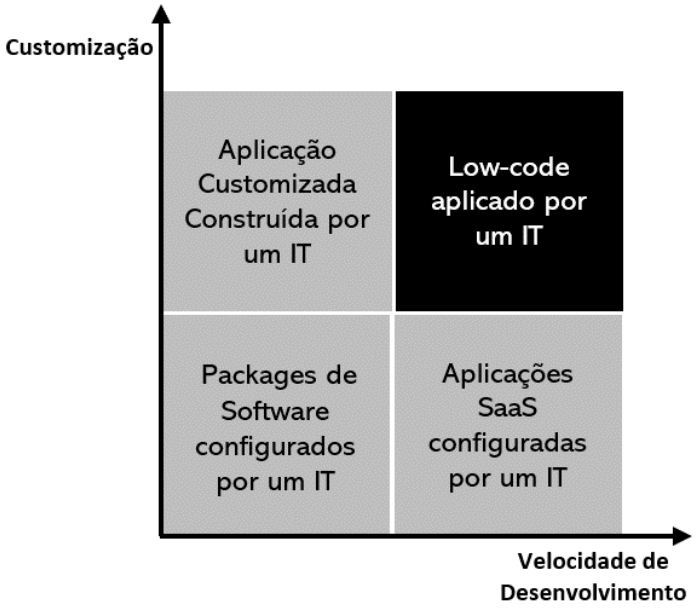


Figura 15 – Correlação entre grau de personalização e velocidade de desenvolvimento

A Figura 15 evidencia a relação entre o nível de personalização e a velocidade de desenvolvimento. Deste modo, apesar de o *no-code* apresentar-se como a melhor solução caso o propósito seja elevar o nível na capacidade de resposta, o *low-code* permite, através da presença de código, contornar algumas limitações.

No entanto, o contexto do projeto exige o desenvolvimento de um produto altamente personalizável e escalável, condições essas que não são proporcionadas por esta gama de plataformas. Devido à sua natureza genérica e dificuldade em lidar com elevadas cargas de utilização, representam uma barreira na expansão do sistema e, por isso, torna-se inviável a utilização de plataformas *no-code* ou *low-code* nesta conceção.

2.4.4 Developer-Code

A metodologia em questão designa-se como a mais tradicional, trata-se da criação de uma aplicação através da escrita de código por um conjunto de pessoas especializadas, programadores. *Developer-code* é usado em projetos bem mais complexos do que os de *low-code* e *no-code*, no entanto, o seu processo de desenvolvimento classifica-se como lento, devido a vários fatores como a obrigação da realização de um número de testes superior e haver uma maior probabilidade de surgirem *bugs* (Maruti TechLabs Pvt Ltd, 2020).

Esta mecanismo deve ser usado, não só quando as empresas desejam construir uma aplicação que envolva uma quantidade elevada de funcionalidades distintas e específicas, mas também quando há a necessidade da integração com sistemas externos (Kumar, 2021).

Dentro das tecnologias de desenvolvimento, por escrita de código, distinguem-se em várias categorias: *full stack* dedicado a desenvolvimento ponta a ponta*, front-end* foca-se na interface disponibilizada ao utilizador e *back-end* onde está contida a lógica da aplicação.

2.4.4.1 Tecnologias de Full Stack

Uma ferramenta de código *full stack* tem a potencialidade de criar uma aplicação de ponta a ponta, sem depender de qualquer outro *software*, isto significa que uma única plataforma tem a capacidade de criar o front-end e back-end de um projeto (upGrad, 2019).

Atualmente, existe uma quantidade exorbitante de tecnologias de programação, das quais algumas são consideradas *full stack*, contudo poderão ter uma performance extraordinária no front-end e uma performance que deixa a desejar em back-end ou o inverso.

"O desenvolvimento web não está a morrer, está a fragmentar-se." (Gall, 2018). O uso de plataformas full stack começou a entrar em desuso, principalmente pelo facto da conexão de uma tecnologia especializada em *front-end,* com uma especializada em *back-end* acabar por multiplicar o desempenho de uma aplicação, posto isto, *full stack* fica posto de parte quanto às tecnologias a considerar para o projeto em contexto.

2.4.4.2 Tecnologias de Front-End

"Uma interface de utilizador corresponde a uma piada. Se for preciso explicá-la, qualifica-se como insatisfatória" (Leblanc, 2018). O propósito de uma ferramenta de desenvolvimento de *front-end* é tornar uma aplicação intuitiva e esbelta.

Um dos principais objetivos do desenvolvimento de *front-end* passa por criar uma experiência fluída ao utilizador. Em outras palavras, o *front-end* representa a face do produto e, portanto, deve ser intuitivo. Embora pareça um objetivo simples, trata-se de um processo altamente complexo, pois nem todos os utilizadores e dispositivos são iguais e podem ter comportamentos e reações diferentes perante o mesmo aspeto visual. (Airfocus, 2020)

Assim, de uma forma resumida, o front-end é a combinação do design gráfico com a interatividade do utilizador e, por isso, uma interface ao ser *user-friendly*, resulta num grau de satisfação do cliente inequivocamente superior, levando ao aumento do prestígio do produto.

2.4.4.3 ReactJS

ReactJS caracteriza-se como uma ferramenta extremamente poderosa no que diz respeito à criação de interfaces de utilizador, devido à sua rapidez, responsividade e aos seus *templates* atrativos. Atualmente, esta tecnologia é a mais popular, sendo usada por muitos dos monopólios *web* existentes no mercado como *Facebook, Instagram, Reddit* e *Netflix* (Coder Academy, 2016).

"*JavaScript* é a fita-cola da internet." (Campbell, 2017). Na contemporaneidade, todas as páginas *web* usam as linguagens *HTML* e *CSS*, com o propósito de estruturar e embelezar a interface do utilizador. *ReactJS* é uma biblioteca *JavaScript*, ou seja, beneficia dos potenciais da linguagem de programação, a qual disponibiliza elementos e eventos que despoletam na página, promovendo assim a interatividade com o utilizador (Hack Reactor, 2018).

Deste modo, trata-se de um dos *softwares* mais recentes no mercado, sendo criado em 2011 por Jordan Walke, um engenheiro de *software* do *Facebook*. A sua estrutura de código estende as características da biblioteca de componentes de *HTML* da *XHP*, uma plataforma proveniente de *PHP*. Só em 2013, *React* tornou-se *open-source* e, posteriormente, em 2015, surgiu *React Native* que permitiu o desenvolvimento em *Android* e *iOS*, tendo no mesmo ano se tornado igualmente numa tecnologia de código aberto (Education Ecosystem, 2019).

Complementando, considera-se *ReactJS* uma excelente opção quanto à criação de páginas *web* tanto para *browsers* como dispositivos móveis. Comparado a outras bibliotecas de desenvolvimento, o início do seu desenvolvimento é demorado, todavia a criação de novas funcionalidades acaba por ser mais célere. A tecnologia em questão apresenta uma enorme vantagem, a capacidade de se desenvolver num browser virtual (DOM), tornando a sua evolução mais eficiente e intuitiva. Finalizando, *ReactJS* ainda disponibiliza facilidades na criação de interfaces de utilizador como a sua estrutura por componentes e o uso de *JSX* (DA-14, 2017).

2.4.4.4 AngularJS

AngularJS trata-se de uma tecnologia de código aberto, tendo como ponto forte a criação de aplicações *web*, tornando o código estático de *HTML* em código dinâmico, através do uso de *TypeScript* (Samadder, 2020).

TypeScript define-se como um parente de JavaScript, onde o código do primeiro necessita de compilação, enquanto o segundo não. Além disso, a arquitetura de ambos é distinta, em que no caso do *TypeScript* coexiste tipos de variáveis e funcionalidades que permitem a criação de modelos básicos.

O início da génese de *AngularJS* deu-se em 2008 por Misko Hevery e Adam Aborns, ambos programadores da *Google*, com o objetivo comum de simplificar o desenvolvimento web. O projeto foi bem-conceituado, fazendo com que, subsequentemente, a tecnologia avançasse para novos patamares, acabando por se tornar *open-source*, adaptando uma arquitetura à base de componentes (Gudelli, 2019).

Em suma, o *software* em estudo é uma ferramenta que favorece de *data-binding* e *dependency injection* (AngularJS, 2020), isto é, o código permanece limpo e com baixo acoplamento, o que evita a reescrita de código e amplia a facilidade na criação de novas funcionalidades.

2.4.4.5 Django

Django representa uma das variadas *frameworks* da linguagem de programação *Python*, tratando-se de uma plataforma de licença *BSD*, sendo uma licença de código aberto utilizada nos sistemas operacionais do tipo *Berkeley Software Distribution*. O seu ponto forte é segurança, o que o torna especialmente poderoso em termos de transações. Esta tecnologia utiliza-se com frequência em redes sociais, em que uma aplicação tem de ser capaz de aguentar um número elevado de utilizadores e transações, “Quando é preciso uma aplicação que cresça em profundidade e complexidade para qualquer escala e que seja capaz de aguentar muitos utilizadores e/ou transações, o Django brilha.” (Protasiewicz, 2018).

Python distingue-se como uma das linguagens mais genéricas, podendo ser usada tanto para desenvolvimento *web*, como para inteligência artificial. No entanto, existem outras ferramentas preferíveis para um tipo de produto específico (Mindfire Solutions, 2017). Assim sendo, a sua vantagem passa pelo seu vasto leque de paradigmas de programação, o qual é usado pelas *big techs*, como a *Google, Instagram, Facebook* e *Netflix* (Reynolds, 2020).

O *software* em contexto foi criado em 2003, por Adian Holovaty e Simon Willison. A sua finalidade era conceber uma plataforma de *Python* com o propósito de elaborar páginas *web*. Só em 2005 foi lançada como ferramenta *open-source*, por outras palavras, o seu uso é gratuito, mas são impostas restrições mínimas (Ecosystem, 2019).

Como anteriormente mencionado, *Django* possui funcionalidades altamente abrangentes, porém, projetos de inteligência artificial e *machine learning* são a gama de produtos para os quais melhor se enquadra, principalmente nas áreas da ciência e análise de dados.

2.4.4.6 Análise Comparativa das Tecnologias Front-end

No seguimento das tecnologias de *front-end* apresentadas, a Tabela 6 visa os aspetos diferenciais e relevantes de cada uma das *frameworks*.

Tabela 6 – Análise comparativa das tecnologias de *front-end*

	ReactJS	***AngularJS***	***Django***
Conceito	DOM virtual com renderização do lado do servidor	DOM real com renderização do lado do cliente	DOM real com renderização do lado do cliente
Ano de Criação	2011	2008	2003
Vinculações	★★★	★★☆	★★☆
Flexibilidade Sintática	★★★	★★☆	★★★
Gestão de Dependências	★★☆	★★★	★★☆
Performance	★★★	★★☆	★☆☆
Debug/Testes	★★★	★★☆	★★☆
Escalabilidade	★★★	★★☆	★☆☆

Fonte: (Simform, 2020)

A Tabela 6 evidencia o facto de o *Django* ser a tecnologia mais antiga de entre as três debatidas, tendo perdido grande parte do mercado para os *softwares* de *front-end* de *JavaScript*.

Enquanto *Django* renderiza o *HTML* no servidor web, *ReactJS* e *AngularJS* renderizam no *web browser*, o que resulta numa enorme vantagem para as duas últimas referências, em termos de eficiência. Com isto, pode-se concluir que *Django* não será a ferramenta mais apropriada para um sistema de gestão, mas sim para projetos que envolvam ciência de dados.

Relativamente a *ReactJS*, ao contrário de *AngularJS*, oferece *JSX, DOM* e *React Native*, por outras palavras, o uso desta *framework* no desenvolvimento de produtos oferece diversas vantagens, nomeadamente, a rapidez, interatividade e maior rendimento na criação de

interfaces, além de facilitar a construção de aplicações *mobile*. Após esta avaliação, considera-se inquestionável a escolha de *ReactJS* como tecnologia de *front-end* a ser utilizada no projeto.

2.4.4.7 Tecnologias de Back-End

"Uma dona de casa é quem trabalha no *back-end* para que a família possa atuar no *front-end*." (Joshi, 2020). A componente da aplicação que manipula a interface do utilizador é o *front-end*, enquanto o *back-end* tem a responsabilidade de controlar e manusear os dados, que estabelecem um vínculo simbiótico onde ambas as ferramentas se correlacionam. Adicionalmente, o utilizador final não tem acesso a esta camada, sendo só mesmo utilizada para operações invisíveis.

Como mencionado precedentemente, *FaaS* será a arquitetura a abordar, na qual o utilizador não se preocupa com a manutenção das infraestruturas do lado do servidor. Desta forma é imprescindível o uso de uma plataforma *cloud* que preste um serviço serverless.

2.4.4.8 AWS Lambda

A primeira grande empresa fornecedora de serviços *cloud* a disponibilizar uma arquitetura *serverless* foi a *Amazon Web Services*, a partir de *AWS Lambda*. Esta tecnologia permite a execução de código sem haver a preocupação de gerir os servidores que o detêm (Gancarz, 2019).

A arquitetura de *AWS Lambda* designa-se por *FaaS*, ou seja, é um serviço que proporciona código sem conter qualquer tipo de estrutura complexa. O facto de apenas suportar certos tipos de linguagens ou ferramentas de programação, provocando uma limitação evidente à própria plataforma, contudo, a mesma é compatível com as mais poderosas do mercado, como por exemplo, *Java, Python, C#* e *NodeJS.* (Amazon Web Services, 2020).

AWS Lambda devota-se à manutenção contínua dos servidores, oferecendo obrigatoriamente alta disponibilidade, isto faz com que os seus clientes não se tenham de preocupar com o acréscimo de novas funcionalidades ao sistema. Além disso, ainda segue o lema "O que você usa, é o que você paga", por outras palavras, os usufruidores não pagam pelo tempo de conexão do servidor, mas sim pelo número de solicitações (Butler, 2016).

Em suma, esta plataforma de computação *serverless* não só concede uma performance e escalabilidade soberbas em relação a qualquer outra arquitetura de *back-end*, como também faculta mecanismos de segurança, podendo ser ainda mais aprimorada mediante a extensão de outros serviços da *AWS*.

2.4.4.9 Azure Functions

Microsoft decidiu apostar nos serviços *cloud*, de modo a promover soluções simples e escaláveis aos seus clientes. Com este objetivo em consideração criaram as *Azure Functions*, sistemas de computação orientados a eventos, os quais são ativos como *triggers* por eventos do *Azure* ou serviços externos (Mashkowski, 2016).

Da mesma forma, todas as plataformas *serverless* tem a vantagem de desacoplar e garantir a segurança dos dados e código, todavia tem a desvantagem de apenas suportar certas

linguagens de programação, entre estas destacam-se *NodeJS, C#, Java, F#* e *Python* (Serverless360, 2020).

Uma das táticas de melhoria de performance e redução de custo das *Azure Functions* é o uso de *endpoints HTTP*, acessível pelos mais variados tipos de aplicações, os seus utilizadores apenas têm a responsabilidade de se conectarem às fontes de dados ou transmitirem mensagens (Mashkowski, 2016).

Findando, este *software* tem o cargo de conglomerar os eventos e código num só processo, de modo a maximizar a eficiência de um projeto. Assim, é possível reduzir os custos, código e complexidade de uma aplicação, além de deixar as preocupações da manutenção do lado do servidor para uma empresa fiável e conceituada.

2.4.4.10 Google Cloud Functions

Google Cloud Functions tipifica-se como um dos serviços disponibilizados pela *Google*, a qual contém um sistema de nuvem que disponibiliza o mais variado tipo de infraestruturas. A plataforma em análise contém uma arquitetura orientada a eventos. Bem como as outras estruturas *serverless*, esta libera o fardo da manutenção do servidor e da complexidade de configurações adicionais (Google, 2021).

A ferramenta *serverless* apresenta as desvantagens de um sistema "sem servidor" como outro qualquer, a imperiosidade do uso de uma linguagem ou ferramenta compatível (*NodeJS, Python, Java, .NET*) e a vulnerabilidade de deixar terceiros encarregados do domínio do servidor (Google, 2021).

Esta é mais recente dos três *softwares* mais estimados do mercado, sendo as outras duas *AWS Lambda e Azure Functions*. Consequentemente, apresenta menos possibilidades em termos de funcionalidades, linguagens e performance (TechMagic, 2020).

Depreendendo, *Google Cloud Functions* é uma plataforma que apresenta uma qualidade a estimar, tendo em conta os anos que se encontra no mercado e as funcionalidades que já aborda. Com isto, pode-se concluir que este *software* ainda se encontra incompleto, mas com competências para permanecer no topo dos *rankings*.

2.4.4.11 Síntese Comparativa de Tecnologias Serverless

Existem várias características por entre as quais se podem avaliar os *softwares* "sem servidor", sendo estas identificadas na Tabela 7 com a sua devida avaliação.

Tabela 7 – Comparação *AWS Lambda* vs *Azure Functions* vs *Google Cloud Functions*

	AWS Lambda	**Azure Functions**	**Google Functions**
Gestão de Dependências	★★★	★★★	★★★
Escalabilidade	★★★	★★☆	★☆☆
Performance	★★★	★★☆	★☆☆
Persistência	★★★	★★☆	★★☆

Debug / Testes	★★☆	★★★	★☆☆
Custos	★★★	★☆☆	★★☆

Fonte: (Simform, 2020)

Google Cloud Functions minucia-se como a infraestrutura serverless mais recente de todas em estudo, apresentando um grau de imaturidade no produto, oferecendo um menor grau de requisitos não funcionais, todavia também a expõe a um maior grau de evolução. Assim sendo, descarta-se a *Google Cloud Functions*, pois não só se prevê o acompanhamento com as outras duas ferramentas restantes.

Deste modo, *AWS Lambda* e *Azure Functions* dominam por completo o pináculo das tecnologias "sem servidor", a sua escalabilidade e performance são monumentais. *Azure Functions* subjuga qualquer outro *software* da mesma arquitetura em ambiente *Windows*, já *AWS Lambda* controla todos os outros ambientes, não obstante o facto de se encontrar quase par-a-par com *Azure,* mesmo no sistema operativo *Windows*.

Adicionalmente, *AWS Lambda* oferece melhores condições a nível de custos, em que um cliente apenas tem de pagar pelo que usa e não pelo tempo de uso, desta forma este *software* concede as melhores condições comparativamente aos restantes apresentados.

2.4.5 Interpretadores de Código

Cada tecnologia "sem servidor" usufrui de um interpretador de código, caso contrário não haveria um conjunto de instruções a ser executado. Por conseguinte, torna-se inevitável a escolha de uma ferramenta ou linguagem de programação compatível com a infraestrutura *serverless*.

2.4.5.1 NodeJS

O *NodeJS* foi fundado em 2009, por Ryan Dahl, com a finalidade de colmatar as limitações do servidor web mais usufruído da altura, Apache HTTP Server. A falha mais predominante do servidor Apache dava-se quando ocorriam muitas conexões simultâneas. Como solução, o *software* em análise foi criado com o intuito de evitar o código bloqueado por entre as conexões, através de *sequential programming* [1] (Magaji, 2020).

A principal função de *NodeJS* passa por interpretar código *JavaScript* e tem a particularidade de poder correr aplicações tanto do lado do servidor como do cliente, sendo mais adequado a sua utilização como tecnologia de *back-end*. É uma ferramenta de código aberto e está disponível para os sistemas operativos *Windows, Linux, Sun OS* e *macOS* (Joseph, 2016).

[1] Sequência de instruções realizadas sempre pela mesma ordem e que apresenta sempre os mesmos resultados

NodeJS apresenta várias conveniências como plataforma do lado do servidor, como a sua rapidez e em particular a sua escalabilidade. Isto significa que consegue adicionar novos nós a sistemas já existentes num curto espaço de tempo, tal como o próprio nome indica. (Program-Ace, 2020).

Muitas das grandes empresas usam *NodeJS*, como por exemplo *PayPal, Uber, LinkedIn* e *Netflix* (Thor, 2018). A razão pela qual estas organizações investem no *software*, associa-se ao seu poder de adaptação, que através do uso de módulos externos consegue refinar as suas funcionalidades.

2.4.5.2 Java

Java foi inicialmente concebido por James Gosling, em 1995, para programar dispositivos móveis, televisões, entre outros eletrodomésticos, todavia acabou por ser uma ferramenta demasiado vigorosa, tendo acabado por se tornar uma linguagem de programação da internet (Javatpoint, 2018).

Uma das linguagens de programação mais gerais e aproveitadas na época atual é *Java*. A sua arquitetura orientada ao objeto reduz o tempo de programação e salienta-se pela capacidade de poder desenvolver eficazmente tecnologias *Android*, pois este *software* já consiste em várias bibliotecas *Java* (Belani, 2020).

Existem várias razões pelas quais *Java* é atualmente a linguagem de programação *web* mais utilizada. Destaca-se o seu potencial de escalonamento, tanto vertical como horizontal, dependendo da plataforma, o seu suporte de *multi-threading* (manuseamento de vários utilizadores ou *threads* em simultâneo) e o seu vasto leque de *API's* e bibliotecas (Malhotra, 2019).

Resumindo, *Java* caracteriza-se por ser uma ferramenta poderosa em termos de segurança, performance e *multi-threading*, o que a classifica como uma boa linguagem de *back-end* (IntexSoft, 2019). Contudo, apresenta desvantagens, nomeadamente, o facto de ser uma linguagem muito pesada e com morosidade na saída de novas atualizações.

2.4.5.3 C#

C# foi criado por Anders Hejlsberg, baseado em C++ e *Java*, todavia apresenta uma gama de funcionalidades superior. Isto não significa que C# seja a melhor das linguagens, mas sim que depende do sistema operativo em uso e da finalidade do projeto a desenvolver (Javatpoint, 2018).

A linguagem de programação de servidor mais conceituada em sistemas *Windows* é C#, isto porque não são necessárias configurações extra para correr nesse ambiente. Este *software* é uma das linguagens de .NET, ou seja, beneficia das suas deleitáveis funcionalidades de programação de páginas *web* e serviços *cloud* (Agilites, 2017).

A tecnologia em foco descreve-se como sendo *open-source* e orientada ao objeto, para além de conseguir produzir aplicações de ponta a ponta. C# fornece reutilização de código e recursos

que decrescem o tempo de codificação, sem a necessidade de bibliotecas ou ferramentas suplementares (Sam, 2018).

Depreendendo, C# é uma ferramenta geral com vantagens em sistemas *Windows*. As suas funcionalidades proporcionam confiabilidade e escalabilidade, porém, existem linguagens que concedem melhores condições noutros sistemas operativos.

2.4.5.4 Síntese Comparativa dos Interpretadores de Código

Na Tabela 8 encontram-se as dissimilitudes entre as linguagens de programação analisadas, tendo em foco os seus requisitos não funcionais e arquitetura.

Tabela 8 – Síntese dos interpretadores de código

	NodeJS	**Java**	**C#**
Arquitetura	Multiparadigma	Orientado ao objeto	Orientado ao objeto
Modelo de Processamento	Assíncrono, single-thread sem bloqueio, modelo I/O	Síncrono, multi-thread, I/O bloqueante	Síncrono, multi-thread, I/O bloqueante
Flexibilidade Sintática	Sim	Não	Não
Escalabilidade	★★★	★☆☆	★★☆
Performance	★★★	★★☆	★☆☆
Debug	★☆☆	★★☆	★★★
Sistemas Operativos	★★★	★★★	★★☆

Fonte: (Goyal, 2017)

A Tabela 8 evidencia que *Java* e C# são muito semelhantes, contudo a segunda inclui funcionalidades poderosas para desenvolvimento web que a primeira não incorpora, como o suporte de programação de funções ou criação de tipos de variáveis definidos pelo desenvolvedor.

Por outro lado, *NodeJS* salienta-se em termos de simplicidade, flexibilidade e rapidez, principalmente graças ao seu modelo de processamento *single-thread* sem bloqueio. Contrariamente, Java e C#, apresentam um sistema *multi-thread* que contém limites causando problemas de memória.

Adicionalmente, uma outra vantagem de *NodeJS* particulariza-se pelo uso de *JSON*, o que simplifica imensamente o uso de objetos de dados, enquanto que no caso de *Java* e C# é necessário pré-definir e utilizar objetos complexos.

Concluindo, no seguimento da análise e características abordadas, *NodeJS,* comparativamente aos restantes apresentados*,* qualifica-se como sendo a linguagem mais vantajosa no que toca a benefícios / custos para o projeto em causa.

2.4.6 Base de dados

"Os dados são preciosos e durarão mais que os próprios sistemas." (Berners-Lee, 2006). Os dados são algo indispensável a uma empresa, é a partir dos mesmos que são tomadas decisões, através da extração de informação, de forma a maximizar o valor do produto.

O objetivo de uma base de dados passa por guardar a informação de forma estruturada em coleções dentro de um sistema, tornando assim possível o manuseamento dos dados com *queries*, de maneira segura.

Distinguem-se dois tipos de base de dados: relacional e não relacional. A primeira consiste numa arquitetura orientada a tabelas o que a torna eficaz quanto a *queries* complexas, apesar da sua estrutura rígida, o que obriga ao uso de esquemas pré-definidos forçando os dados a seguirem a sua composição. Por outro lado, base de dados não relacional, apresenta uma estrutura dinâmica, facilitando a organização de dados não estruturados, seguindo uma arquitetura orientada a documentos, pastas e arquivos (SQL & NoSQL Databases, 2019).

2.4.6.1 Linguagens de Base de Dados

As bases de dados usufruem de um conjunto de *scripts* que controlam e manipulam as transações de dados, essas instruções são designadas de linguagens de base de dados. No presente subcapítulo são abordadas as duas principais categorias e no epílogo é feita uma analogia entre ambas.

2.4.6.2 SQL

SQL provém de *Structured Query Language*, esta linguagem de base de dados considera-se a mais usada atualmente, isto deve-se ao facto de conseguir guardar e manipular grandes quantidades de dados de forma estruturada (Romanowski, 2020).

De acordo com a informação mencionada anteriormente, apresenta uma arquitetura orientada a tabelas, ou seja, a informação encontra-se organizada em linhas e colunas, oferecendo um desempenho elevado a *queries* de linhas múltiplas. Na maioria dos casos a escalabilidade caracteriza-se como vertical, isto é, para obter uma melhor performance é necessário o acréscimo de recursos como CPU, memória e armazenamento. *SQL* ainda segue as propriedades *ACID*: atomicidade, consistência, isolamento e durabilidade (Sharma A. , 2018).

A manipulação dos dados realiza-se através de operações *CRUD* (*create, read, update, delete*), mais especificamente pelo uso dos comandos *insert* para criação de novos registos, *select* para a leitura de registos, *update* para a atualização de registos e *delete* para a eliminação de registos. Da mesma forma, existem diversos recursos mais complexos como variáveis, cursores e operadores, com o propósito de facilitar e acelerar a leitura ou manuseamento de dados relacionados entre várias tabelas (Viescas, 2018).

2.4.6.3 NoSQL

NoSQL não só tem a capacidade de organizar conjuntos de dados, independentemente da disposição dos mesmos, como também consegue estruturá-los quando existem em grandes quantidades, mais concretamente *big data*. A escalabilidade é horizontal, ou seja, o desempenho aumenta a partir da fragmentação dos dados, ou pela adição de novos servidores.

NoSQL rege-se pelo teorema *CAP* (*Consistency, Availability, Fault-Tolerant Partition*), o qual afirma que é impossível um armazenamento de dados garantir mais do que dois dos três atributos de qualidade (Sharma A. , 2018).

Tabela 9 – Arquiteturas *No-SQL*

ARQUITETURA	**DESCRIÇÃO**
Orientada ao Documento	Os dados são guardados num documento, com estrutura semelhante a *JSON*
Orientada a *Key-Value*	Cada registo contém uma chave e um valor, em que para se obter um determinado valor é preciso indicar a chave
Armazéns *Wide-Column*	Tal como as bases de dados relacionais, guarda os registos em tabelas, no entanto, cada linha de uma mesma tabela não necessita de ter o mesmo número de colunas
Orientada a Grafos	Os nós e arestas guardam os dados, sendo os nós responsáveis pela informação de um determinado objeto e as arestas pelas relações entre os objetos

Fonte: (Pandit, 2020)

A arquitetura de uma base de dados *NoSQL* pode variar entre diferentes tipos. A Tabela 9 apresenta e descreve cada um deles.

*"*Bases de dados *NoSQL* emergiram de necessidades não reconhecidas" (Sullivan, 2015). A manipulação dos dados é feita sob a forma de documentos sem uma estrutura rígida a ser cumprida, o que torna este tipo de base de dados excecional para grandes volumes de dados ou aplicações de tempo-real, evitando o uso de joins.

2.4.6.4 Comparação de Linguagens de Estruturas de Dados Relacionais e Não Relacionais

Quando se trata de escolher uma base de dados, embora ambas as estruturas apresentadas sejam opções viáveis, existem diferenças importantes que se devem ter em conta.

Tabela 10 – Síntese comparativa SQL e NoSQL

	Relacional	**Não Relacional**
Modelo	Relacional	Não Relacional
	Armazena a informação em Tabelas	Armazena informação em documentos JSON
Informação	Ótimo para soluções em que todos os registos possuem as mesmas propriedades	Oferece flexibilidade, nem todos os registos precisam de armazenar as mesmas propriedades
	Adicionar uma nova propriedade pode exigir alterações de esquemas	Novas propriedades podem ser adicionadas em tempo real

	Relacional	Não Relacional
	Os relacionamentos ocorrem através da junção entre tabelas	Os relacionamentos ocorrem pela seleção de dados não normalizadas e apresenta-os como um único registo
	Bom para dados estruturados	Bom para dados semiestruturados
Esquema	Restrito	Dinâmico e flexível
Transações	Suporta transações ACID	O suporte a transações ACID varia de acordo com a solução
Consistência	Forte consistência suportada	Variável consoante a solução
Escala	Escala bem na vertical	Escala bem na horizontal

Fonte: (Darmon, 2016)

Não existe uma linguagem de base de dados perfeita, o que obriga ao estudo das vantagens e desvantagens entre SQL e NoSQL. Enquanto que a primeira se diferencia por ser benéfica para aplicações em que o modelo de dados apenas se altere raramente, a segunda já se considera a melhor opção quando os dados são desestruturados e volumosos. Posto isto, há que calcular o valor de cada uma das alternativas a partir dos requisitos do projeto.

2.4.6.5 Tecnologias de Base de Dados

As linguagens de base de dados por si só não são capazes de ser integradas em projetos extensos. Para isso é imprescindível recorrer a ferramentas que ajudem na sua incorporação com outros sistemas, de modo a possibilitar o processamento de milhões de transações e ao mesmo tempo assegurar caches de memória e alta velocidade.

Este subcapítulo descreve detalhadamente a análise de várias hipóteses de plataformas de base de dados, tanto relacionais como não relacionais, além de ser feita a sua comparação, com o intuito de selecionar a mais adequada para o sistema de gestão têxtil.

2.4.6.6 MySQL

Este *software* foi fundado em 1995 por uma empresa sueca, dirigida por David Axmark, Michael Widenius e Allan Larsson, com base num outro produto de manuseamento de base de dados, denominado UNIREG. Em 2000 tornou-se *open-source* sob a *GNU General Public License (GPL).* Posteriormente, *MySQL* iniciou uma parceria com a *SAP*, de onde surgiram recursos poderosos como *triggers* e procedimentos. Já em 2008 a *MySQL* juntamente com a *Sun Microsystems*, que por sua vez foi apoderada pela *Oracle*, sendo esta a atual detentora do *software* de base de dados (Exadel Media, 2017).

MySQL é um sistema de gestão *open-source* que, tal como o nome indica, opera com a linguagem *SQL*. Na atualidade, trata-se da ferramenta de base de dados que abrange um elevado número de soluções no mercado, não só por oferecer código aberto, mas também pelo facto de estar disponível para os sistemas operativos mais conceituados, incluindo *Windows, macOS* e *Linux*. As linguagens de programação que constituem *MySQL* são *C* e *C++* (Romanowski, 2020).

A tecnologia em questão representa-se como sendo segura, escalável e de rápido processamento (EDUCBA, 2021). Assim sendo, a sua performance é magnânima em comparação a outras ferramentas semelhantes, e poderá ser facilmente integrada em arquiteturas cliente-servidor ou ambientes incorporados.

2.4.6.7 PostgreSQL

PostgreSQL foi baseada num *package* denominado *POSTGRES*, o qual foi concebido pela Universidade de California em Berkeley, mediante uma equipa liderada por Michael Stonebreaker. O seu desenvolvimento iniciou-se em 1986 escrito em linguagem C e, posteriormente, em 1994 Andrew Yu e Jolly Chen acrescentaram um interpretador de SQL ao *POSTGRES* tornando-o um *software open-source*, alterando o nome da aplicação para *Postgres95*. No ano 1995, com a instalação de novas aptidões *SQL*, o nome foi novamente modificado, para o atual (PostgreSQL, 2014).

A tecnologia de base de dados em foco refere-se a mais uma ferramenta de administração *open-source*, esta também inclui mecanismos de *SQL* implementados. É considerado o mais avançado e poderoso sistema de gestão *SQL* de código aberto, graças à sua capacidade de extensibilidade, conformidade, escalabilidade e integridade de dados (Hristozov, 2019).

O controlo de simultaneidade de várias versões (MVCC) denota-se por ser o método mais prestigiado de *PostgreSQL*, pois permite múltiplas leituras e manipulações de dados ao mesmo tempo (Bhatia, 2020). Esta ferramenta apresenta grandes vantagens para projetos que envolvam especificamente o armazenamento de dados relativos a uma empresa ou a gestão de dados, com o propósito de obter informações úteis que suportem a tomada de decisões.

2.4.6.8 MS SQL Server

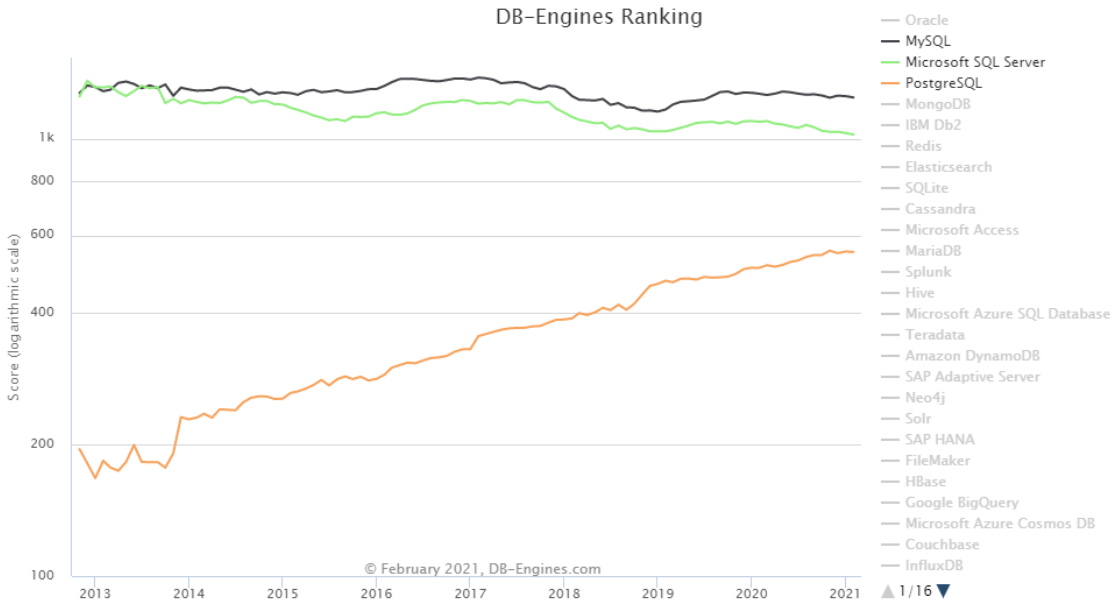
Assim como o nome indica, *MS SQL Server* foi criado pela *Microsoft* e *Sybase* em 1993. Em 1996 foi introduzido o suporte a *data warehousing*. Já no ano 1998, para além da *Microsoft* ter ficado com a patente total do sistema, foram acrescentadas inúmeras novas funcionalidades. Apenas por volta de 2016 foi acrescentado o suporte aos sistemas operativos *Linux* e *macOS* (Toprak, 2020).

MS SQL Server, nomeadamente *Microsoft SQL Server*, é um sistema de base de dados relacional com licença comercial, desenvolvido em C e C++. Este *software*, como referido anteriormente, encontra-se disponível para os principais sistemas operativos, contudo, para ser possível correr em *macOS* incumbe-se o uso de um *Docker* (G., 2019).

A tecnologia em evidência apresenta limitações para os sistemas operativos não nativos do seu desenvolvimento, porém, tem feito evoluções com o objetivo de contrabalançar esta fraqueza. Igualmente se distingue uma outra grande fragilidade, nomeadamente, a obrigatoriedade do uso de licenças pagas para ser possível obter a eficiência total do *software*. Não obstante as barreiras apresentadas, *MS SQL Server* posiciona-se no mercado como uma estrutura de dados relacional que oferece maior escalabilidade e fiabilidade.

2.4.6.9 Síntese Comparativa de Bases de Dados Estruturadas

A Figura 16 apresenta as três tecnologias de gestão de base de dados relacionais analisadas (*MySQL, PostgreSQL* e *MS SQL Server*), com base no seu grau de popularidade ao longo do tempo.



Fonte: (DB-Engines, 2021)

Figura 16 - Popularidade das bases de dados relacionais

Entre os três sistemas de base de dados apresentados no gráfico, destaca-se o *MySQL*, pela sua permanência em primeiro lugar durante vários anos consecutivos, embora quantidade não signifique qualidade.

A Tabela 11 indica os atributos de cada uma das bases de dados relacionais em estudo, diferenciando cada uma delas com base nas propriedades de integração com outros sistemas.

Tabela 11 - Síntese comparativa das bases de dados estruturadas analisadas

	MySQL	***PostgreSQL***	***MS SQL Server***
Arquitetura	Base de dados relacional	Base de dados relacional	Base de dados relacional

	MySQL	***PostgreSQL***	***MS SQL Server***
Empresa	Oracle	PostgreSQL Global Development Group	Microsoft
Licença	*Open-source*	*Open-source*	Licença Comercial
Baseada em nuvem	Não	Não	Não
Sistemas operativos	FreeBSD, Linux, OS X, Solaris, Windows	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix, Windows	Linux, Windows
Esquema de dados	Sim	Sim	Sim
Índices Secundários	Sim	Sim	Sim
Suporta SQL	Sim	Sim, com extensões	Sim, com extensões
API's / Métodos de Acesso	*ADO.NET, JDBC, ODBC, API* nativa própria	*ADO.NET, JDBC*, biblioteca nativa de C, ODBC, API de transmissão para objetos extensos	*ADO.NET, JDBC, ODBC, OLE DB, Tabular Data Stream*
Suporta linguagens apresentadas (2.4.5)	Sim	Sim	Sim
Triggers	Sim	Sim	Sim
Chaves estrangeiras	Sim	Sim	Sim
Conceitos de transação	*ACID*	*ACID*	*ACID*
Capacidade de memória	Sim	Não	Sim
Conceitos de utilizador	Autenticação de utilizadores, não existem grupos de utilizador nem funções	Direitos de acesso segundo o padrão *SQL*	Direitos de acesso segundo o padrão *SQL*

Fonte: (DB-Engines, 2021)

Como mencionado anteriormente, os três tipos de plataforma de base de dados são relacionais, isto significa que a sua estrutura representa-se como orientada a tabelas. *MySQL* e *PostgreSQL* são *softwares* de código aberto, enquanto que *MS SQL Server* contém uma licença comercial, o que implica custos adicionais para um melhor suporte e eficiência máxima da aplicação.

Todas as plataformas são independentes de nuvem, isto é, integram-se em estruturas, mesmo sem a utilização de serviços *cloud*. Os sistemas operativos mais utilizados na atualidade são suportados pelas três ferramentas, contudo para *MS SQL Server* torna-se inevitável o uso de um Docker na sua incorporação com *macOS*.

Da mesma forma, todas as tecnologias são compatíveis com as seguintes *API's*: *ADO.NET, JDBC, ODBC*, ainda que *MySQL* disponibiliza sua própria e, por outro lado, *PostgreSQL* é conciliável com bibliotecas de *C* nativas e *MS SQL Server* com *OLE DB* e *Tabular Data Stream*[2] (TDS).

A congruência de uma base de dados em relação às linguagens de programação deve ser tida em conta, entre as quais destaca-se *MySQL*, pelo facto de conectar-se com os interpretadores de código mais empregues, nomeadamente, *JavaScript, Java, C#, Python, C++,* oferece liberdade ao desenvolvedor (DB-Engines, 2021).

Assim, de acordo com esta análise, *MS SQL Server* acaba por ser a menos indicada para a realização do projeto, estando *MySQL* e *PostgreSQL* equiparadas em termos de capacidades. Portanto, concentrando a análise nas restantes, evidenciam-se maiores diferenças ao nível da arquitetura, em que a *PostgreSQL* salienta-se pela sua eficácia para *data warehousing* e *data analysis,* enquanto que *MySQL* realça-se num contexto direcionado a páginas *web*. Sumindo, dado que o projeto têxtil abrange essencialmente web, a ferramenta mais apropriada das bases de dados relacionais é MySQL.

2.4.6.10 MongoDB

MongoDB foi baseada na base de dados da empresa *DoubleClick*, tendo colmatado as suas maiores falhas de *performance* a partir da linguagem *C++.* Foi criada em 2007 por Dwight Merriman, Eliot Horowitz e Kevin Ryan, equipa que dirigia as bases de dados da *DoubleClick*, estando atualmente disponível para os sistemas operativos mais utilizados (MongoDB, Inc., 2021).

A tecnologia em análise trata-se de uma gestão de base de dados *open-source,* orientada a documentos, ou seja, implementa as funcionalidades *NoSQL*. Os dados são armazenados de forma flexível em formato *JSON* ou *BJSON* (*JSON* binário), isto significa que cada documento pode ter um número diferente de campos e a sua estrutura é facilmente alterada (Shannon Bradshaw, 2019).

Ao contrário de muitas bases de dados não relacionais, *MongoDB* tem capacidades vantajosas, como comandos poderosos que possibilitam *joins* de *queries*, dois tipos de relações entre entidades em vez de um (referências e incorporações) e suporta transações *ACID* de vários documentos com isolamento instantâneo (MongoDB, Inc., 2021).

A tecnologia em apreço identifica-se por ser uma base de dados distribuída (DDB), em outros termos, é uma coleção integrada de bases de dados que está distribuída fisicamente através de uma rede de computadores (Shannon Bradshaw, 2019). Esta habilidade oferece várias vantagens como um grau de disponibilidade alto, bom desempenho das transações e flexibilidade.

[2] Protocolo utilizado para a transferência de pedidos entre dois sistemas

2.4.6.11 DynamoDB

DynamoDB tipifica-se como um sistema de gestão de base de dados não relacional, criado pela *Amazon Web Services (AWS*). Ao contrário dos seus concorrentes, a empresa é que trata do manuseamento das operações das aplicações que recorrem aos seus serviços, ou seja, os clientes evitam custos de administração de base de dados deixando a *AWS* gerir, por exemplo, a partir do uso de *DynamoDB* (Crosett, 2019).

O *software* em pauta foi concebido com o intuito de ultrapassar a maior fraqueza dos serviços fornecidos pela *Amazon*, a falha dos sistemas devido ao alto tráfego. O desenvolvimento desta plataforma foi iniciado a pedido de Werner Vogels e deu origem a várias bases de dados *open-source*, como Cassandra (Deshpande, 2014).

Esta ferramenta está disponível para todos os sistemas operativos (*Linux, Windows, iOS, Android, Solaris, AIX* e *HP-UX*) graças a uma *API* oferecida pela *AWS*. A plataforma, orientada a *key-value*, mais especificamente a documentos, oferece uma rapidez e flexibilidade superior ao consuetudinário, no que toca às operações de leitura e edição de dados. (Amazon Web Services, Inc., 2021). A sua escalabilidade é magnânima, ainda assim existem limitações, como a exigência do uso de *AWS* ou a quantidade máxima de dados que podem ser manipulados.

2.4.6.12 Cassandra

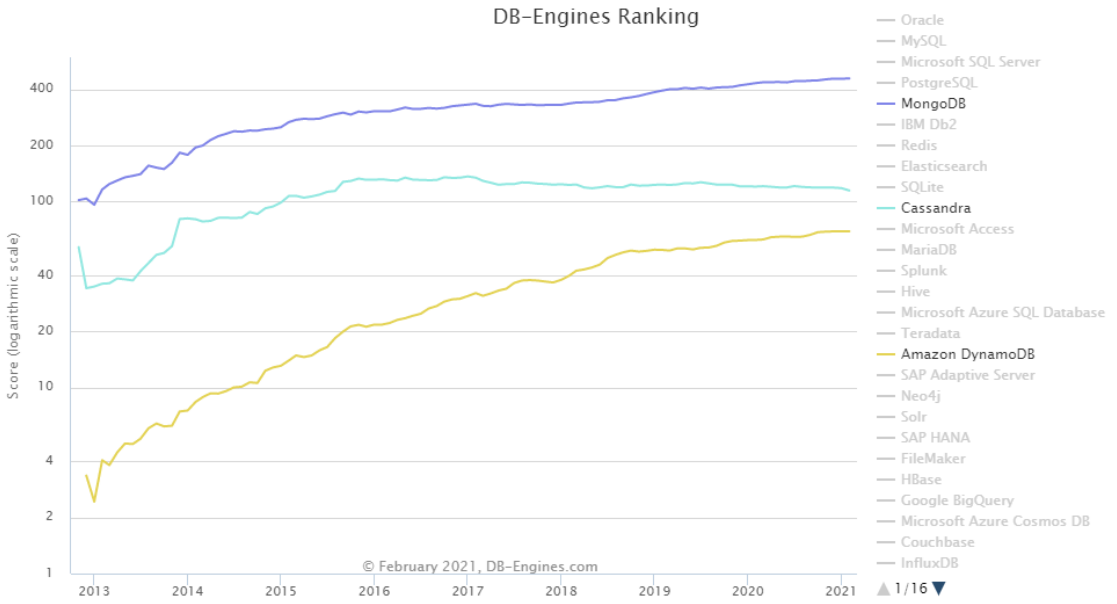
Cassandra provém de *DynamoDB*, no entanto, esta integração apenas foi realizada anos após a criação da plataforma. Avinash Lakshman e Prashant Malik foram os desenvolvedores iniciais da aplicação, no âmbito da empresa Facebook. O seu código fonte foi disponibilizado para a comunidade em 2008 e ,posteriormente, em 2019, tornou-se um projeto da *Apache*, o qual em 2010 chegou ao seu auge (Zaforas, 2020).

A estrutura de dados em causa é uma plataforma de base de dados *open-source*, a que acarreta as propriedades de uma base de dados não relacional. O *software* apresenta uma arquitetura orientada a *wide column store*, portanto, no sentido de se obter o máximo da sua eficiência, os dados devem ser partidos em nós e as entidades devem ser coesas e pouco acopladas (Schulz, 2018).

O *software* em foco foi desenvolvido em *Java* e está disponível para *Windows*, *Linux* e *macOS*. A sua estrutura corresponde a um modelo semelhante ao de *SQL*, em que os dados estão separados por tabelas identificadas por uma chave. Esta arquitetura designa-se por *Cassanda Query Language (CQL)* (Simplilearn, 2021). Deste modo, salienta-se a sua excelência na gestão de *big data*, tratando-se de ser uma base de dados distribuída, favorecendo a manipulação de dados. Em contrapartida, as leituras de dados não são tão eficazes, ou seja, projetos que incluam armazém de dados não são os mais apropriados para a utilização desta plataforma.

2.4.6.13 Síntese comparativa das bases de dados analisadas

Na Figura 17 e Tabela 11 encontram-se as comparações entre os sistemas de gestão de base de dados *NoSQL* descritos anteriormente, especificando o grau de popularidade de cada um dos *softwares* ao longo dos anos.



Fonte: (DB-Engines, 2021)

Figura 17 – Popularidade das bases de dados não relacionais

Das bases de dados não relacionais em análise, distingue-se a *MongoDB*, pela sua subida ininterrupta em termos de usabilidade. A Tabela 12 destaca as diferenças e semelhanças entre as bases de dados não relacionais em análise, especificamente *MongoDB, DynamoDB* e *Cassandra*, destacando os atributos mais significativos para a sua integração num *software*.

Tabela 12 – Síntese comparativa das tecnologias de bases de dados não relacionais analisadas

	MongoDB	***DynamoDB***	***Cassandra***
Arquitetura	Base de dados não relacional, orientada ao documento	Base de dados não relacional, *wide column store*	Base de dados não relacional, orientada ao documento
Empresa	MongoDB, Inc	Amazon	Apache *Software* Foundation
Licença	Open-source	Licença Comercial	Open-source
Baseada em nuvem	Não	Sim	Não
Sistemas operativos	Linux, OS X, Solaris, Windows	Hospedado	BSD, Linux, OS X, Windows
Esquema de dados	Não	Não	Não
Índices Secundários	Sim	Sim	Restritos
Suporta SQL	Suporta queries de leitura com a sua camada de tradução	Não	Estrutura semelhante a SQL (CQL)
API's ou métodos de acesso	Protocolos próprios usando JSON	RESTful HTTP API	Protocolos próprios, Thrift
Linguagens de programação que suportam	Actionscript, C, C#, C++, Clojure, ColdFusion, D, Dart,	.Net, ColdFusion, Erlang, Groovy, Java,	C#, C++, Clojure, Erlang, Go, Haskell, Java, JavaScript, Perl,

	MongoDB	***DynamoDB***	***Cassandra***
	Delphi, Erlang, Go, Groovy, Haskell, Java, JavaScript, Lisp, Lua, MatLab, Perl, PHP, PowerShell, Prolog, Python, R, Ruby, Rust, Scala, Smalltalk, Swift	JavaScript, Perl, PHP, Python, Ruby	PHP, Python, Ruby, Scala
Triggers	Sim	Com integração com AWS Lambda	Sim
Chaves estrangeiras	Não	Não	Não
Conceitos de transação	Transações ACID de vários documentos com isolamento instantâneo	ACID	Sem nenhum
Capacidade de memória	Sim	Não	Varia consoante a disponibilidade financeira
Conceitos de utilizador	Direitos de acesso para utilizadores e funções	Os direitos de acesso e funções são definidos através do AWS Identity e Access Management (IAM)	Direitos de acesso segundo o padrão SQL

Fonte: (DB-Engines, 2021).

A Tabela 12 esclarece que *MongoDB* e *DynamoDB* dispõem uma arquitetura orientada a documentos, o que significa que os dados são guardados numa única instância e cada objeto pode diferenciar entre si. Por outro lado, *Cassandra* possui uma arquitetura *wide column store* em que os dados são partidos automaticamente entre servidores.

Adicionalmente*, DynamoDB* além de ser o único dos sistemas de gestão de base de dados que tem uma licença comercial, também se distingue por ser o único disponível a partir de serviços de nuvem, ao contrário das restantes apresentadas que são *open-source*.

No que toca a compatibilidade com sistemas operativos, *MongoDB* como *Cassandra* suportam *Windows*, *macOS* e *Linux*. No caso da *Amazon Web Services* oferece um ambiente próprio a *DynamoDB* que possibilita o seu uso a todos os sistemas operativos, através de uma *API*.

No sentido de possibilitar a utilização destas tecnologias, aplicam-se métodos de acesso, sendo em *DynamoDB* através da utilização de uma conexão de *API,* enquanto que o *MongoDB* e *Cassandra* recorrem a protocolos específicos, *JSON* e *CQL* respetivamente. Um aspeto pouco diferencial entre as ferramentas de base de dados em estudo, é a compatibilidade com as linguagens de programação, apesar de *MongoDB* apresentar mais hipóteses (DB-Engines, 2021).

Assim, no contexto do projeto têxtil, será indispensável a execução de *queries* a partir de índices secundários, devido à vasta lógica de negócio e à quantidade de entidades relacionadas entre si.

Desta forma, a escalabilidade e documentação são igualmente fatores a notar, provocados pela necessidade de análise em tempo real, o que coloca *MongoDB* e *DynamoDB* um passo à frente de *Cassandra*.

Posto isto, analisando o *DynamoDB*, evidencia-se a facilidade numa fase embrionária do projeto, por disponibilizar recursos que descartam precauções de configuração, consequentemente, eleva o seu custo primordial, mesmo com a ausência de despesas de administração. Finalizando, considera-se o *MongoDB* a melhor opção, pelos gastos iniciais associados diminuídos, mas também pelo facto de suportar cargas de dados de *key-value* relativamente maiores.

Análise comparativa *MySQL* vs *MongoDB*

A Figura 18 identifica várias peculiaridades entre as operações CRUD de MySQL e MongoDB, por outras palavras, diferencia a sua sintaxe e estrutura.

Figura 18 – Sintaxe CRUD de MySQL vs MongoDB

MySQL	MongoDB
INSERT	

`INSERT INTO estudante (`
`'numEstudante', 'primeiroNome', 'ultimoNome')`
`VALUES ('1140338', 'Vasco', 'Lusitano');`
	`db.account.insert ({`
	`numEstudante: "1140338",`
	`primeiroNome: "Vasco",`
	`ultimoNome: "Lusitano"`
	`});`

UPDATE	

`UPDATE estudate`
`SET contacto = 919191919`
`WHERE numEstudante = '1140338'`
	`db.estudante.update (`
	`{ numEstudante: '1140338' },`
	`{ $set: { contacto: '919191919' }`
	`});`

DELETE	
`DELETE FROM estudante` `WHERE email = '1140338@isep.ipp.pt';`	`db.estudante.remove ({` `"email": "1140338@isep.ipp.pt" });`
DOCUMENTO	

`numEstudante primeiroNome ultimoNome`
`1140338 Vasco Lusitano`
	`{`
	`numEstudante: "1140338"`
	`primeiroNome: "Vasco"`
	`ultimoNome: "Lusitano"`
	`}`

Fonte: (Smallcombe, 2020)

	MySQL	MongoDB
Modelo de Dados Toda a lógica do domínio é implementada em entidades distintas	✔	✘
Esquemas de Informação Bem Estruturados Dados facilmente organizados e versados por linguagem de máquina	✔	✘
Relações Entre Entidades Ajuda do reconhecimento das dependências entre entidades	✔	✘

Transações Complexas Transações de dados ACID	✔	✘
Auto - Sharding Fragmentação da base de dados em partições horizontais	✘	✔

Fonte: (Smallcombe, 2020)

Após a análise da Tabela 12 e da pesquisa bibliográfica apresentada anteriormente, confirma-se que *MySQL* corresponde à opção mais adequada ao contexto do projeto, considerando os seguintes requisitos:

- O modelo de dados apenas será modificado em situações muito esporádicas;
- Não se lida propriamente com *big data*, ou um envio continuo de dados que necessitam de ser armazenados na base de dados;
- *Queries* complexas são executadas com frequência.

Por outro lado*, MongoDB* apresenta igualmente características capazes de realizar operações que cumpram com os critérios referidos anteriormente, no entanto, quando comparado com as capacidades de *MySQL*, apresenta as suas limitações. O objetivo primordial passa por manter a integridade dos dados e assegurar a segurança e rapidez da sua manipulação, sempre cumprindo com as propriedades *ACID*.

3 Análise de Valor

Atualmente, o mercado exige inovação contínua por parte das organizações, onde novos produtos e serviços são lançados a um ritmo crescente.

Segundo a Will Purcell, "No mundo dos negócios, existem muitos tipos diferentes de inovação que uma empresa pode buscar. Frequentemente, estão diretamente vinculados a produtos individuais, processos ou fluxos de trabalho internos ou modelos de negócios" (Purcell, 2019), refere a inovação como um processo que pode ocorrer em áreas distintas dentro da mesma organização.

A transformação de uma ideia num produto, trata-se de um processo longo e complexo, que requer a estruturação de um processo de inovação, com o objetivo de potencializar a eficiência dos novos desenvolvimentos.

O desenvolvimento do serviço deve ser trabalhado com foco no cliente, priorizando aspetos como a usabilidade e visibilidade, tendo em conta que a escolha recai sob uma interface mais *user-friendly*, ao invés de uma que apesar de mais íntegra, aparenta ser menos intuitiva.

3.1 *Front-end* no Desenvolvimento de Novos Produtos

Os projetos de inovação nas organizações geralmente ocorrem num determinado fluxo, que se pode dividir entre três principais atividades: *front-end*, desenvolvimento de produto e comercialização.

Figura 19 – Fluxo de atividades do desenvolvimento de novos produtos

O conceito *front-end*, no contexto de desenvolvimento de novos produtos, surge na primeira fase, que agrega o período desde a criação e ideia até à sua aprovação para desenvolvimento ou complementação (Murphy & Kumar, 1997), ou seja, são definidos os conceitos inerentes à inovação idealizada e à disponibilidade de recursos.

A segunda fase, desenvolvimento do produto (NPD), aborda o desenho e a implementação. Por outras palavras, num primeiro passo realiza-se a análise e eleição da arquitetura do produto e, posteriormente, procede-se ao seu concebimento respeitando toda a documentação adquirida nas etapas anteriores (Byrne, 2019).

O conceito de comercialização diz respeito à entrada do produto no mercado. Este processo envolve vários patamares desde o seu marketing até à sua venda. Apesar de ser a última fase, considera-se tão importante quanto as outras para alcançar o sucesso, pois é nesta que se prova a qualidade ao cliente (Spacey, 2017).

Atualmente, a fase inicial do processo de inovação tem um carácter experimental, que envolve altos níveis de incerteza e, por isso, surge uma de várias denominações para este período como o *Fuzzy Front End* (FFE) (Moenaert, Meyer, Souder, & Deschoolmeester, 1995), tendo em conta o seu encadeamento intrincado sem uma linguagem definida.

Seguidamente, em 2002, Peter Koen foi um dos primeiros a usar o termo *Front End of Innovation* (FEI), com o intuito de abdicar da palavra "*Fuzzy*" e, consequentemente, afastar o mistério envolvido neste processo, elaborando assim o seu raciocínio com uma linguagem holística (Koen, 2018).

3.2 New Concept Development Model

O objetivo final do *Front-End* passa por definir que tipo de funções desempenha o produto e características que o mesmo deve ter no futuro. Não se trata apenas da ideia de criar um produto ou serviço, mas também conhecer o valor agregado que representa.

De acordo com Cooper, "O objetivo do FE é a criação de um conceito de produto bem definido." (Cooper, 1990), ou seja, quando refere um conceito de produto bem definido, permite um entendimento mais exato em vários aspetos inerentes à sua criação, nomeadamente, o tempo de desenvolvimento, custos associados, conhecimentos técnicos necessários, potencial e posicionamento de mercado, risco e adaptação por parte da organização.

Desta forma, surgiram novos conceitos associados ao *Front End of Innovation*, um deles desenvolvido por Koen que, como anteriormente mencionado, pretendia arredar as incógnitas associadas a este processo, criando o New Concept Model (NCD), aliado ao processo de *Stage-gate* elaborado por Cooper (Reid & Brentani, 2004).

A estrutura *Stage-gate* utiliza um processo sequencial com etapas e tempo especificados, enquanto o *NCD Model* apresenta um modelo de relacionamento não sequencial. A Figura 20 fornece um resumo das principais atividades que ocorrem antes do *New Product Development* (NPD), ou seja, que antecedem a fase de desenvolvimento de um produto.

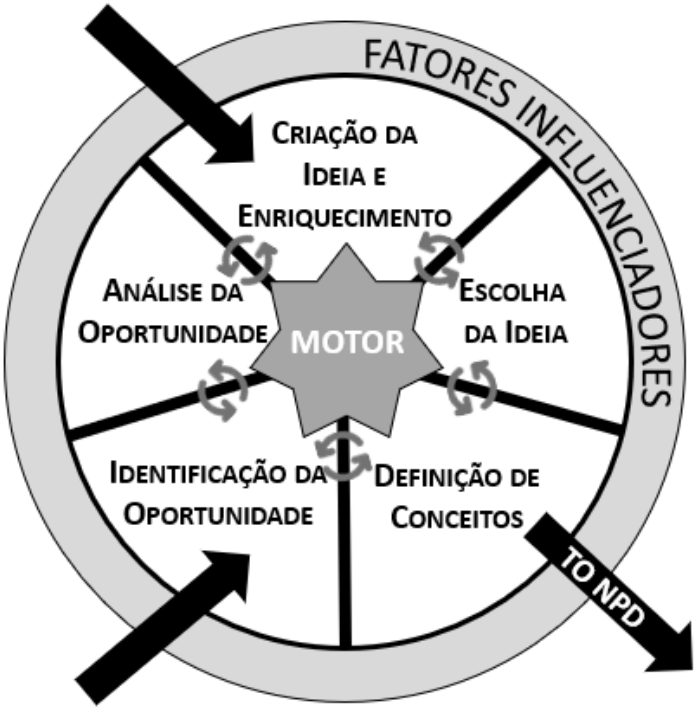


Figura 20 – *New Concept Development* (NCD) *Model* desenvolvido por Koen

Koen representa *NCD Model* através da forma circular, evidenciando as misturas e interações entre as partes que compõe a estrutura, que devem ocorrer de forma contínua. As secções que separam o modelo são os fatores influenciadores, os cinco elementos-chave e o motor, mais especificamente:

- Fatores externos incontroláveis e influenciadores do desenvolvimento, regidos no mundo externo, ou seja, terceiras partes envolvidas, como por exemplo, política governamental, legislação, canais de distribuição, clientes, concorrentes e clima político e económico que, de alguma forma influenciam a tomada de decisão na organização;
- Cinco elementos-chave do *front-end* que constituem a oportunidade e ideia: identificação da oportunidade, análise da oportunidade, criação da ideia e enriquecimento, escolha da ideia e definição de requisitos;

- Motor impulsiona os cinco elementos-chave, alimentando este modelo, através da estratégia e cultura de negócios da organização.

As setas direcionadas para o interior do modelo sinalizam pontos de partida para novos desenvolvimentos, nomeadamente, a identificação da oportunidade e a criação da ideia (Koen, 2018).

3.2.1 Identificação da Oportunidade

Esta fase refere-se ao processo de identificar novos mercados com necessidades não atendidas e emergentes, determinando a solução para o problema identificado no espaço de oportunidades.

Deste modo, o nível competitivo do mercado implica a utilização de técnicas adequadas para explorar as melhores oportunidades que vão de encontro às necessidades dos clientes, tendo em conta a oferta existente.

No contexto do projeto, no nicho do mercado direcionado a empresas de tinturaria e acabamentos do ramo têxtil, a oferta resume-se a produtos maioritariamente limitativos, onde a escalabilidade nesses sistemas obtém-se através de improvisos e remendos. Esta realidade, traduz-se na consequência da adoção de arquiteturas monolíticas, aliadas a tecnologias obsoletas.

3.2.2 Análise da oportunidade

Nesta segunda atividade, reúne-se a informação necessária para converter a oportunidade identificada num modelo de negócio específico.

No seguimento da identificação de oportunidades de mercado, torna-se vital para o crescimento e sobrevivência do negócio, a análise de diversos aspetos em particular, nomeadamente:

- **Análise e segmentação de mercado**: definir uma base de clientes que compartilham características comuns e necessidades, possibilitando assim uma segmentação ao nível de produtos e serviços;

- **Análise da concorrência**: conhecer empresas existentes no mercado, no sentido de entender a proposta de valor e a vantagem competitiva, com a finalidade de identificar o que atrai os clientes alvo;

- **Análise de tendências do mercado**: consiste na pesquisa de tendências tecnológicas relevantes competitivas para o projeto e recolha de informação de potenciais clientes, que auxiliam no desenvolvimento aprimorado do serviço.

No projeto em foco, a indústria do têxtil e vestuário trata-se de um nicho de mercado, que evidencia crescimento consistente ao longo dos últimos anos, não apresentando sinais de decadência, nem qualquer previsão a longo prazo de se tornar num processo produtivo substituível por métodos alternativos.

Assim, a situação pandémica atual, leva as empresas a retraírem-se nos gastos de atualizações de *software*, o que representa um bom *timing* para iniciar um desenvolvimento, de forma a entrar no mercado aquando da normalização.

3.2.3 Criação da ideia e enriquecimento

O terceiro elemento deste modelo, à semelhança da identificação da oportunidade (3.2.1), representa um dos pontos de entrada do *NCD Model*, focando-se na descrição do desenvolvimento e maturação da oportunidade da ideia concreta.

Desta forma, habitualmente trata-se de um processo ativo, incluindo sessões de *brainstorming*, e na criação / reformulação de ideias.

O sistema projetado para desenvolvimento foi idealizado com base em *feedbacks* e agregação de conceções obtidas através de diferentes empresas que, apesar de serem do mesmo ramo, representam diferentes realidades e problemas distintos. Esta abordagem proporciona um desenvolvimento aprimorado e maduro, desde uma fase embrionária.

3.2.4 Escolha da ideia

Este processo pressupõe na escolha da ideia que efetivamente avança e são alocados recursos, de forma a definir prioridades e atividades a realizar, no sentido de obter o maior valor para a organização e o cliente.

O projeto idealizado, no presente desenvolvimento, não contempla todos os desenvolvimentos subadjacentes a esta ideia principal. Selecionaram-se dois módulos que correspondem a departamentos distintos e que melhor se enquadram com os objetivos definidos inicialmente.

3.2.5 Definição de conceitos

Neste processo, frequentemente visto como o resultado do *FEI*, elabora-se o desenvolvimento do *business case*, com base em estimativas das diversas atividades:

- Definir objetivos, períodos associados, e resultados esperados;
- Potencial de mercado e descrição da atratividade do projeto;
- Avaliação do nível de inovação;
- Compreender as limitações do produto;
- Estabelecer parcerias e participação de empresas em testes de produto.

Em relação ao *software* de gestão têxtil, os objetivos primordiais além da pesquisa bibliográfica desenvolvida, passa pela criação de um *Minimum Viable Product* (MVP), ou seja, uma versão simplificada de um produto, com o intuito de testar o encaixe do produto no mercado.

3.2.6 Sistema Stage-gate

"A inovação de produto é um processo dinâmico; começa com a descoberta de novas oportunidades e ideias de produto e termina com um lançamento bem-sucedido de um novo produto." (Cooper, 1990). O modelo *Stage-Gate*, introduzido em meados da década de 1980 por Cooper, foi baseado em pesquisas que se concentraram no trabalho das equipas de desenvolvimento de projetos e organizações de sucesso na criação bem-sucedida de novos produtos.

Um sistema *Stage-Gate* fornece uma rota conceptual e operacional, no sentido de guiar um projeto desde a elaboração da ideia, até ao lançamento do produto. Desta forma, trata-se de um plano para aprimorar a eficácia e eficiência (Cooper, 1990).

Os *stages* assinalam o momento onde determinado trabalho se realiza e cada *gate* serve como um ponto de decisão, ou seja, avançar, eliminar, manter ou reciclar. Desta forma, fornece uma ideia clara de onde o projeto está, em que sentido segue e as tarefas a serem realizadas a seguir (Vliet, 2019).

Contrariamente ao NCD *Model* (3.2), o sistema *Stage-Gate* define um processo sequencial, dividindo o processo em etapas, entre as quais ocorrem determinadas atividades, que resultam em decisões.

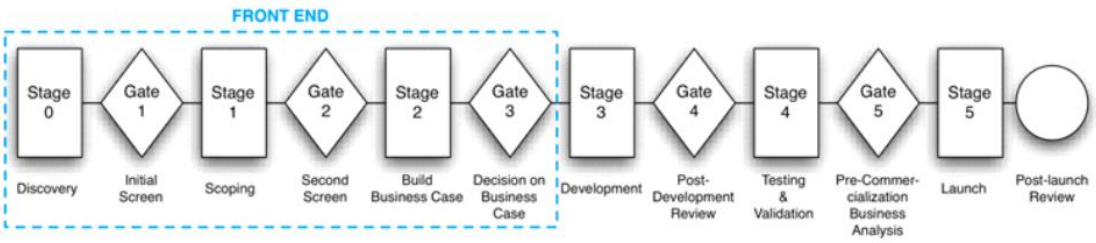


Figura 21 – Representação do sistema *Stage-gate* de Robert Cooper

A

Figura 21 evidencia os diferentes *stages* que compõem o sistema elaborado por Cooper e, como mencionado anteriormente, cada uma destas etapas tem atividades associadas (Cooper, 1990):

- **Etapa 0 (Ideia / Descoberta):** Atividades destinadas a descobrir oportunidades e gerar novas ideias de produtos.

- **Etapa 1 (Avaliação Preliminar):** Avaliação geral sob aspetos económicos e tendências de mercado, de maneira a proceder com uma reavaliação na etapa seguinte.

- **Etapa 2 (Construir Caso de Negócio):** Processo final antes de iniciar o desenvolvimento do produto, onde se verifica a atratividade do projeto antes de alocar recursos. Processos como pesquisa de mercado, avaliação técnica detalhada e análises financeiras detalhada são realizados nesta etapa.

- **Etapa 3 (Desenvolvimento):** Nesta terceira etapa, inicia-se o desenvolvimento do produto e são elaborando-se testes que refinam o mesmo. Igualmente engloba atividades de marketing e elaboração de planos de ação. Análises financeiras são retocadas e tratam-se questões legais / patentes / direitos autorais.

- **Etapa 4 (Teste e Validação):** O objetivo desta etapa é testar a viabilidade do projeto: desde o produto em si, o processo de produção, aceitação do cliente e a sustentabilidade / benefício económico do projeto.

- **Etapa 5 (Lançamento):** A etapa final envolve a comercialização do produto e a execução do plano de marketing associado ao lançamento do *software*, bem como o plano de operações.

- **Avaliação pós-implementação:** Posteriormente, em algum momento, realiza-se uma pós-auditoria, onde dados mais recentes de receitas, custos, despesas e lucros são analisados com uma avaliação crítica. Da mesma forma, ocorre o levantamento dos pontos fortes e fracos do projeto, a aprendizagem obtida e o que pode ser melhorado.

No âmbito do projeto têxtil, as atividades inerentes às etapas anteriores à terceira fase foram especificadas e realizadas na abordagem ao NCD *Model* (3.2). Segue-se o desenvolvimento do produto, seguindo os padrões definidos, tanto na pesquisa bibliográfica, como na análise de valores.

Posteriormente, os testes asseguram a fiabilidade da aplicação, assim como a recetividade do cliente, estando definidas as diferentes categorias de validações a realizar, com detalhe, na secção 6.3.

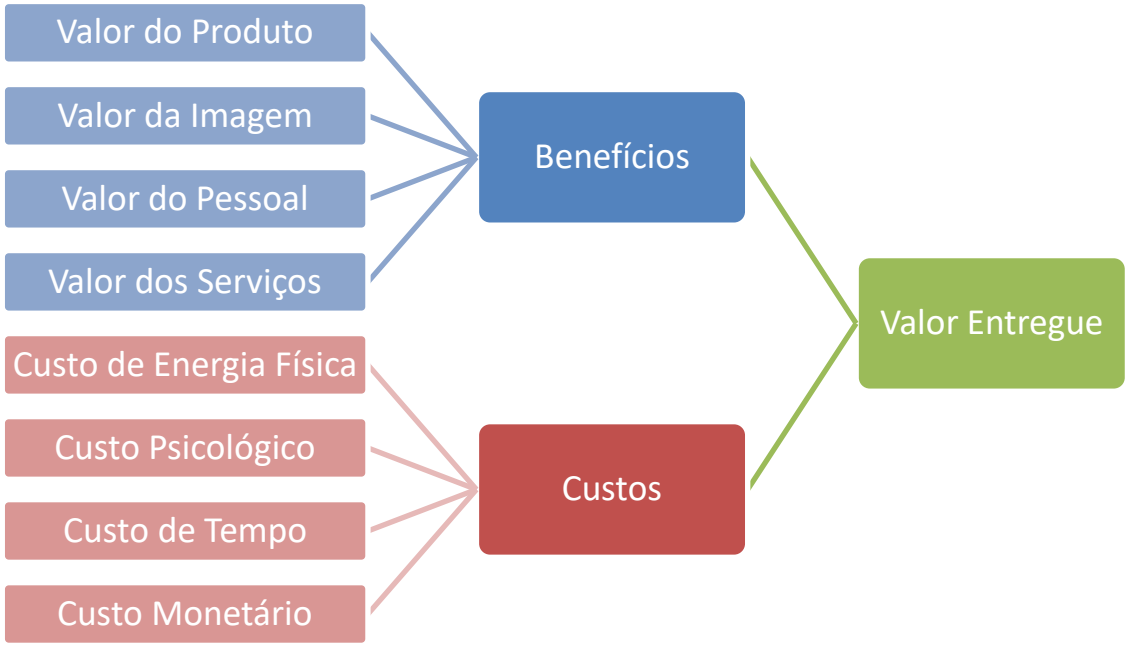
Por último, o lançamento do serviço envolve o emprego da MVP como meio de demonstração de valor do produto, e aquisição de clientes. A avaliação pós-implementação será baseada no *feedback* obtido através do suporte providenciado aos parceiros chave, e auxiliada através da realização de inquéritos igualmente descritos no subcapítulo 6.3.

3.3 Valor, Valor para o Cliente e Valor Percecionado

"O seu produto pode ser incrível. Mas sem uma proposta de valor sucinta, os seus clientes podem nunca saber porquê." (Enfroy, 2020). A ideia errada do valor de um determinado produto pode induzir em erro um cliente e, ainda que um determinado produto seja a solução ideal, o comprador foca-se no benefício que lhe oferece.

No contexto do projeto, as empresas não procuram apenas *software* de gestão, mas sim uma solução e, a forma como as organizações avaliam um determinado sistema, não depende inteiramente de quão bem a solução atende às suas necessidades. Por isso, há vários fatores em jogo que determinam como o cliente avalia o valor de um determinado produto (Kokemuller, 2019).

Deste modo, o valor por vezes não está correlacionado com o preço, mas sim com custos abstratos, sendo determinado pela relação entre os benefícios e os custos que determinam o valor entregue ao cliente.



Fonte: (Paulillo, 2020)

Figura 22 – Fórmula para criação de valor para o cliente

A Figura 22 detalha os fatores inerentes aos benefícios e custos que resulta no valor gerado para o cliente e, desta forma, a Tabela 13 define cada um dos elementos mencionados e, adicionalmente, para cada um deles, como se aplica ao projeto em contexto no presente relatório.

Tabela 13 – Definição de conceitos no valor entregue ao cliente e a aplicabilidade no projeto

	CONCEITO	DEFINIÇÃO	APLICABILIDADE NO PROJETO
Benefícios	Produto	O valor do produto para o cliente baseia-se nos benefícios que o mesmo obtém, em que o serviço evidencia o reconhecimento de um problema e detalha a forma como o produto o colmata	O processo de análise da oportunidade e idealização do produto decorreu junto de empresas que sentem diariamente a ausência de soluções para os objetivos

	CONCEITO	DEFINIÇÃO	APLICABILIDADE NO PROJETO
			propostos neste projeto, garantindo assim que o foco de desenvolvimento se direciona aos problemas sentidos diariamente
	Imagem	O prestígio por detrás de um produto, traduz-se em credibilidade no valor percebido pelo cliente. A confiança na marca, geralmente tem um efeito positivo na satisfação do cliente, aumentando a fiabilidade	A identificação da oportunidade deste projeto decorreu junto de empresas prestigiadas, de dimensão média, que se demonstram disponíveis na aquisição do produto, servindo, assim, como uma rampa de lançamento e obtenção de prestígio numa fase embrionária
	Pessoal	O cliente necessita de confiar na competência da equipa fornecedora do produto que consequentemente se reflete num maior grau de confiança no produto e na estrutura	Prestar um serviço de consultoria de excelência ao cliente, mostrando conhecimento do produto e dos problemas sentidos no terreno e, da mesma forma, conhecer o mercado e os produtos concorrentes.
	Serviços	Consiste no suporte proporcionado ao cliente e na disponibilidade e agilidade na resolução de problemas	Aspeto recorrentemente referido por diversas empresas, enaltecendo a dificuldade de contactar o suporte ou de transmitir problemas. Este processo ocorrerá através de uma plataforma desenvolvida unicamente com o propósito de dar suporte e acompanhar a evolução do processo
Custos	Energia Física	Níveis de stress e energia despendida no processo de análise e negociação do produto	Apresentações objetivas e argumentadas, de maneira a facilitar a compreensão e benefícios obtidos
	Psicológico	O peso da responsabilidade de tomar uma decisão complexa e que pode impactar tanto positiva como negativamente o método de trabalho da empresa	Apresentar casos de estudo e dados concretos que assegurem a confiança e aumentem a segurança do cliente
	Tempo	Custo de tempo na tomada de decisão e no tempo de implementação da solução. Diminuir este custo, reflete na diminuição dos restantes custos	Agilidade no processo de implementação, prazos definidos através de uma *checklist* associadas para clarificar objetivos em cada momento da implementação

	CONCEITO	DEFINIÇÃO	APLICABILIDADE NO PROJETO
	Monetário	O preço do produto deve ser encarado como o fator de menor relevância no processo de venda	O cumprimento dos restantes conceitos resultam numa proposta de valor positiva para o cliente, em que através de um preço justo, evita-se a diminuição do preço, apenas depreciaria o produto

Fonte: (Mansfield, 2018)

3.4 Proposta de Valor

"Faça algo que as pessoas queiram e venda isso, ou seja alguém que as pessoas precisem e venda-se." (Lilly, 2021). Todas as organizações procuram algo ou alguém que lhes ofereça benefícios e lucro aos seus projetos, essa investigação dá-se como finalizada quando são avaliadas todas as possibilidades de valor de cada um dos prestadores, sendo um deles o mais adequado.

O *Value Proposition Canvas* trata-se de um recurso facultado pela *Strategyzer*, o qual veicula as necessidades, dificuldades e ganhos de um cliente ao aderir a um produto (Strategyzer, 2017). Estes atributos servem como método de cálculo da proposta de valor, como representado na Figura 23.

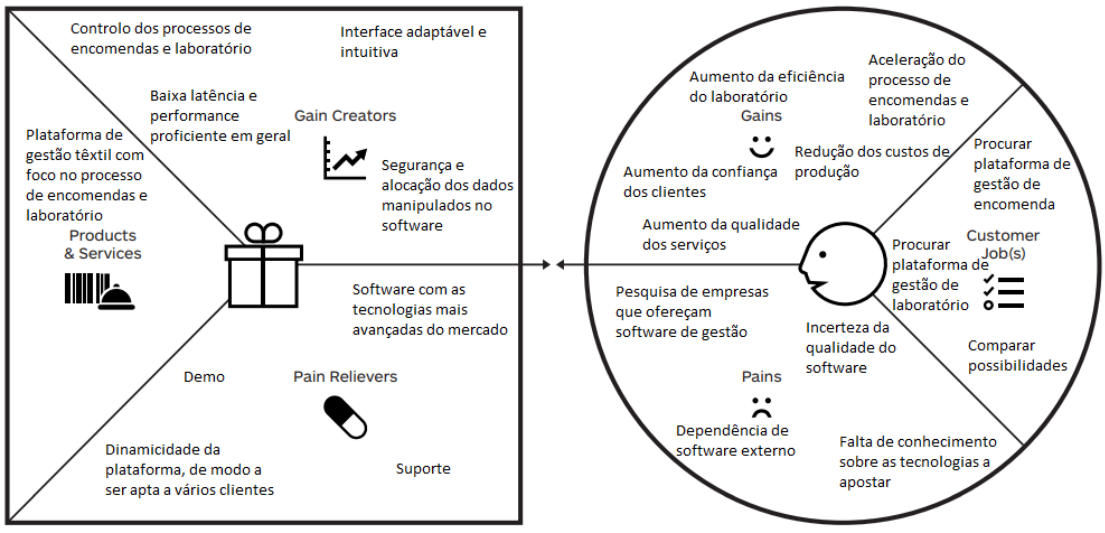


Figura 23 – *Canvas* da Proposta de Valor

No contexto do projeto, orientado a organizações do setor têxtil de tingimento e acabamento, a proposta de valor foca-se na criação de um sistema de gestão, com base nos processos de encomendas e laboratório.

O *software* providencia ao seu cliente uma aceleração e aumento de controlo de ambos os módulos, permitindo um acompanhamento minucioso dos diversos processos e ocorrências. Adicionalmente, oferece vantagens como o aumento da eficiência do laboratório, o que resulta na redução dos custos e, consequentemente na menor utilização de produtos químicos, diminuindo a pegada ecológica da empresa.

Da mesma forma, a aplicação ainda apresentará uma interface de utilizador refinada, interativa e dinâmica concebida a partir das tecnologias de desenvolvimento mais atuais do mercado, o que promove a usabilidade e performance dos seus utilizadores. Resumindo, as características descritas realçam os pontos fortes do produto, aumentando os níveis de confiança dos clientes e o leque de possíveis usufruidores.

3.5 Modelo de Negócio Canvas

"As *startups* não falham pela falta de um produto; elas falham porque não têm clientes e um modelo de negócios lucrativos." (Blank, 2009). O modelo de negócio é uma estratégia muito utilizada para documentar e definir quais os passos necessários para transformar uma ideia em algo palpável, pois o conceito por si só, mesmo que seja excecional, não assegura lucros aos seus potenciais clientes.

Uma das formas de evidenciar o modelo de negócio é através do *Business Model Canvas.* Este trata-se de mais um meio disponibilizado pela *Strategyzer* e encontra-se delimitado por 9 componentes, como demonstrado na Figura 24.

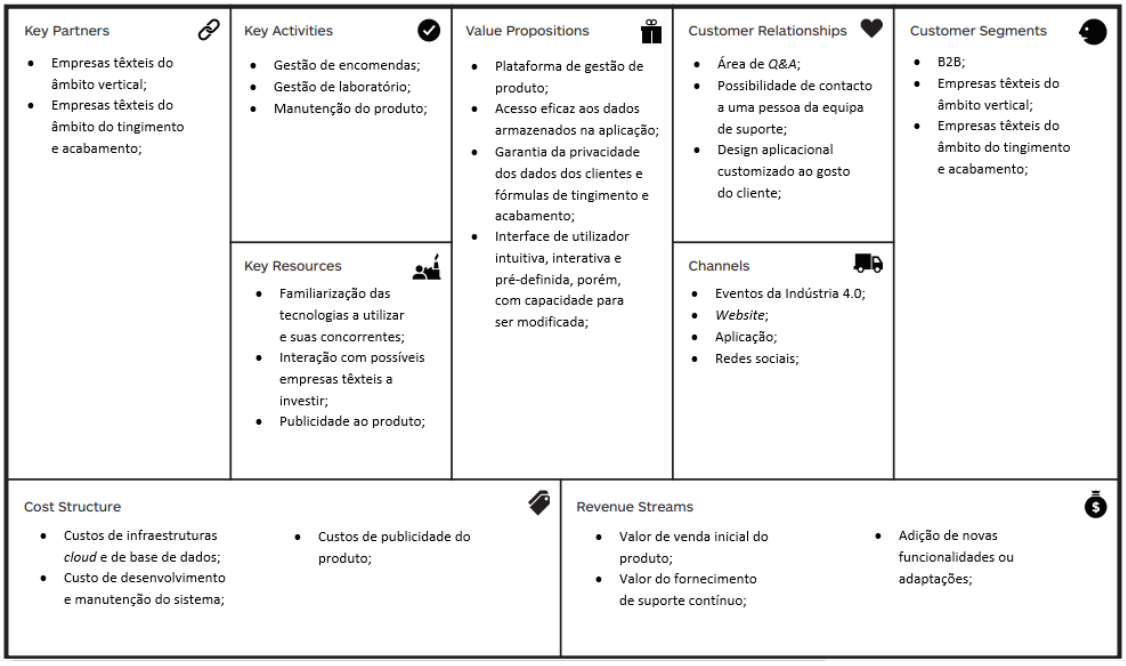


Figura 24 – *Canvas* do Modelo de Negócio

O *Canvas* exibe todos os principais módulos nos quais uma organização deve apostar, com o objetivo de esclarecer qual o caminho a seguir para se atingir o sucesso, apresentando os mesmo num formato sequencial.

3.5.1 Propostas de Valor

Como mencionado anteriormente, a proposta de valor é algo que garante a qualidade e segurança do produto aos olhos do cliente. Assim sendo, este módulo ajuda o cliente a entender a forma como irá ser beneficiado através do alívio das suas angústias, e aumentando o seu lucro (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018).

No âmbito do projeto têxtil de tingimento e acabamento, a principal proposta de valor é a criação de um software de gestão de negócio que oferece vantagens, como o aumento da eficiência e eficácia do processo, levando ao acréscimo dos lucros da empresa. O *BMS* ainda cede propostas de valores secundárias:

- Acesso eficaz aos dados armazenados na aplicação;
- Garantia da privacidade dos dados dos clientes e fórmulas de tingimento e acabamento;
- Interface de utilizador intuitiva, interativa e pré-definida, porém com capacidade para ser modificada.

3.5.2 Segmentos de Clientes

A secção em análise corresponde à identificação de todos os grupos de procuradores possíveis para o produto em constituição. Esses clientes devem ser separados por conjuntos de mercados, como por exemplo mediante a sua cultura (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018).

Em relação ao software de gestão têxtil, os clientes abrangentes são *B2B*, Business to Business, ou seja, os potenciais consumidores são apenas empresas. As instituições interessadas nesta aplicação são da área têxtil com origem no tingimento e acabamento, no entanto, existem organizações têxteis de produção vertical que recorrem a plataformas diferentes para cada módulo, o que avoluma o leque de clientes.

3.5.3 Canais

Este módulo é o intermediário entre a proposta de valor e os segmentos de clientes, ou seja, após a definição do público-alvo e do produto, o objetivo é saber como fazer com que a ideia seja entregue ao cliente. (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018)

Os canais de distribuição de aplicações têxteis devem ter origem em eventos conhecidos mundialmente ou por continente, como o exemplo da ITMA, uma cadeia de valor integrada de manufatura têxtil e de vestuário, contudo os seus preços são exorbitantes. Como hipótese para pequenas e médias empresas existem *webinars* e outros tipos de eventos da indústria 4.0 que justificam a presença. Ainda há a possibilidade de se partilhar informação do produto através do *website* ou aplicação *web* do próprio ou até mesmo a partir das redes sociais.

3.5.4 Relacionamentos com Clientes

A fase em questão já corresponde à interação com o cliente, especificamente em como o manter mesmo após a venda. Por outras palavras trata-se da fidelização do cliente, o conservar da sua satisfação, não só pelo fornecimento do produto, mas também pelo atendimento (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018).

Para manter o cliente integrado com o sistema de gestão têxtil é imprescindível o contacto e comunicação constantes, de modo a receber o *feedback* da plataforma e, com isso, desempenhar ações que surpreendam pela positiva. Como opções há o uso de *Q&A*'s no website e aplicação mobile do produto e a possibilidade de se contactar um membro da equipa de suporte. Uma outra característica fundamental e muito requerida no mundo têxtil é a dinâmica de um menu, pois num dia podem desejar a interface de uma maneira, como noutro dia alterar completamente essa mesma interface.

3.5.5 Fontes de Receita

Após a definição dos módulos relacionados ao cliente, deve ser feita uma leitura global que auxilie na decisão de valores a cobrar. A receita a obter pode ser por intermédio de venda final do produto, por serviço, por mensalidade, entre outras possibilidades (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018).

No contexto da ferramenta têxtil, usualmente, existe sempre um valor de venda inicial do produto que é calculado a partir dos gastos temporais e financeiros do seu concebimento (licenças das tecnologias, hardware, tempo de desenvolvimento e análise). Posteriormente, as remunerações são obtidas através do suporte prestado, novas funcionalidades e manutenção da aplicação.

3.5.6 Recursos Chave

A primeira fase de entre as quais se encontra direcionada às necessidades da empresa que deseja conceber um novo produto é a identificação dos recursos chave. Corresponde à seleção dos meios essenciais indispensáveis para o sucesso da proposta de valor, que podem ser tanto matéria-prima como mão-de-obra (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018).

Relativamente ao projeto têxtil, como se trata de um software, não haverá o problema de procura por matéria-prima. No entanto, em primeiro lugar, deverá ser feito o estudo das tecnologias de desenvolvimento a utilizar, de modo a oferecer ao cliente o software mais atualizado do mercado nas melhores condições. Um outro recurso chave é a comunicação com as empresas têxteis, de forma a obter o *feedback* delas quanto às aplicações concorrentes e possíveis sugestões de melhoria. Para finalizar, o último recurso chave fundamental trata-se de obter vias de publicidade à plataforma, para introduzir o produto no mercado e aumentar a gama de clientes.

3.5.7 Atividades Chave

Os recursos chave por si só não são capazes de se converter em propostas de valor, para isso é fulcral a laboração de atividades chave. Estas práticas correspondem, por exemplo, ao estudo e execução de técnicas de desenvolvimento de produtos (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018).

No enquadramento da plataforma de gestão têxtil, mais especificamente na conjuntura da dissertação, as atividades chave principais são a elaboração dos módulos de encomenda e laboratório, que incluem a manipulação de encomendas, cartazes, cores e ensaios. Para além das atividades anteriores, ainda existe a manutenção do sistema, que não deve ser menosprezada, pois é o que mantem um software sem erros e atualizado.

3.5.8 Parceiros Chave

O módulo em foco é responsável pelo reconhecimento dos fornecedores, que podem ser uma organização ou um trabalhador independente. Os recursos obtidos pelos parceiros podem ser materiais ou não materiais, como informações cruciais para o sucesso do produto (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018).

No caso do sistema de gestão têxtil, os parceiros chave são as empresas têxteis, ou seja, os próprios clientes. Como se trata de um software que engloba uma lógica de negócio massiva e complexa não existe ninguém melhor que o próprio consumidor para elucidar e clarificar o que fazer e como fazer. O projeto têxtil dá cerne ao tingimento e acabamento, portanto os parceiros chave preferíveis são as empresas desta área, que oferecem também soluções para os problemas das ferramentas atuais do mercado.

3.5.9 Estrutura de Custos

Por último, a estrutura de custos equivale ao conjunto de todos os gastos do plano de negócios. Os custos podem ser financeiros ou temporais e podem ser provenientes dos recursos chave, das atividades chave, dos canais de entrega, do relacionamento com o cliente e até mesmo dos

parceiros chave (Bernardo, Fast MBA - Empreendedorismo, Negócios e Startups na Prática, 2018).

Nas circunstâncias do software têxtil, a maior parte dos custos financeiros encontram-se associados às tecnologias de desenvolvimento, nomeadamente, o uso de infraestruturas *cloud*, de licenças de software e de base de dados. Os restantes custos financeiros consistem na publicidade, com o objetivo de aumentar exponencialmente o número de pessoas à qual a plataforma alcança. O último tipo de custo é o temporal, pois criar um sistema complexo como este exige um longo período, tal como diz o ditado, "tempo é dinheiro".

3.6 QFD

"Entre o risco calculado e a tomada de decisão imprudente está a linha divisória entre os lucros e as perdas." (Duhigg, 2016). A tomada de uma decisão deve ser feita cautelosamente, após uma análise intensiva, pois trata-se de uma ação que pode afetar o futuro de uma empresa. Existe um vasto número de métodos de tomadas de decisão e avaliação de um produto. Estas técnicas são simples, no entanto poderosas e essenciais para evitar erros e facilitar a manutenção da aplicação.

Um dos métodos é o *QFD*, nomeadamente *Quality Function Deployment*, que é essencialmente utilizado para o planeamento do processo produtivo, ou seja, a sua arquitetura baseia-se na projeção de um produto com origem nos requisitos do cliente. Foi inicialmente implementado em 1972 na Mitsubishi, criando uma matriz onde os eixos verticais representavam os requisitos do cliente e os eixos horizontais os métodos pelos quais cada requisito seria gerido. Isto levou ao aperfeiçoamento da comunicação entre as várias equipas, e concorrentemente ao aumento de eficiência da organização (Warwick Manufacturing Group, 2007).

Reforçando a ideia, a matriz concebida a partir do método em questão ajuda no interpretar das demandas de um cliente, facilitando a troca de informação entre designers, desenvolvedores, gestores, entre outros membros da mesma empresa. Deste modo, este processo acarreta os seguintes benefícios (Quality-One, 2018):

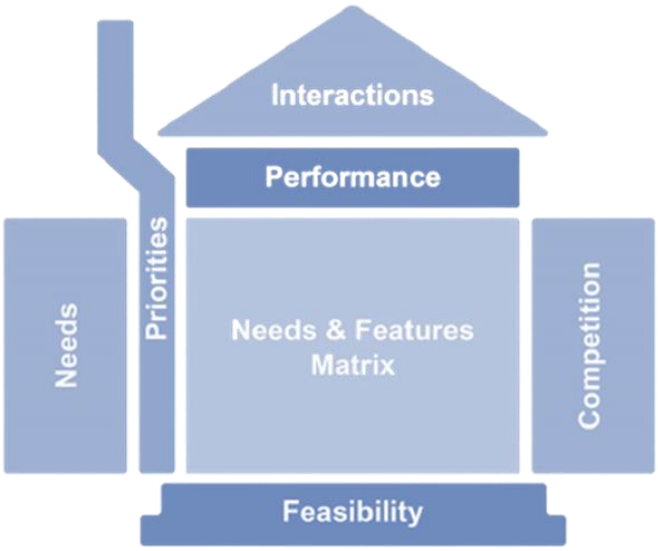
- Método focado no cliente, por outras palavras, grandes quantidades das novas empresas fraquejam porque criam requisitos com vista no que acreditam que seja melhor para o consumidor, ao invés de darem ênfase aos desejos e necessidades do mesmo;

- Ferramenta *VOC*, ou *Voice of the Customer*, tem como objetivo transformar os desejos e necessidades do cliente em requisitos representados numa estrutura hierárquica, separados por importância e satisfação dos futuros utilizadores do produto;

- Diminuição dos custos de desenvolvimento, isto é, ao definir desde a génese do projeto os requisitos realmente importantes e estabelecendo uma arquitetura coerente aos mesmos, evitar-se-ão desperdícios e reformulações;

- Documentação, por outros termos, o método QFD dispõe de ferramentas que registam informações sobre as decisões tomadas que podem servir como forma de transferir conhecimento entre todos os membros de uma organização.

A matriz utilizada no método *QFD* para agrupar os requisitos do cliente com as técnicas logradas pela organização com o intuito de resolver as suas exigências denomina-se de *House of Quality* (*HOQ*). Tal como o nome indica, segue uma estrutura semelhante a uma casa, em que cada secção corresponde a uma particularidade a ser avaliada, como é possível visualizar-se na Figura 25.



Fonte: (Fedon, 2020)

Figura 25 – *House of Quality*

- ***Feasibility*** - Trata-se da secção que faz a comparação entre os atributos técnicos do produto em conceção e dos produtos concorrentes, principalmente em termos de performance;

- ***Needs/Priorities*** - Corresponde à fração que engloba os *Voice of the Customer* (*VOC*) a qual equivale aos requisitos imprescindíveis ao cliente;

- ***Competition*** - Tal como a base da *HOQ*, realiza uma comparação entre o produto em análise e os seus concorrentes, contudo essa avaliação é feita a partir dos requisitos do cliente;

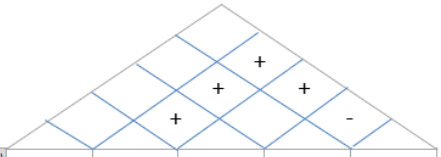
- ***Performance*** - Diz respeito à seleção de todas as primordialidades técnicas do produto, com base nos requisitos do cliente;

- ***Interactions*** - É a matriz de correlação que tem como objetivo calcular o peso dos requisitos técnicos com foco nas suas similitudes;

- ***Needs & Features Matrix*** - Representa a matriz de relação que avalia todos os requisitos técnicos e de cliente identificados, tendo em atenção as comparações com produtos concorrentes.

Assim sendo, a matriz da casa possui as suas próprias matrizes internas, nas quais são feitas apreciações. A matriz de correlação evidencia as relações entre os requisitos técnicos, que podem ter uma conexão simbiótica ou prejudicial, sendo esses vínculos representados respetivamente por "+" e "-". Por outro lado, a matriz de competição, que indica para cada requisito do cliente qual o produto com melhores competências. Ainda existe a matriz de viabilidade, a qual avalia o futuro produto aos seus concorrentes segundo os requisitos técnicos. Por último, a matriz central ou de relação que contém o desenlace do produto relativamente a todos os dados presentes na *HOQ*.

A Figura 26 retrata a casa de qualidade com os respetivos requisitos técnicos e expectativas de cliente, destacando as metas, ações e procedimentos no desenvolvimento deste projeto. Da mesma forma, estas mesmas expectativas do cliente são devidamente quantificadas para os softwares concorrentes.



Correlação:	+		-
	Positiva	Sem	Negativa

Relação:	9	3	1	0
	Forte	Moderada	Fraca	Sem

	Direção desejada (↑,0,↓)										
1: baixa, 5: alta	**Requisitos Técnicos (Como?)** → **Requisitos do Cliente - (O quê?)** ↓	Interface dinâmica e dúctil	Disponibilidade	Registo de logs	Sistema escalável	Base de dados estruturada	Classificação Ponderada	Avaliação Competitiva (1: baixo, 5: alto) Rating Satisfação	Texplus	4Tex	MacWin
Importância para o Cliente											
4	Interface gráfica intuitiva	9	0	0	0	0	36		2	3	4
3	Baixa latência	9	9	1	9	3	93		2	3	4
3	Compatibilidade com vários Sistemas Operativos	9	1	3	3	3	57		4	1	4
5	Privacidade dos dados	3	9	3	3	9	135		3	3	3
4	Suportar vários utilizadores concorrentemente	0	9	3	9	9	120		3	3	4
4	Capacidade de interoperabilidade	0	0	0	9	9	72		3	2	4
2	Suporte a várias línguas	9	0	0	3	3	30		2	2	3
							0				
							0				
	Importância Técnica	123	111	39	129	141	543				
	Importância %	23%	20%	7%	24%	26%	100%				
	Prioridade	3	4	5	2	1					
	Performance Atual										
	Objetivo	80% das interfaces dinâmicas e 100% dúcteis	Aplicação 99% online	100% de logs quanto a conexões API e queries	Aguentar mais que 10 utilizadores ao mesmo tempo	90% abaixo dos 3 segundos de duração					
	Benchmark										
	Dificuldade	5	1	3	2	4	1: very easy, 5: very difficult				
	Custo e Tempo	5	2	3	2	3	1: low, 5: high				
	Prioridade para melhorar	**25**	**2**	**9**	**4**	**12**					

Figura 26 – *House of Quality*, Projeto Têxtil

Na figura podem ser identificados os requisitos técnicos e de cliente identificados no âmbito do projeto têxtil, os quais estão espelhados na Tabela 14.

Tabela 14 – Requisitos Técnicos e de Cliente

REQUISITOS DO CLIENTE	**REQUISITOS TÉCNICOS**
Interface gráfica intuitiva	Interface dinâmica e dúctil
Baixa latência	Disponibilidade
Privacidade dos dados	Registo de *logs*
Capacidade de interoperabilidade	Sistema escalável
Suporte a várias línguas	Base de dados estruturada
Compatibilidade com vários Sistemas Operativos	
Suportar vários utilizadores concorrentemente	

Quanto aos requisitos de cliente, não existe uma empresa no mercado atual que satisfaça por completo as exigências dos consumidores. Tal como mencionado no subcapítulo 2.2.2, as competências oferecidas pelas grandes empresas de software de gestão têxtil não são as mais amistosas, sendo isto confirmado pela casa de qualidade, isto leva à supressão da fidelização do cliente. Apesar da *MacWin* disponibilizar as melhores condições atualmente, não tem as capacidades de gestão de uma ferramenta focada no tingimento e acabamento.

Em relação aos requisitos técnicos, essenciais para o bom funcionamento da aplicação têxtil, não possuem muitas correlações, destacando-se a negativa entre o sistema escalável e a base de dados estruturada, pois uma *database* relacional tem dificuldades a escalar, quando comparada com uma não relacional. Ainda existem correlações simbióticas como:

- Disponibilidade <-> Registo de *logs*: quando uma funcionalidade do sistema falha, o registo de *logs*, habitualmente, detém indícios de onde ocorreu a falha, facilitando a pesquisa e correção do *bug*;

- Disponibilidade <-> Sistema escalável: por norma, um sistema altamente escalável encontra-se subdividido por módulos, esta arquitetura acarreta vantagens como uma elevada disponibilidade, visto que ao ocorrer uma falha num módulo não irá obrigatoriamente afetar todo o software;

- Disponibilidade <-> Base de dados estruturada: a disponibilidade da base de dados tem de ser total, impreterivelmente, caso contrário não será possível efetuar a manipulação dos dados;

- Base de dados <-> Registo de *logs*: frequentemente todas as *queries* são registadas nos *logs*, ou seja, quando uma chamada à base de dados retorna um erro, este é escrito no ficheiro de *logs*, facilitando a correção do software.

Adicionalmente, os atributos técnicos que devem ser respeitados pelo software têxtil têm os seus próprios pesos, em que, como previsto, os de maior importância e prioridade exigem maior sacrifício em termos de custo, tempo e dificuldade.

Em suma, a *House of Quality,* além de sintetizar de forma sucinta quais os requisitos cruciais para o sucesso de um projeto, também avalia as suas reciprocidades e calcula características significativas para cada um, nomeadamente, prioridade, dificuldade, custo e tempo. Estas medidas são baseadas em comparações com produtos já existentes no mercado e na definição de objetivos primordiais.

4 Análise de Requisitos e Conceção

"Arquitetura de software trata-se de um conjunto de decisões de *design,* que se tomadas incorretamente, podem fazer com que seu projeto seja um fiasco." (Woods, 2011). O presente capítulo proporciona uma visão do produto desenvolvido e da sua arquitetura, e adicionalmente do processo de engenharia adjacente à sua arquitetura e implementação.

A subsecção inicia-se pela engenharia de requisitos, abordando os requisitos funcionais e não funcionais, oferecendo uma perspetiva global da arquitetura do sistema e da interligação entre os diversos conceitos do domínio.

4.1 Engenharia de Requisitos

"O aspeto mais importante do desenvolvimento de software é ser claro sobre o que se está a tentar construir." (Stroustrup, 2000). O presente capítulo diz respeito à fase mais fundamental da engenharia de software, a definição de requisitos do projeto.

Este estágio é responsável pela transformação das ideias em negócio, o que acarreta as suas dificuldades, sendo esta a maior razão de falhas de sistemas com bom potencial. Em março de 2020, a famosa empresa *Space X*, dedicada à construção de naves aeroespaciais, sofreu um dispêndio de milhões de euros devido a uma análise de requisitos errónea no cálculo de dados (Foust, 2021).

A secção atual contém a análise realizada, contendo os requisitos funcionais e não funcionais, além dos casos de uso e conceitos do domínio.

4.1.1 Modelo FURPS+

As condições que uma aplicação deve cumprir de modo a satisfazer um cliente são especificadas nos requisitos, o que dá lugar à necessidade de os categorizar. O modelo FURPS simplifica a classificação de cada requisito, enfatizando num grau superior a compreensão das qualidades que um software dispor.

O modelo em questão engloba os requisitos funcionais, requisitos não funcionais e limitações não funcionais (Ottinger & Langr, 2009). A primeira categoria retrata as funcionalidades da aplicação, nomeadamente:

- F <-> *Functionality* (Funcionalidade);
- U <-> *Usability* (Usabilidade);
- R <-> *Reliability* (Confiabilidade);
- P <-> *Performance* (Performance);
- S <-> *Supportability* (Suportabilidade).

Sendo o sinal de soma a representação das limitações não funcionais, que tomam em conta necessidades adicionais que um cliente poderá ter, particularmente:

- *Design Constraints* (Limitações de Desenho);
- *Implementation Constraints* (Limitações de Implementação);
- *Interface Constraints* (Limitações de Interface);
- *Physical Constraints* (Limitações Físicas).

4.1.2 Requisitos funcionais

O tipo de requisitos em foco, definem o que um software deve fazer, descrevendo os seus serviços, tarefas ou funcionalidades (AltexSoft, 2021). Existem diversas formas de representar os requisitos funcionais, tal como:

- Documento de especificação de requisitos de software: retrata num documento o que um software irá fazer e como;
- Casos de uso (*Use cases*): identificam as funcionalidades de uma aplicação com base nos requisitos do sistema;
- Histórias de Utilizador (*User stories*): idênticos aos casos de uso, no entanto são baseadas na interação do utilizador;
- Decomposição funcional (*Functional decomposition*): descrevem os passos necessários para o funcionamento de um processo;
- Modelos e diagramas: descrevem a partir de um desenho a estrutura e comportamento de um sistema.

No contexto do projeto foram utilizados os casos de uso em conjunto com modelos e diagramas, nos quais os primeiros dão foco na sintetização do comportamento de um utilizador numa

determinada funcionalidade (que estão descritos em formato de *user stories*) e os segundos descrevem as funções de alto nível no domínio do sistema.

4.1.2.1 Casos de uso

"Um caso de uso é um guia completo de eventos no sistema, visto da perspetiva do utilizador." (Jacobson, 1992). O sistema é gerido através de eventos que condicionam o seu estado, e estes são despoletados por um ator, usualmente o utilizador (Usability, 2013). Cada caso de uso especifica como uma determinada funcionalidade deve agir para os possíveis eventos.

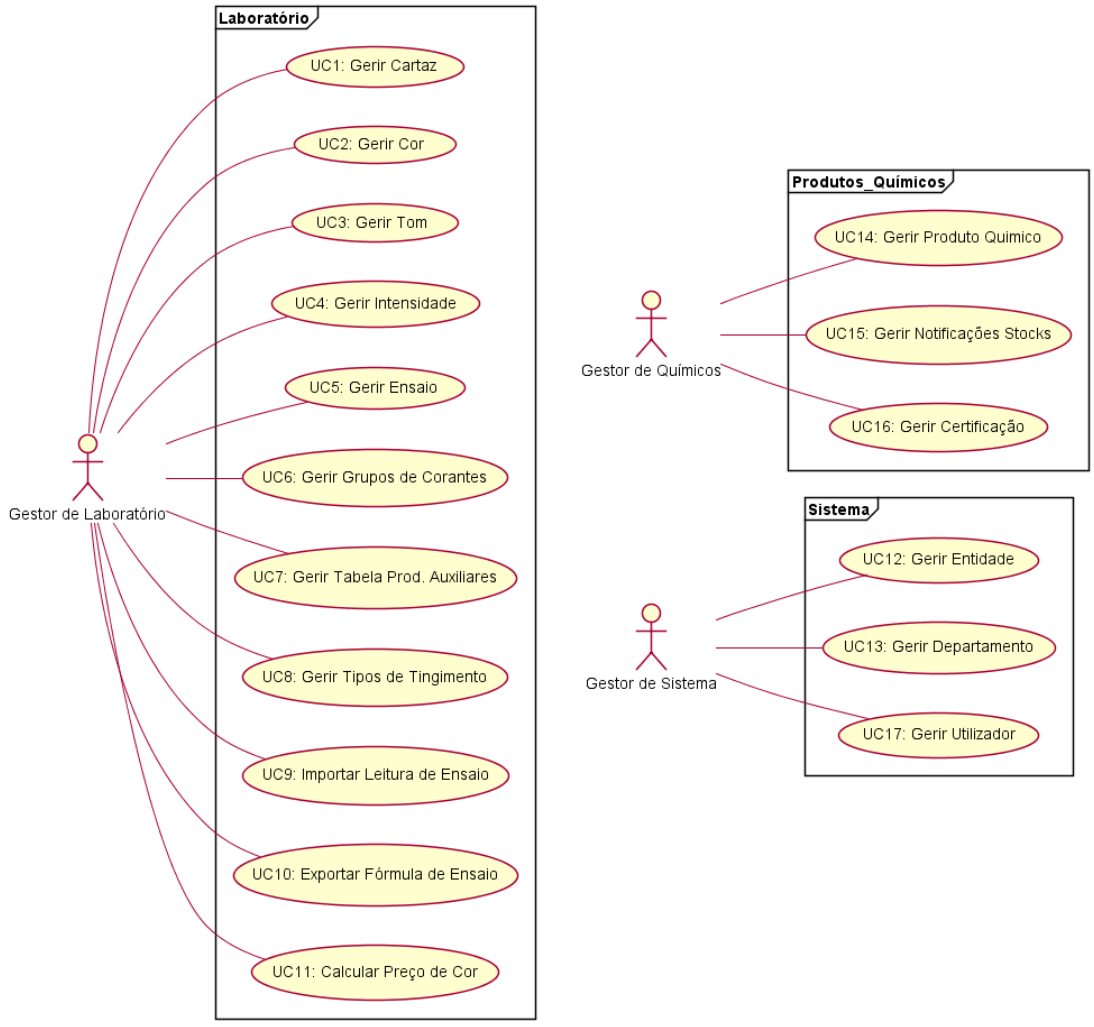


Figura 27 – Diagrama de Casos de Uso

Como exibido na Figura 27, o projeto de gestão têxtil apresenta múltiplos atores responsáveis por diferentes funcionalidades às quais têm acesso segundo a posição que ocupam na empresa. A descrição individual dos casos de uso encontra-se no Anexo E .

4.1.2.2 Gerir Cor (UC2)

A gestão de cor, agrega múltiplas funcionalidades para o utilizador. A Figura 28 ilustra a expansão dos casos de uso relativos à gestão de cor.

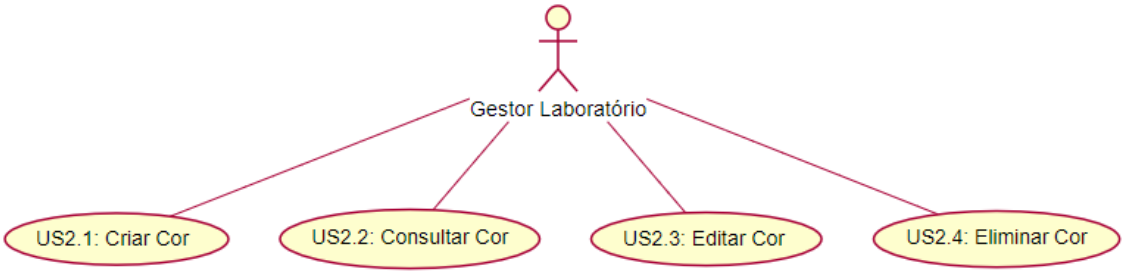


Figura 28 – Casos de uso da gestão de cor

UC2.1: Criar Cor

A criação de uma cor por parte do gestor de laboratório pode realizar-se em dois contextos distintos:

- Na sequência de um cartaz de cores proveniente de um cliente, onde as cores são criadas independentemente umas das outras, em que apenas partilham o mesmo número de cartaz.

- Por iniciativa da própria empresa de tinturaria e acabamentos, ou seja, por vezes, a organização pretende criar as suas cores sem depende de um pedido de um cliente, a fim de propor cores já elaboradas pela própria organização.

Posteriormente, permite ao utilizador associar diferentes ensaios, cada um com a sua fórmula correspondente, na tentativa de reproduzir a cor pretendida (UC4 – Gerir Ensaio, mencionado no Anexo F).

UC2.2: Consultar Cor

A opção de consultar os dados de uma cor, assemelha-se a uma funcionalidade de pesquisa, permitindo ao gestor de laboratório procurar e recuperar registos específicos na tabela de cores, e consultar atributos específicos de uma determinada cor.

UC2.3: Editar Cor

A função de edição, proporciona ao gestor de laboratório a oportunidade de modificar as cores existentes na base de dados. Qualquer um dos campos que compõe a cor, podem ser alterados pelo utilizador.

UC2.4: Eliminar Cor

O sistema proporciona ao gestor de laboratório, o direito de remover cores desnecessárias da base de dados. Os dados não são complementa excluídos do sistema, tratando-se de um *soft delete*, em que simplesmente o estado da cor é alterado, não sendo listada nos registos visíveis para o utilizador, no entanto, a informação permanece presente e intacta na base de dados.

Tabela 15 – Análise integral ao UC2.1

UC2.1: Criar Cor

Como gestor de laboratório, pretendo criar uma cor.

Critérios de Aceitação	
AC01:	A cor deve ser criada no sistema
AC02:	O código da cor deve ser único, e com tamanho máximo de 12 caracteres
AC03:	O nome da cor não deve ultrapassar os 35 caracteres
AC04:	O *pantone* deve seguir o seguinte formato: 99-9999 XYZ, ou seja, 2 números seguidos de um hífen, 4 números e por último 3 letras
AC05:	A entidade deve estar criada previamente no sistema e ser do tipo "Cliente";
AC06:	A intensidade deve estar criada previamente no sistema
AC07:	O tom deve estar criado previamente no sistema
AC08:	Todos os campos são obrigatórios
AC09:	O nome e o código devem ser únicos para cada cor de um respetivo cliente. Quando esta condição não for cumprida, o sistema deve rejeitar a operação e o utilizador deve ter a oportunidade de modificar os dados inválidos.

Dependências:	Criar entidade do tipo cliente (UC12.1)
	Criar tom de cor (UC3.1)
	Criar intensidade de cor (UC4.1)

Dados de entrada e saída	Dados Introduzidos	Código de cor
		Nome de cor
		Pantone
	Dados selecionados	RGB
		Cliente
		Intensidade
		Tom
	Dados Saída	(In) Sucesso da operação

Diagrama de Sequência do Sistema (SSD)

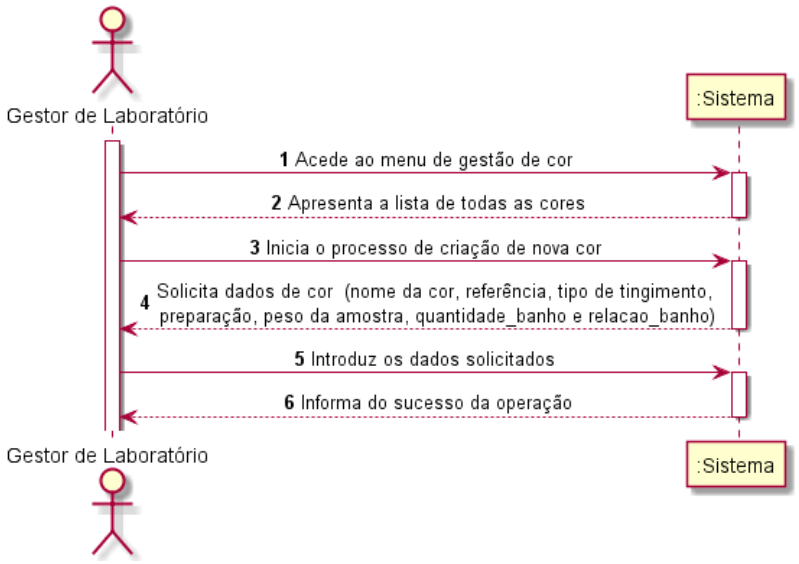


Figura 29 – Diagrama de sequência do sistema UC2.1

A referência indica as fibras e a percentagem de cada uma delas que compõe a malha a tingir. O tipo de tingimento permite filtrar posteriormente a gama de corantes que vão ser utilizados na fórmula da cor.

O peso da amostra, a quantidade de banho e a relação de banho são valores que influenciam no cálculo das quantidades dos corantes tendo em conta que na fórmula a unidade é percentua.

Sistematização	
Da base lógica resulta que os componentes criados no sistema são:	Cor
	Cartaz
	Referência

Outros componentes criados não identificados no domínio:	RegistarCorUI
	RegistarCorController
	CorRoutes

Diagrama de Sequência

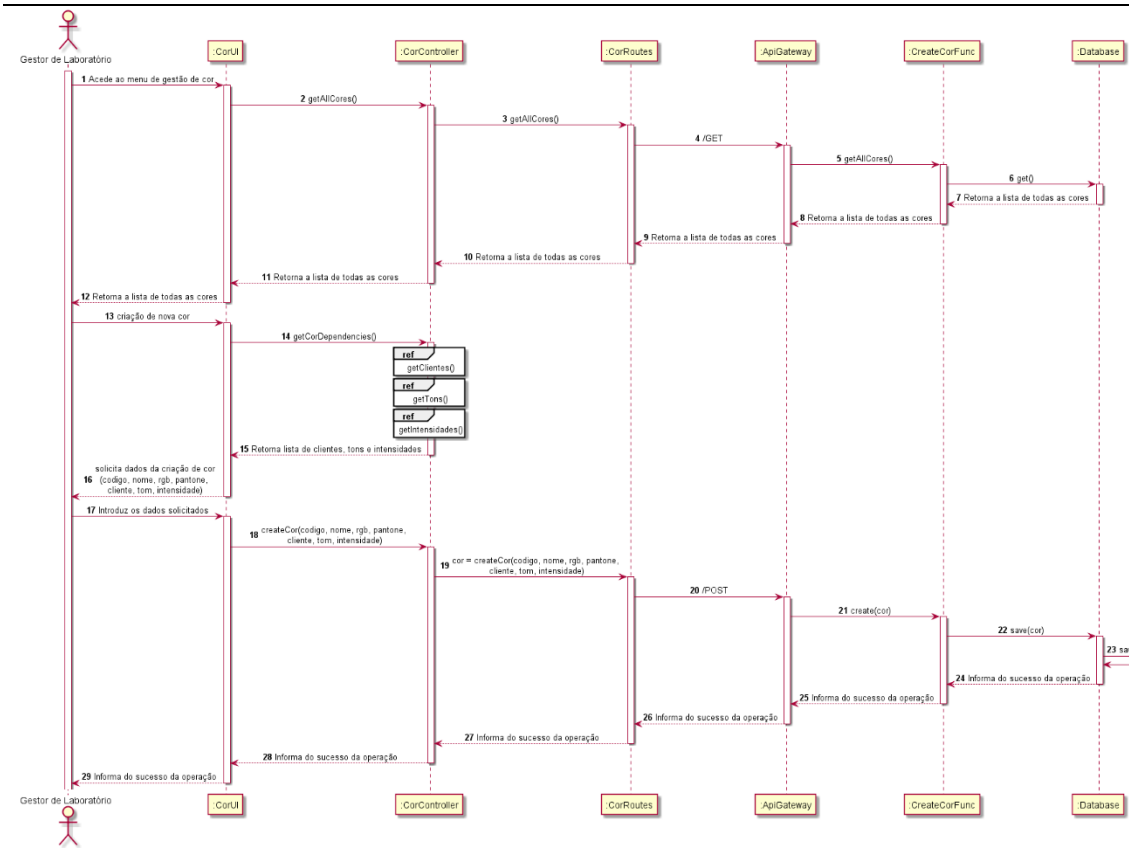


Figura 30 – Diagrama de sequência do UC2

O diagrama de sequência apresenta na Figura 30, é composto por uma ação não especificada, nomedamente a obtenção da lista de todas as entidades, onde a especificação da mesma é apresentada num diagrama independente.

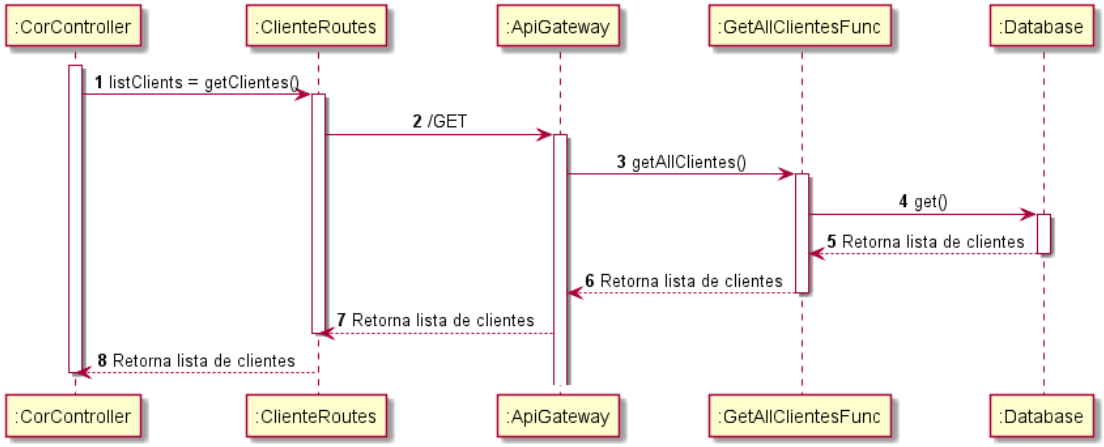


Figura 31 – Diagrama de sequência referente à obtenção de todos os clientes

As restantes dependências, procedem-se pela mesma lógica que foram obtidos os clientes, no entanto, os diagramas relativos às restantes ações encontram-se no Anexo F .

Quanto aos **casos de uso que englobam a gestão de um elemento do domínio seguem a mesma lógica e fluxo** que esta mesma lógica mencionada na Tabela 15. Os diagramas de sistema e sequência de todos as operações apresentam-se detalhados no Anexo F .

4.1.2.3 Calcular Preço de Cor (UC11)

Tabela 16 – Análise integral ao UC11

UC11: Calcular Preço de Cor
Como gestor de laboratório, pretendo calcular o custo de uma cor.

Critérios de Aceitação	
AC01:	O preço deve ser registado no sistema
AC02:	O preço não pode ser igual nem inferior a 0
AC03:	O preço não é um campo obrigatório
AC04:	A cor e o ensaio devem estar registados previamente no sistema
AC05:	O ensaio deve conter uma fórmula associada

Dependências:	Criar cor (UC2.1)
	Criar ensaio (UC5.1)

Dados de entrada e saída	Dados Introduzidos	N/A
	Dados selecionados	Cor
		Número do Ensaio
	Dados Saída	Custo do ensaio

Diagrama de Sequência do Sistema (SSD)

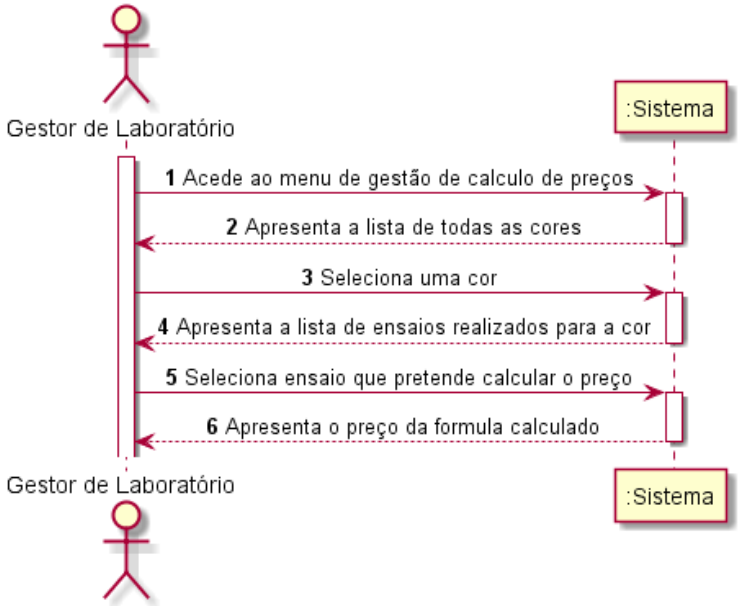


Figura 32 – Diagrama de sequência do sistema UC11

A cor é constituida por diversos ensaios, e por isso, o utilizador especifica o ensaio o qual pretende calcular o preço. O preço varia de ensaio para ensaio, porque cada um deles tem a sua fórmula específica.

O resultado devolvido pelo sistema indica o custo que a empresa tem ao produzir aquela cor, com base nos produtos utilizados e o preço unidade de cada um. Este cálculo servirá

posteriormente como indicador para o preço a fatura ao cliente, no entanto esse departamento não se aplica ao contexto do projeto.

Sistematização	
Da base lógica resulta que os componentes criados no sistema são:	Cor
	Cartaz
	Referência

Outros componentes criados não identificados no domínio:	RegistarCorUI
	RegistarCorController
	CorRoutes

Diagrama de Sequência

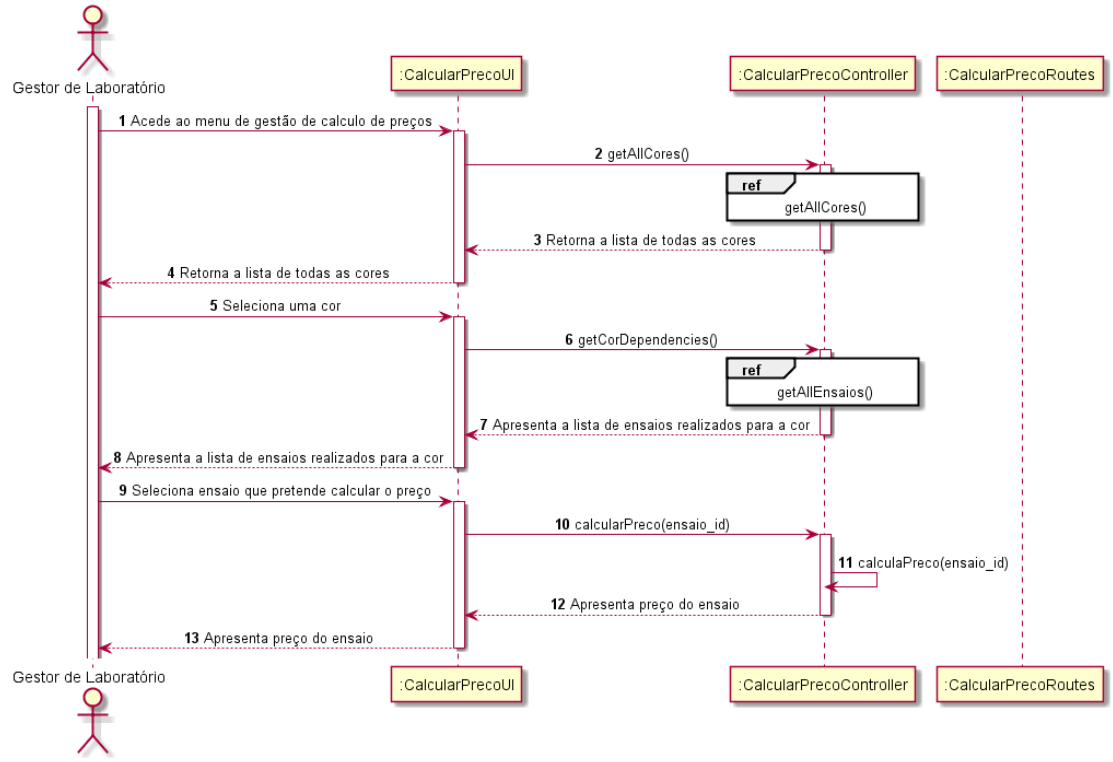


Figura 33 – Diagrama de sequência do UC11

O cálculo do custo de um ensaio tem como propósito definir o preço de venda ao cliente quando forem faturadas encomendas.

Neste caso, como a parte contabilistica não está contemplada no projeto, o preço é meramente indicativo dar uma ideia do custo aproximido da fórmula utilizada, não havendo nenhuma informação guardada por parte do utilizador após a realização desta ação.

Adicionalmente, O diagrama de sequência apresentado na Figura 33, é composto por diversas ações não especificadas, nomeadamente, a obtenção de todos os ensaios de uma determinada cor, onde a especificação das mesmas são apresentadas em diagramas independentes.

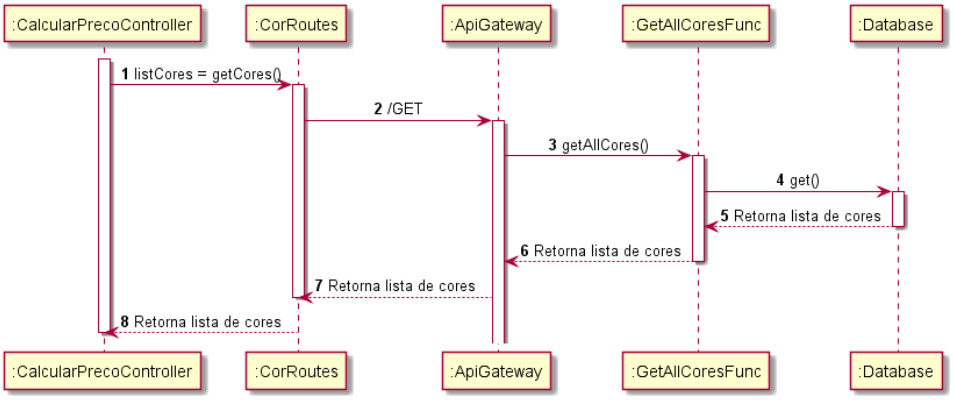


Figura 34 – Diagrama de sequência referente à obtenção de todos os cartazes

4.1.2.4 Gerir Notificações de Stocks (UC15)

Tabela 17 – Análise integral ao UC15

UC15: Gerir Notificações de Stocks
Como gestor de químicos, pretendo definir um limite de stock que notifica o ator quando o mesmo for atingido.

Critérios de Aceitação	
AC01:	O limite deve ser registado no sistema
AC02:	O limite não pode ser inferior a 0
AC03:	O limite quando atingido deve notificar o utilizador no sistema e via email
AC04:	O produto deve estar registado previamente no sistema

Dependências:	Criar produto (UC14.1)

Dados de entrada e saída	Dados Introduzidos	Valor do limite
	Dados selecionados	Tipo de limite
	Dados Saída	(In)Sucesso da operação

Digrama de Sequência do Sistema (SSD)

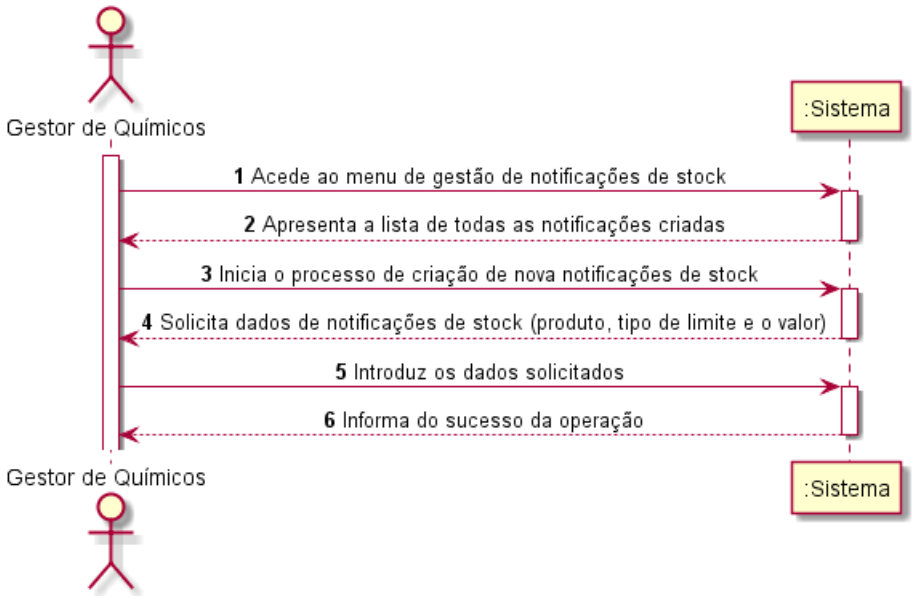


Figura 3535 – Diagrama de sequência do sistema UC15

Tipos de limite que dispoleta uma notificação:

- Quantidade Mínima - Stock inferior à quantidade definida
- Número Máximo de Dias - Nº de Dias excede a Data da Última Encomenda
- Dias de Não Utilização - Nº de Dias após a Data da Última Utilização

Sistematização	
Da base lógica resulta que os componentes criados no sistema são:	Notificação

Outros componentes criados não identificados no domínio:	RegistarNotificacaoUI

Diagrama de Sequência

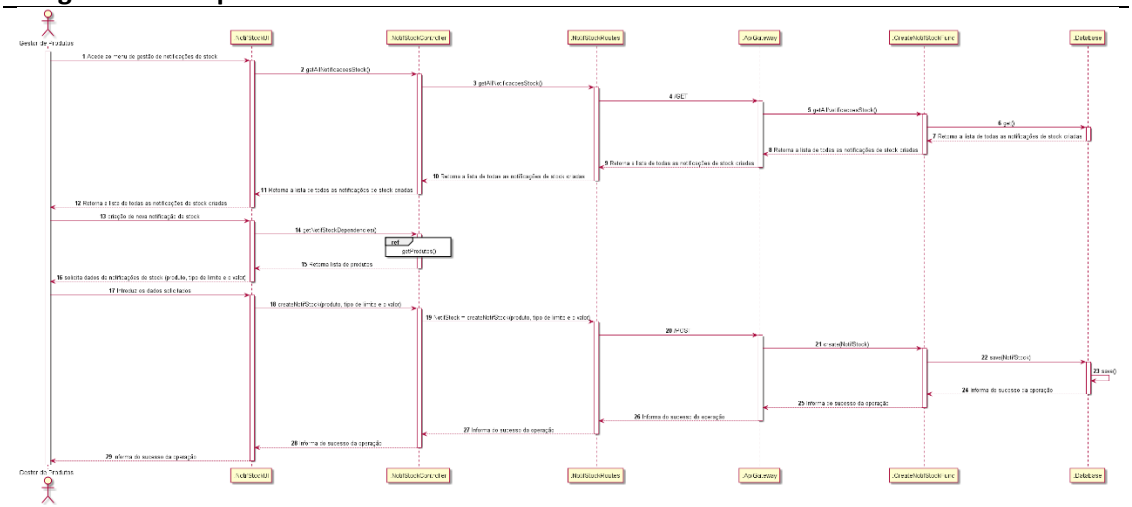


Figura 3636 – Diagrama de sequência do UC15

O diagrama de sequência apresenta na Figura 3636, é composto por uma ação não especificada, nomeadamente a obtenção de todos os produtos, onde a mesma é apresentada num diagrama independente no Anexo F .

4.1.3 Requisitos não funcionais

A implementação de todos os módulos exigidos pelo cliente é fulcral para o êxito de um projeto, no entanto, só isso não garante o seu sucesso. Os requisitos não funcionais complementam o sucesso na disponibilização dos serviços. Estes focam-se na entrega com qualidade, por outras palavras, abordam as restrições dos requisitos funcionais de um sistema. (Phelps, 2021)

4.1.3.1 Usabilidade

A usabilidade trata-se do aumento da eficácia e eficiência do desempenho do cliente na execução dos seus objetivos. Tem grande foco nos aspetos de visualização e apresentação da interface do utilizador. No âmbito do sistema de gestão os requisitos associados a esta categoria são:

- Interface do utilizador intuitiva e inteligível, sem ser sobrecarregada de elementos ou funcionalidades e modulada por níveis;
- Todas as operações e chamadas de serviços com menos de 2 minutos de latência e sem distúrbios;
- Interface gráfica de aprendizagem rápida e fácil, de maneira que não seja necessário a repetição de uma operação constantemente para que o utilizador se recorde da sua funcionalidade.

4.1.3.2 Fiabilidade

A fiabilidade corresponde à disponibilidade e capacidade de uma aplicação executar os seus serviços sem que ocorram erros, ou caso ocorram que estejam devidamente controlados. No contexto do projeto foram aplicados os seguintes requisitos:

- A falha num componente não deve afetar os restantes, garantindo a disponibilidade da aplicação;
- A disponibilidade do servidor deverá ser a tempo inteiro;
- O servidor deverá ter cópias de *backup* diárias num local diferente de onde se encontra a aplicação;
- A indisponibilidade do servidor não deverá provocar perda de informação no sistema.

4.1.3.3 Desempenho

O desempenho visa estipular se um determinado software tem as competências de performance requeridas em termos de escalabilidade e responsividade, ou seja, qualifica o potencial de transferência de informação. No seguimento do projeto temos os seguintes requisitos:

- Independentemente da carga de utilização deverá ser mantida a responsividade e celeridade do software;

- O sistema deve ser altamente escalável, possibilitando a adição de novas funcionalidades sem grandes custos.

4.1.3.4 Suportabilidade

A suportabilidade envolve os mecanismos imprescindíveis para o bom funcionamento de uma aplicação, como manutenibilidade, configurabilidade e escalabilidade. Nesta categoria foram identificados os subsequentes requisitos:

- Armazenamento de *logs* informativos sobre o estado dos serviços executados para o auxílio na busca de anomalias no projeto;

4.1.3.5 Restrições de Desenho

As restrições de desenho têm como objetivo limitar a arquitetura do sistema através do uso de padrões de software, para que este seja mais conciso, dinâmico e acessível. No software têxtil estão expostas as restrições enumeradas:

- *Observer Pattern* - Num software é comum o despoletar de eventos, como um clique num botão. Estes são tratados por funções *event handler*, que administram os eventos adquirindo os seus detalhes e atuando de acordo com o recebido;

- *Strategy Pattern* - O código de uma aplicação deve ser dinâmico para possibilitar a satisfação de diversos clientes. Esta dinamicidade é alcançável através da implementação de um algoritmo que varia dependendo das instruções do utilizador;

- *Template Method Pattern* - Tal como o padrão anterior o seu foco é tornar um software dinâmico. Esta restrição manipula partes do algoritmo segundo as orientações do utilizador, ao contrário do *Strategy Pattern* que altera o algoritmo por completo;

- *Controlled Component Pattern* - Um componente não controlado manipula o seu estado internamente, já o estado de um componente controlado é influenciado pelo pai, recebendo por parâmetro valores e retornando eventos ou notificações;

- *API Gateway Pattern* - Consiste na abstração da camada do cliente da camada de serviços através de um intermediário responsável por monitorizar os pedidos feitos entre essas camadas. Providencia o desacoplamento de ciclos de versão, aumento da interoperabilidade e escalabilidade de um software;

- *Model-View-Controller Pattern* (MVC) - Representa um padrão de desenho que desacopla os dados (*models*), da interface gráfica (*views*), da lógica de negócio (*controllers*);

- *Active Record Pattern* - Corresponde a um padrão próprio a projetos que comuniquem com bases de dados relacionais. A sua especialidade é traduzir dados entre sistemas incompatíveis.

4.1.3.6 Restrições de Implementação

As restrições de implementação dizem respeito ao conjunto de normas que o desenvolvimento de uma aplicação deve seguir, o que acaba por afetar indiretamente o desenho da sua arquitetura. Na conjuntura do sistema de gestão temos como restrições de implementação:

1. A linguagem de programação a ser utilizada na implementação da aplicação é JavaScript;

2. Devido à complexidade e interligação de entidades do sistema de gestão o tipo de base de dados a utilizar tem de ser relacional;

3. A comunicação entre base de dados e o programa deve ser assíncrona.

4.2 Análise e Conceção da Solução

Este capítulo especifica a arquitetura global da solução, oferecendo uma visão da solução idealizada previa, antes de avançar com a sua implementação, referenciando os requisitos e conceitos inerentes ao projeto.

A secção inicia-se com a apresentação do modelo domínio, onde são detalhados os atributos de cada um dos elementos, e as suas respetivas relações.

Segue-se, com a demonstração das diferentes camadas da arquitetura do sistema, especificando a intervenção de cada uma delas.

4.2.1 *Modelo de Domínio*

O modelo de domínio consiste numa representação visual dos conceitos inerentes ao sistema em desenvolvimento detalhando as relações entre cada um deles. Derivado da compreensão dos requisitos de nível de sistema, a definição de uma entidade domínio, bem como as suas ligações fornece uma base eficaz para a compreensão e ajuda os profissionais a projetar sistemas para manutenção, testabilidade e desenvolvimento incremental (Leffingwell, 2021).

À medida que ocorrem alterações no sistema, torna-se vital atualizar o modelo de domínio para manter um entendimento contínuo do sistema, a fim de controlar a complexidade associada aos sistemas de software.

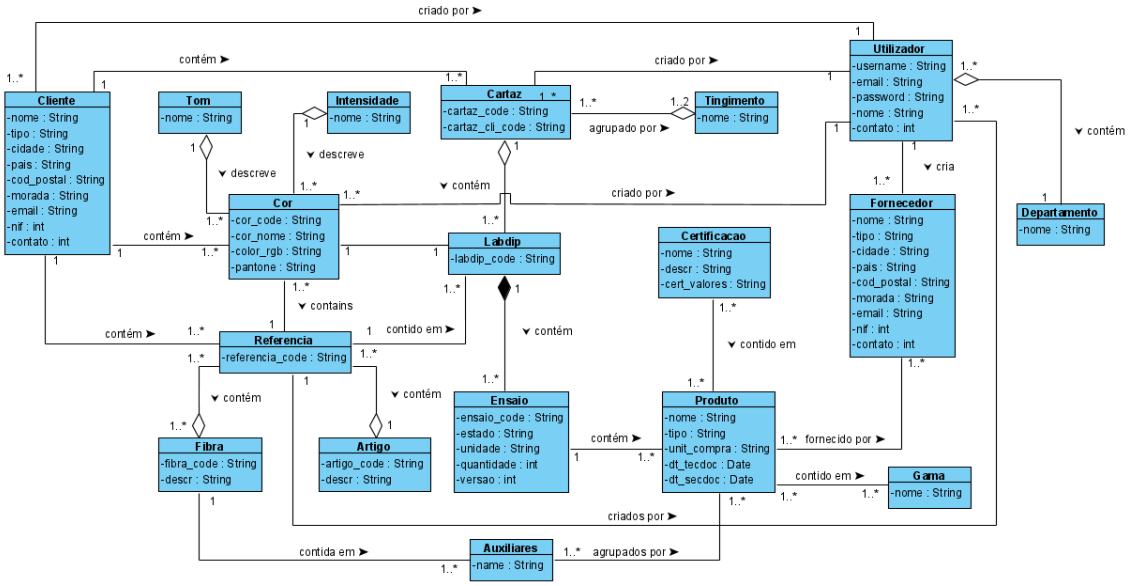


Figura 3737 – Modelo domínio do sistema

De acordo com o modelo domínio apresentado na Figura 3737, um **cliente** pode disponibilizar vários **cartazes**, da mesma forma que pode conter inúmeras cores e referências, onde as mesmas estão interligadas.

No seguimento do **cartaz**, são criados os ***lab-dips***, os quais estabelecem uma relação de composição com **ensaios**, em que a inexistência de um ***lab-dip***, impossibilita a criação de **ensaios**, porque estes representam tentativas de reprodução de uma determinada **cor**.

A **referência** compõe-se por um único **artigo**, e por uma **fibra**, ou uma mistura de várias. O mesmo tipo de relação ocorre na **cor**, onde a mesma se define por um único **tom** e **intensidade**. As agregações foram criadas essencialmente para agrupar conceitos que tipicamente constituem um elevado número de registos, e desta forma permite facilitar consultas e pesquisas posteriormente.

No que toca ao **produto**, este é disponibilizado por um **fornecedor**, onde seguidamente são associados diversos **certificados** e **grupos**, que servem para filtrar a vasta gama de **produtos** habitualmente existente, de maneira a selecionar os que cumprem com determinadas normas exigidas pelo **cliente**.

O **utilizador** é responsável por criar os diversos elementos do sistema, sendo que depende das permissões que lhe são atribuídas. Os acessos de cada um são geridos consoante o **departamento** ao qual está atribuído.

4.2.2 *Diagrama de Camadas*

O presente diagrama responsabiliza-se por demonstrar as diversas partes que constituem um sistema, compreensível do ponto de vista do *end-user*. Para além de exibir a segregação da aplicação em vários módulos, também exemplifica as suas relações.

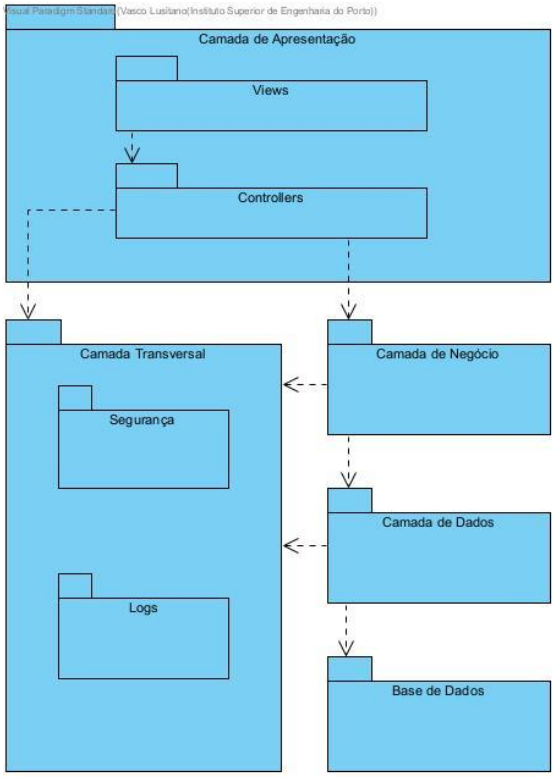


Figura 38 38 – Diagrama de camadas

No enquadramento do projeto existem quatro camadas, a de apresentação diretamente ligada ao cliente, a de negócio responsável pelas regras de negócio e lógica do sistema, a camada de dados composta pelos pedidos à base de dados e a camada transversal que contem medidas de segurança e rastreamento de serviços invocados ou erros do sistema.

4.2.2.1 *Camada de Apresentação*

Esta camada tem como função apresentar ao utilizador o conjunto de dados manipulados nas outras secções. A descodificação dos dados em conteúdo compreensível ao cliente é realizada nesta camada, como também a sua apresentação na interface do utilizador.

No contexto do sistema de gestão esta camada está subdividida numa estrutura MVC, ou seja, é composta por views, controllers e models, mais especificados no diagrama de implementação (Gregg, 2006). Ainda contem um módulo de services que organiza as chamadas a API's externas.

4.2.2.2 *Camada de Negócio*

A camada em apreço contém a lógica de negócio, por outras palavras estão nela presentes as principais funções e orquestrações de dados, e pode ser dividida em múltiplos servidores, cada um com a sua própria responsabilidade.

Na conformidade do software de laboratório estão nesta secção presentes componentes já abordados na camada de apresentação, os *models*, *controllers* e *services* (TechTarget, 2021). Além destes existe o pacote *route handler*, que recebe pedidos HTTP de aplicações externas.

4.2.2.3 *Camada de Dados*

Os acessos e manipulação de informação proveniente de uma base de dados é responsabilidade da camada em foco, isto inclui operações de leitura, edição inserção e eliminação de dados (TechTarget, 2021).

No cenário do projeto têxtil, o acesso à base de dados é exercido a partir de um ORM, *Object Relational Mapping*, ou seja, descarta-se a necessidade e complexidade de utilizar comandos *query* através de ferramentas SQL reduzindo em grande quantidade o número de linhas de código. Outra vantagem deste padrão é o facto de não ser obrigatório o uso de expressões específicas de uma dada *framework* de base de dados, facilitando uma futura migração de *MySQL* para qualquer outro software.

4.2.2.4 Camada Transversal

O objetivo da camada em evidência é prestar auxílio a múltiplas camadas, isto é, disponibiliza componentes comuns entre elas, melhorando a adaptabilidade do software. No âmbito do projeto as subcamadas dentro da camada transversal são (Mao, 2010):

- Segurança - fundamental em qualquer software que envolva utilizadores registados e dados pessoais, esta subcamada compreende a autenticação e autorização de utilizadores, além da encriptação de dados;
- *Logs* - formidável para o rastreamento do estado da aplicação, facilitando na busca de *bugs* no sistema ou simplesmente validação de dados informativos;

A autenticação é o processo que se encarrega de validar se um utilizador está registado na aplicação, já a autorização é o processo que se encarrega de validar se o utilizador já registado tem permissões para uma dada funcionalidade. No caso do sistema têxtil são aplicadas as seguintes validações de segurança:

- *Token* - Biblioteca digital encarregada de fazer uma autenticação dupla (autenticação e autorização);
- AWS *Role and Policy* – cada identidade ou recurso têm as suas dadas permissões nos serviços AWS.

Um ficheiro de *logs* contem informação sobre os eventos despoletados no sistema, estes podem ser informativos, de erro ou de aviso. Nas circunstâncias do projeto de gestão os *logs* são gerados a partir de duas formas diferentes, particularmente:

- *React-native-logs* - biblioteca própria da linguagem *react* que regista os *logs* do segmento de interface gráfica da aplicação, os quais são separados por níveis;
- AWS *CloudWatch* - serviço Amazon que fornece dados e *logs* de vários graus, que ficam alistados na consola da AWS;

4.2.3 Diagrama de Implementação

O diagrama em análise apresenta a arquitetura de implementação de um software, onde estão expostos os seus subsistemas e as suas dependências para servirem de guia aos programadores. O diagrama de implementação do projeto têxtil está representado na Figura 39 39 e posteriormente a definição de cada subcomponente.

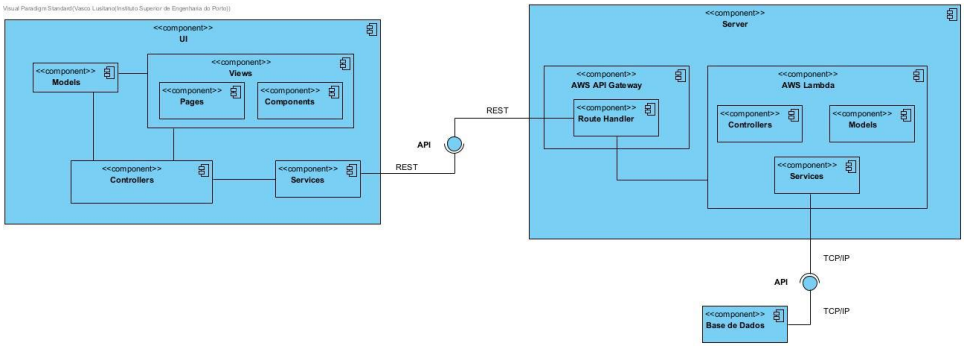


Figura 39 39 – Diagrama de Implementação

A Figura 39 39 apresenta a arquitetura adotada para implementação, e a mesma está composta pelos seguintes elementos:

- ***View*** *c*orresponde ao componente que insere e traduz os dados num formulário específico que é empacotado e apresentado no browser, de modo a satisfazer a gráfica do cliente. É composto por ***Components***, excertos de código HTML que constituem uma página, e ***Pages***, agregados de components que constroem uma página complexa;

- ***Model***: Representa os dados manipulados nos *controllers* e demonstrados nas *views*, são caracterizações das entidades do domínio que operam desde a camada de dados (base de dados) até à camada de apresentação (interface do utilizador);

- ***Controller***: Responsável por tratar os pedidos do *browser* ou de serviços, retornando uma determinada *view*, dados, ou redirecionando para outro *controller*, além de manipular os *models*;

- ***Services***: Diz respeito à divisão encarregada por estabelecer uma conexão a API's externas, onde são retornados ou enviados dados para sistemas exteriores dos quais depende ou dependem do software em questão;

- ***Route Handler*** - A sua função é estar preparado a receber pedidos HTTP a partir de um determinado endereço web e método específico (*GET, POST, PUT, DELETE, PATCH, HEAD* ou *ANY*) para enviar resposta com dados segundo os parâmetros que recebe;

4.2.4 Diagrama de Implantação

O diagrama em questão tem como objetivo orientar a equipa de engenharia de sistemas, explicitando os ambientes nos quais os componentes do software serão implantados. Por acréscimo ainda são realçados os tipos de conexões entre esses mesmos componentes. A Figura 4040 apresenta o diagrama de implantação do projeto de gestão.

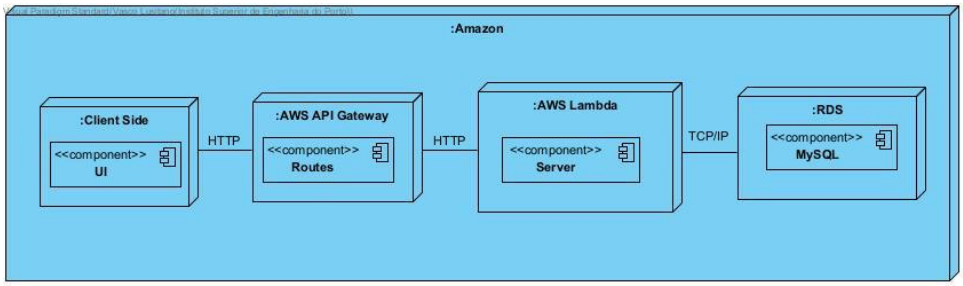


Figura 4040 – Diagrama de Implantação

Na visão do sistema têxtil todos os componentes estão implantados no ambiente da Amazon, dos quais a camada de servidor, base de dados e routes usam serviços próprios da AWS, nomeadamente:

- AWS *API Gateway* - serviço da Amazon que proporciona a criação, manutenção e consulta de API's de forma segura, neste caso através de comunicação HTTP, mais especificamente REST;
- AWS *Lambda* - serviço da Amazon que possibilita a escrita de código sem o problema de manutenção de servidores, evitando problemas de administração de recursos e disponibilizando alta disponibilidade;
- RDS - serviço da Amazon projetado para facilitar as configurações e operações de base de dados, estando disponível para as mais conhecidas e utilizadas bases de dados da atualidade.

Ainda é possível visualizar o tipo de comunicação entre as diferentes secções da aplicação, sendo a ligação entre cliente, servidor e routes através de HTTP e a conexão entre servidor e base de dados por ORM, por outras palavras traduz dados entre sistemas com tipos de dados diferentes.

5 Implementação

O presente capítulo visa descrever todos os detalhes relacionados com o enquadramento e implementação da solução preconizada anteriormente, onde é apresentando de forma exaustiva, todos os elementos que constituem a implementação da solução, e as linguagens utilizadas em cada uma das camadas.

Adicionalmente, abordam-se os testes realizados, às diversas componentes do sistema, recorrendo a testes unitário, integração, sistema e aceitação.

5.1 *Stack* Tecnológico

A seleção da *stack* tecnológica consiste na combinação de tecnologias selecionadas para construir e executar um projeto. Normalmente, esta *stack* agrega as linguagens de programação, base de dados, ferramentas de *front-end*, ferramentas de *back-end*, e pontes de conexão entre ambos através de API's.

Deste modo, a *stack* oferece uma visão geral das linguagens de programação inerentes ao sistema, identificando os principais componentes estruturais do sistema, e as suas respetivas ligações, desde a interação do utilizador, até ao base de dados do sistema (Booth, 2021).

Adicionalmente, como a maioria das linguagens de codificação tem atributos e limitações de desempenho conhecidos, a *stack* tecnológica sugere os pontos fortes e fracos da aplicação, expondo assim possíveis fraquezas a colmatar.

Por conseguinte, a importância associada à construção da *stack* considera-se magnânima, caso contrário, alterações posteriores ou reconstruções no sistema, são muito difíceis de se

concretizar com sucesso, e implicam elevados custos, e por isso, a manutenção representa sempre a maior despesa de um orçamentado de desenvolvimento de uma aplicação.

A Figura 4141, apresenta a estrutura tecnológica do sistema projetado no presente relatório, detalhando a tecnologias utilizadas em cada camada, bem como a comunicação existente entre cada um dos níveis.

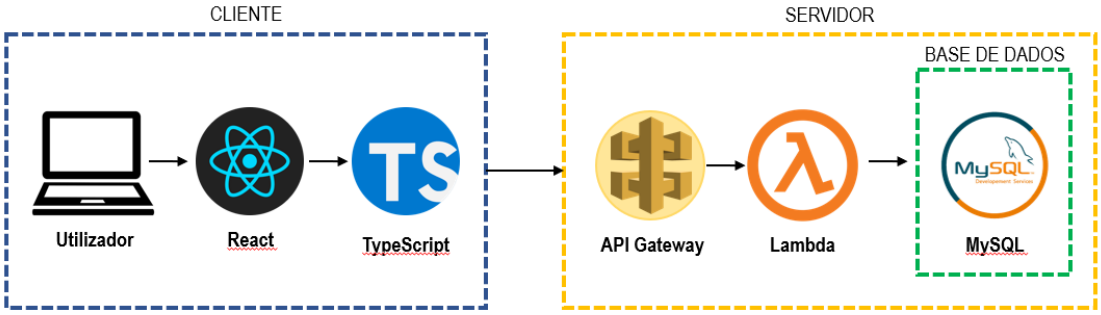


Figura 4141 – *Stack* tecnológica do sistema

Atualmente, os sistemas são separados por dois componentes, como demonstrado na Figura 4141, onde o lado do cliente se concentra em renderização da página, enquanto o servidor concentra-se na lógica e execução de processos e interação com a base de dados que ocorrem em segundo plano, maioritariamente impercetíveis para o utilizador.

A prova de conceito envolve a interação entre estes dois componentes do sistema, o cliente e o servidor, sendo a *API Gateway* responsável por direcionar as comunicações criadas pela interface disponibilizada ao cliente (Wong, 2020).

Adicionalmente, em ambos os componentes mencionados, constam linguagens de programação e *frameworks* que auxiliam o desenvolvimento. Quanto ao lado do cliente otimizam a experiência do utilizador, no entanto, no que toca ao servidor, através de um conjunto de ferramentas distintas potencializam os serviços e as interações com os dados.

Em suma, tendo em conta as tecnologias emergentes disponíveis, e as ferramentas de desenvolvimentos, adotou-se a combinação mais benéfica para o enquadramento em que o projeto se insere:

Tabela 18 – Tecnologias utilizadas e respetiva área

Área		**Tecnologias**
Linguagens	Cliente	*React, TypeScript,* HTML / CSS
	Servidor	*NodeJS*
	Base de Dados	*MySQL*
Frameworks		*Material-UI, Toastify, Axios*
Editor de código		*Visual Studio Code*

5.1.1 Cliente

O aprofundamento da aprendizagem, e o desenvolvimento de aplicações corporativas cada vez mais populares, criou uma tendência de incorporação de diversas *frameworks*, a fim de reduzir as limitações das linguagens de programação.

Deste forma, esta combinação de pequenas ferramentas, aceleram o processo de desenvolvimento, complementam as funcionalidades inerentes ao produto, e diminuem as dificuldades sentidas na manutenção do sistema.

5.1.1.1 Estrutura do Cliente

A estrutura do projeto, consiste na disposição dos diferentes diretórios que compõe a implementação do lado do cliente. Apesar de ser um aspeto que varia consoante os gostos pessoais, e o âmbito da aplicação, trata-se de um ponto muito relevante no projeto.

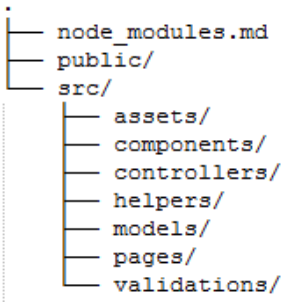


Figura 4242 – Estruturação dos diretórios de código-fonte do projeto

A Figura 4242 evidencia a distribuição adotada para o sistema correspondente ao do presente relatório. Desta forma, as pastas categorizam de uma forma genérica os diferentes componentes criados no sistema.

Tabela 19 – Ficheiros contidos nos diretórios do projeto

Diretório	Ficheiros
node_modules	Guarda todas as dependências necessárias para a execução do projeto, onde através do comando `npm install`, descarrega as dependências para um diretório local do dispositivo em uso.
public	Contém o arquivo HTML renderizado no momento da execução do projeto.
src/*Assets*	Disponibiliza recursos necessários pela aplicação em *runtime*, ou seja, no decorrer da execução, como por exemplo, imagens e vídeos.
src/Components	Agrega os diversos componentes genéricos partilhados entre as várias páginas da aplicação, desde tabelas, formulários, entre outros.
src/Controllers	Estabelece a ponte entre a ação despoletada pelo utilizador e a função que tratará de interagir com o servidor.
src/Helpers	Funções genéricas utilizadas múltiplas vezes ao longo do código que evitam a duplicação de conteúdo, como por exemplo, iteradores.
src/Models	Define as interfaces para cada objeto utilizado no sistema, detalhando os atributos que compõem cada um.

src/Pages	Estruturação das páginas visíveis para o utilizador, importando os componentes e associando a informação vinda do serviço.
src/Services	Chamadas API que interagem com o servidor, e validam a resposta proveniente após a chamada.
src/Validations	Guarda todas as validações dos diferentes componentes e formulários no sistema.

Legenda: - Diretórios comuns; - Interface do Utilizador; - Serviços Web

Na Tabela 19 – Ficheiros contidos nos diretórios do projeto apresenta os diretórios que compõem a aplicação, dividindo-os em três grupos distintos. Os diretórios comuns, como o próprio nome indica, caracterizam habitualmente qualquer projeto *React*.

No que toca às categorias mencionadas, os pastas direcionadas à implementação da interface de utilizador e dos serviços web, são dependentes entre si, e por isso, para um *client side* estável, depende do funcionamento ótimo de ambas as partes.

Interface do Utilizador

"A experiência do utilizador abrange todos os aspetos da interação do beneficiário final com a empresa, os seus serviços e produtos." (Norman, 1990). O desenvolvimento desta camada projeta a *User Interface* (UI), que se responsabiliza pelas interações entre a aplicação e o utilizador.

O design da interface do utilizador concentra-se em prever as necessidades do mesmo, garantindo que a disposição dos seus elementos seja de fácil compreensão, e o acesso às suas ações seja igualmente intuitivo.

Deste modo, definiram-se requisitos associado à interface a desenvolver para a aplicação, tendo em conta os requisitos:

1. Desenvolvimento da interface *web* através da utilização das linguagens comuns HTML e CSS:

    a. HTML: A sigla significa *Hypertext Markup Language*, e trata-se da linguagem padrão utilizada para o desenvolvimento *web*. O HTML recorre a *tags* que estruturam a página que posteriormente renderiza para o utilizador.

    b. CSS: Traduz-se por *Cascading Style Sheets*, e corresponde a uma linguagem de *design* utilizada com o propósito de tornar os diferentes componentes esteticamente mais apelativos e atraentes.

2. A aplicação *web* deve apresentar um elevado grau de customização, permitindo ao utilizador editar a disposição dos elementos, bem como ordenar e agrupar os dados apresentados em tabelas de consulta.

3. Homogeneidade nos diversos elementos da interface, nomeadamente, componentes de entrada (botões, campos de texto, campos de seleção), navegação (campos de pesquisa, paginação, ícones), e informativos (notificações).

A criação de uma interface intuitiva para o utilizador, tem como grande desafio dar resposta aos requisitos de usabilidade e performance da aplicação. Com o propósito de cumprir com as exigências definidas, o procedimento iniciou-se pela pesquisa de *layouts* que se enquadravam ao contexto do projeto, e com requisitos mencionados anteriormente.

Desta forma, através da pesquisa realizada, conclui-se que o recurso ao *AdminLTE*, que corresponde a uma ferramenta *open-source* direcionada a sistemas de gestão preenche todos os requisitos levantados.

A *AdminLTE* disponibiliza os elementos *web* mais comuns e indispensáveis para o desenvolvimento de um projeto. Esses componentes possibilitam a aceleração da construção da interface do utilizador, estruturando uma página *home*.

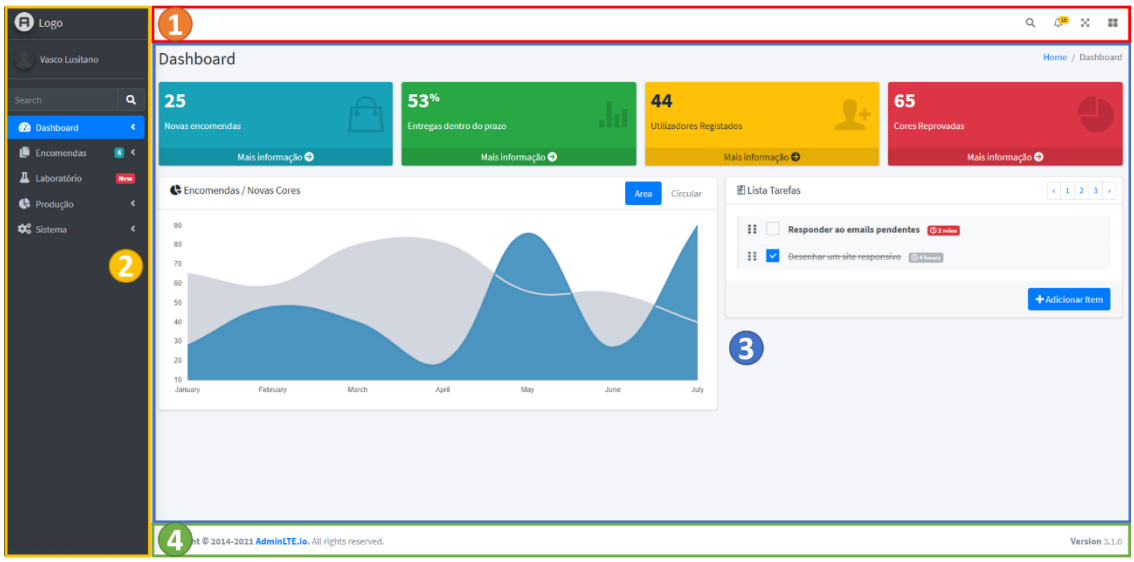


Figura 4343 – Dashboard da página *home*

A utilização de *dashboards* representa um elemento fulcral no momento da demonstração de dados ao utilizador. O facto de fornecerem uma visão facilitada dos principais indicadores de desempenho (KPI's) relevantes, possibilitam uma interação rápida com cada um dos elementos constituintes desse painel.

A Figura 4343 criou-se com base num rascunho elaborada previamente (anexo rascunho), em se destacam as quatro principais regiões que compõem a página:

- Cabeçalho – Constituído pela funcionalidade de pesquisa, acesso a notificações, e ação de expandir e esconder a barra de navegação;

- Barra de navegação – Região composta pelo logotipo da empresa, o nome do utilizador, os menus disponíveis, e uma barra de pesquisa;

- Conteúdo principal – Incorpora o conteúdo a exibir ao utilizador, variando consoante a interação e as ações solicitadas pelo mesmo;

- Rodapé – Semelhante ao cabeçalho no que toca à área da página que ocupa, no entanto, contém informações distintas, como direitos autorais, declarações de privacidade, e outras hiperligações.

Responsividade

A incorporação de um *design* responsivo, trata-se de uma abordagem de desenvolvimento *web* que cria alteras dinâmicas nos componentes da página, dependendo do tamanho da tela, e da orientação do dispositivo onde está a ser visualizado (Schade, 2014).

Desta forma, tendo em conta o setor onde se insere a aplicação, torna-se vital assegurar a responsividade da interface do utilizador, devido à variedade de terminais que acedem à aplicação, e às distintas dimensões do ecrã.

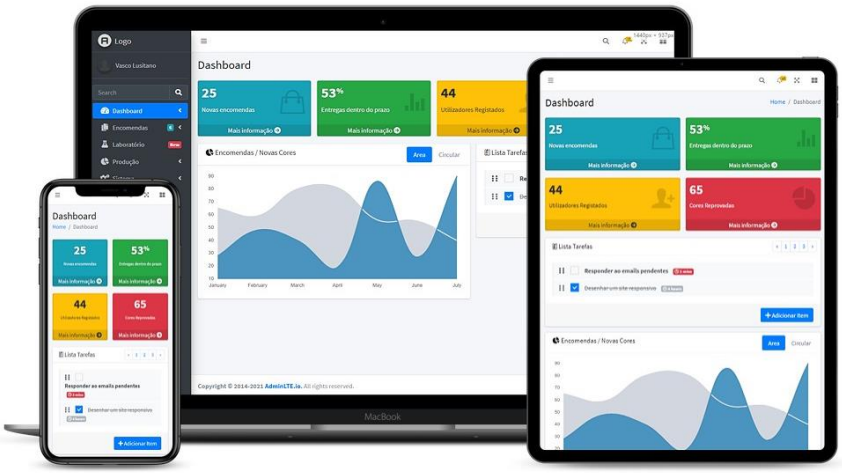


Figura 4444 – Responsividade do *dashboard* inicial da aplicação

Componentes

Na linguagem de programação *React*, os componentes são fragmentos de código independentes e reutilizáveis, ou seja, têm a mesma finalidade que funções de JavaScript, mas funcionam isoladamente e retornam elementos HTML.

Autenticação

O processo de autenticação permite verificar a identificação e autenticidade do utilizador no sistema. O controlo de acesso estabelece-se sobre a forma de credenciais: nome de utilizador e password.

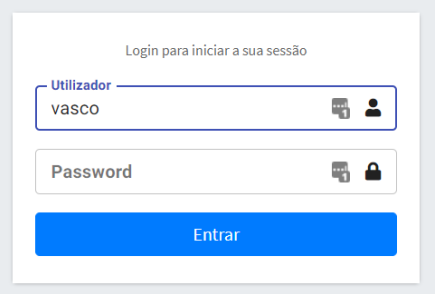


Figura 4545 – Interface na autenticação do utilizador

A submissão do formulário de autenticação, despoleta o evento `onSubmit` que trata de encaminhar para a ação que valida e envia os dados para o servidor.

Neste caso em particular, na autenticação recorre-se ao JSON *Web Token* (JWT) que após uma validação bem-sucedida das credenciais, cada solicitação subsequente incluirá uma "assinatura" permitindo validar os recursos e privilégios de cada utilizador.

```
<div className="login-box">
    <LoginLogo link="" label=""/>
    <div className="card">
        <div className="card-body login-card-body">
            <p className="login-box-msg"><small>Login para iniciar a sua sessão</small></p>
            <FormProvider {...formMethods}>
                <form onSubmit={formMethods.handleSubmit(onLogin)}>
                    <Input name="username" label="Utilizador" icon="user" type="text"/>
                    <Input name="password" label="Password" icon="lock" type="password"/>
                    <Button label="Entrar" class="primary"/>
                </form>
            </FormProvider>
            <p className="mb-1"></p>
        </div>
    </div>
</div>
```

Figura 46 46 – Fragmento código da autenticação do utilizador

Operações CRUD

A sigla CRUD advém do mundo da engenharia de *software*, onde se refere às quatro funções consideradas necessárias para implementar uma aplicação que contenha persistência nos dados que armazena: criar, consultar, atualizar e excluir.

Com o propósito de facilitar a realização das operações mencionadas, recorreu-se ao *Material-UI*, que se trata de uma biblioteca que permite importar e usar diferentes componentes reutilizáveis, de maneira a criar uma interface de utilizador apelativa.

Desta forma, as operações de criação, consulta, edição e eliminação de dados ocorrem na tabela que lista os dados do sistema.

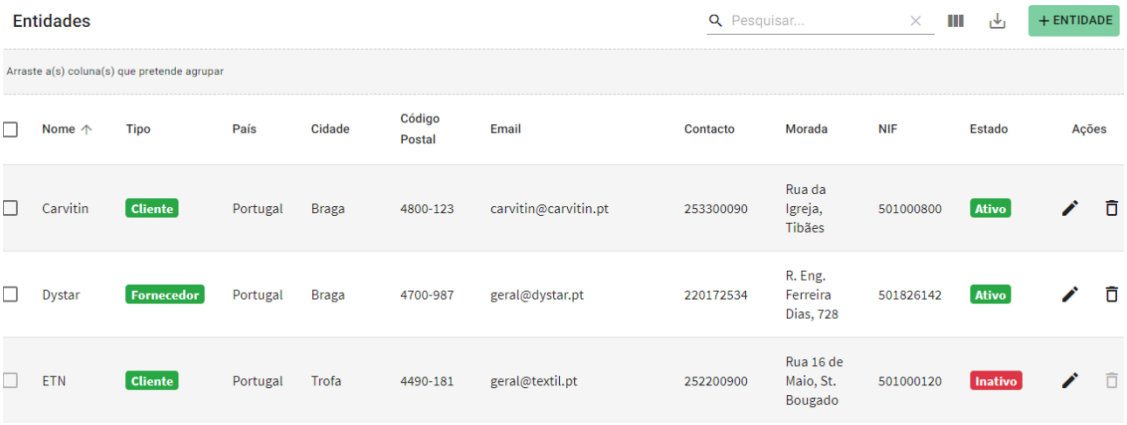


Figura 47 47 – Listagem de entidades do sistema

A Figura 47 47 exibe a lista de entidades guardados na base de dados do sistema, no caso de pretender criar uma nova entidade, a tabela dinamicamente acrescenta uma linha vazia assim que entidade clica em + ENTIDADE

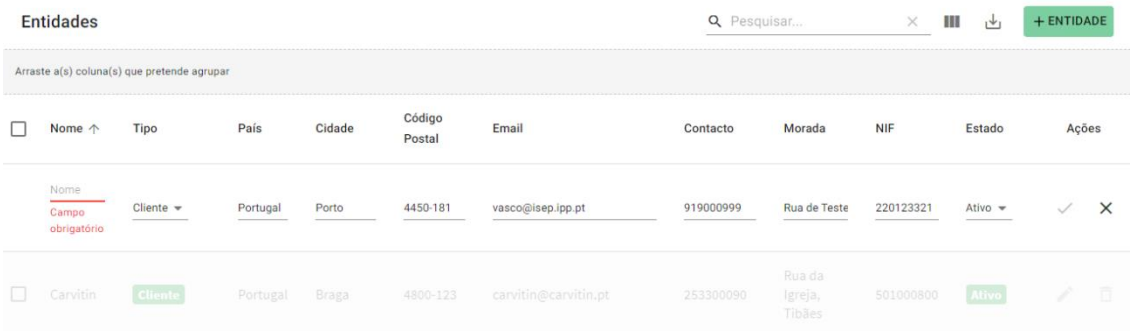


Figura 4848 – Criar nova entidade no sistema

A Figura 4848 evidencia o procedimento de criação da entidade, onde a inserção dos dados e validação dos mesmos decorre na própria tabela, sem recurso a diálogos ou elementos que sobreponham a interface. Por outro lado, caso a operação pretendida seja de edição da informação de uma entidade registada.

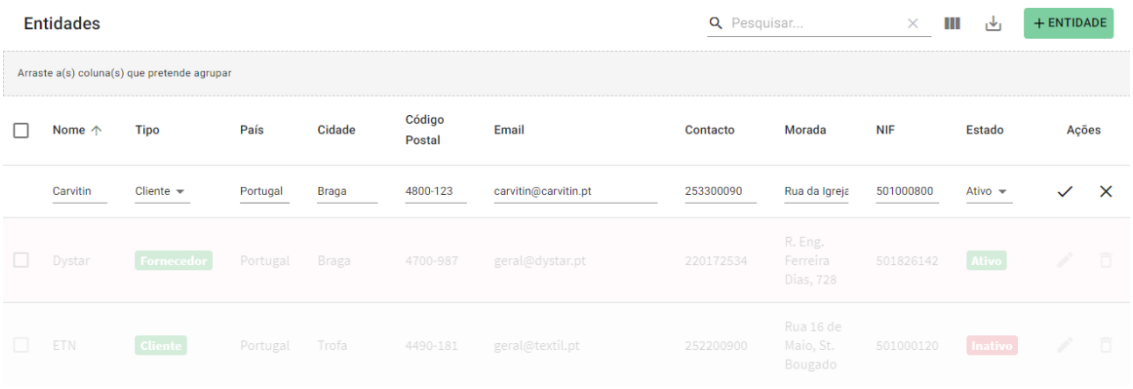


Figura 4949 – Editar dados de entidade

À semelhança do fluxo da criação de uma entidade, a operação de edição dos dados do mesmo, procede-se de forma equivalente à demonstrada na Figura 4949, mas nesta operação, os campos são por definição preenchidos com os dados existentes.

Adicionalmente, há a hipótese de eliminar a entidade, que no contexto da gestão de entidades, ocorre apenas uma inativação, impedindo-o assim de se autenticar. Denomina-se de *soft delete* quando não ocorre a eliminação total do registo da entidade na base de dados, com o propósito de preservar o histórico de dados criados e alterados no sistema pela entidade que se pretende eliminar.

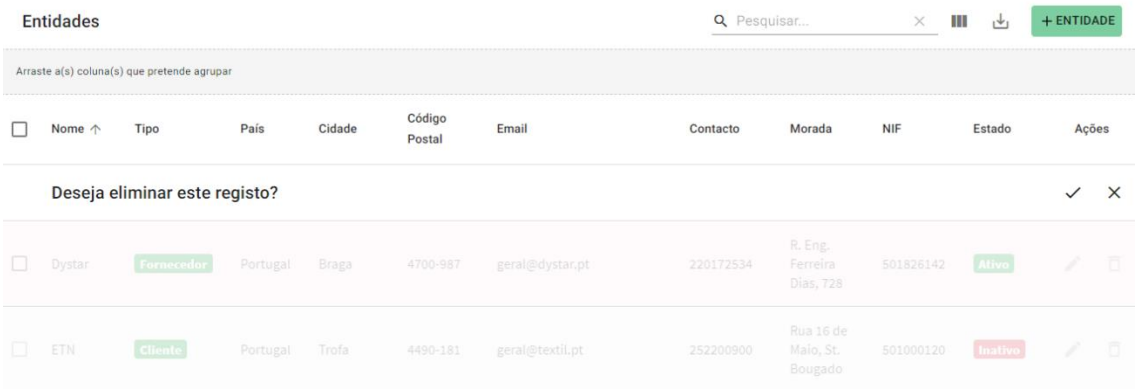


Figura 5050 – Eliminar entidade registada

Como foi mencionado anteriormente, para a construção das tabelas criou-se um componente genérico, com base na utilização dos recursos da Material-UI, de maneira que este possa ser utilizado em contextos diferentes, mas que comportamento seja uniforme entre as diferentes páginas.

```
<MaterialTable
    localization={props.localization ? { ...Localization, ...props.localization } : Localization}
    title={props.title}
    columns={props.columns}
    data={props.list}
    options={{ ...options, ...props.options }}
    editable={props.editable}
    icons={props.icons}
    actions={props.actions ?? []}
    />
```

Figura 51 – Componente reutilizável para a criação de tabela CRUD

A Figura 51 demonstra o componente criado para a listagem de dados e realização de operações CRUD, e este pode ser reutilizado transversalmente ao longo de várias páginas da aplicação. O *React* permite passar informações a um componente através da utilização do *props*. Portanto, os componentes que implementam esta tabela enviam as propriedades exigidas pelo mesmo.

```
<GeneralElem.Table
    title="Entidades"    // Título da Tabela
    columns={columns}    // Array com as diferentes colunas que compõem a tabela
    list={entities}      // Lista de elementos a apresentar
    options={options}    // Opções de personalização (ordenação, paginação, ...)
    editable={{          // Definir comportamentos das Operações CRUD
        isDeletable:(row:any)=>row.state==1,
        onRowAdd:(newRow:any)=>new Promise((resolve, reject)=>{
            resolve(onCreateEditEntity(newRow))
        }),
        onRowUpdate:(newRow:any, oldRow:any)=>new Promise((resolve, reject)=>{
            resolve(onCreateEditEntity(newRow))
        }),
        onRowDelete:(selectedRow : any)=> new Promise((resolve, reject)=>{
            resolve(onInactivateEntity(selectedRow.id))
        }),
    }}
    icons={{             // Personalizar botões / ações do cabeçalho da tabela
        Add: forwardRef((props: any, ref: any) =>
        <Button {...props} ref={ref} style={{background:"#7DCEA0"}}><AddIcon fontSize="small"/>Entidade</Button>)
    }}

/>
```

Figura 52 – Implementação da lista de entidades

A implementação demonstrada na Figura 52, exemplifica a definição dos diferentes atributos que compõem a tabela, e uma visão geral de cada um dos atributos que são passados por parâmetro ao componente genérico.

Desta forma, para aceder à informação enviada através das propriedades enviadas pelo componente, basta colocar após o *props* o nome do atributo definido, como por exemplo, *props.title*.

Na aplicação, as operações mais simples, como a gestão de utilizadores ou entidades, estas operações CRUD realizam-se neste formato da tabela. Por outro lado, as restantes são efetuadas através de formulários.

Formulário

Os formulários são uma parte essencial da interação por parte dos utilizadores com as páginas *web.* Estes estão presentes em diversos contextos, onde a validação dos inseridos pelo utilizador são da responsabilidade do programador.

Nesse sentido, o *React,* disponibiliza uma biblioteca nativa *React-hook-form* corresponde a uma biblioteca que ajuda a validar formulários *React*. Como a biblioteca não possui qualquer tipo de dependências, o peso da importação é mínimo. A sua utilização auxilia o desenvolvimento exigindo menos linhas de código o que posteriormente facilita a leitura da implementação.

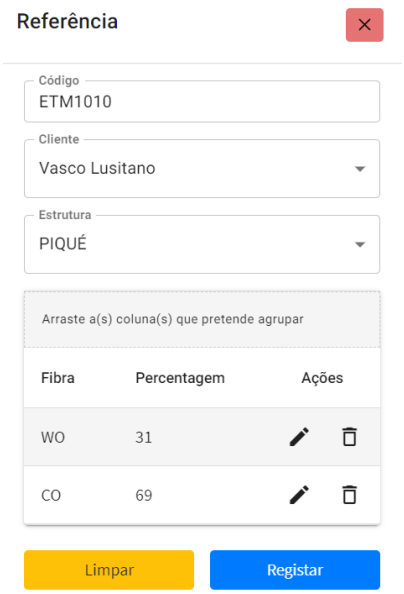


Figura 53 – Formulário de criação de referência

A Figura 53, apresenta o formulário elaborado com o propósito da criação de uma referência no sistema. Apesar de se tratar de um número reduzido de campos a preencher, o formulário exibido constitui-se por diferentes tipos de dados, que resultam em componentes distintos.

Os componentes criados consoante o tipo de dados, foram desenvolvidos com a intervenção do *Material-UI*, seguindo a mesma lógica de implementação que permite que sejam reutilizados em qualquer página.

```
<MuiSelect
    label={props.label}
    name={props.name}
    value={value}
    onChange={onChange}>
    {
        props.options.map(
            (item: any) => (<MenuItem key={item[props.idProp]} value={item[props.idProp]}>{item[props.descrProp]}</MenuItem>)
        )
    }
</MuiSelect>
```

Figura 54 – Implementação do componente de seleção de dados

Alerta

Os alertas exibem mensagens informativas que auxiliam o utilizador na compreensão das diversas interações que ocorrem com a interface. O *Toastity* proporcionou métodos de personalização dos alertas, possibilitando definir *timers*, animações e interações com os próprios *pop-ups* que surgem.

Figura 55 – Notificações de alerta

Em semelhança às bibliotecas apresentadas anteriormente, o *Toastify* define um conjunto de atributos que definem os comportamentos específicos do alerta. Neste caso, o único *props* a ser definido pelo componente que utilizem o *ToastContainer*, definem unicamente o tipo de alerta: sucesso, aviso, erro ou informação.

```
<ToastContainer
  // Configurações transversais a todos os alertas
  position="top-right"      // Posição do alerta
  autoClose={5000}          // Duração da notificação
  hideProgressBar={false}   // Exeibe barra de progresso
  newestOnTop={false}       // Adiciona novos alertas abaixo do anterior
  transition={Slide}        // Animação de entrada
  closeOnClick              // Clicar sob o alerta, esconde-o
  pauseOnFocusLoss          // Duração retoma quando o rato sai da sobreposição o alerta
  draggable                 // Possibilita arrastar o alerta
  pauseOnHover              // Duração definida pausa quando o rato está sobrepondo o alerta
/>;

switch(toastType) {
  case `${process.env.REACT_APP_TOAST_SUCCESS}`:   return toast.success(<div><DoneIcon /> {props.message}</div>, toastConfigs);
  case `${process.env.REACT_APP_TOAST_WARNING}`:   return toast.warning(<div><WarningIcon /> {props.message}</div>, toastConfigs);
  case `${process.env.REACT_APP_TOAST_INFO}`:   return toast.info(<div><InfoIcon /> {props.message}</div>, toastConfigs);
  case `${process.env.REACT_APP_TOAST_ERROR}`:   return toast.error(<div><ErrorOutlineIcon /> {props.message}</div>, toastConfigs);

  default:    return toast(<div><ErrorOutlineIcon /> {props.message}</div>, toastConfigs);
}
```

Figura 56 – Implementação da notificação de alerta

Models

O diretório dos *models* contém as interfaces estruturadas em *TypeScript* que definem uma estrutura na aplicação. As classes derivadas de uma determinada interface devem seguir a sintaxe fornecida pela mesma.

```
export default interface Fiber {
    id: number;
    color_code: string;
    color_name: string;
    color_rgb: string;
    pantone: string;
    entity_id: Entity;
    tone: Tone;
    intensity: Intensity;
    hidden: number;
}
```

Figura 57 – Estrutura de Interface em TypeScript

No exemplo demonstrado na Figura 57, a interface *Fiber* inclui nove propriedades. Para cada um dos atributos, detalha-se o tipo de dados, podendo este ser uma outra interface igualmente definida, uma *string*, um número, entre outros.

Assim, qualquer invocação da interface *Fiber* tem a obrigatoriedade de definir as nove propriedades que a definem, caso contrário ocorrerá um erro na compilação do projeto devido à insuficiência de dados.

Controller

Um *controller* trata os pedidos provenientes das ações do utilizador. Este encarrega-se de encaminhar ou redirecionar para uma determinada view, ou acesso a dados do sistema, a fim de dar resposta ao fluxo das funcionalidades.

```
import { GetEntitiesRoute, InactivateEntityRoute, CreateEntityRoute, EditEntityRoute } from "../../Apis/System/EntityRoutes";
import Entity from "../../Models/Entity";

// Encaminha para o serviço que retorna todas as entidades
export async function getEntities(): Promise<{ success: boolean, message?: string, entities?: Entity[] }> {
    return await GetEntitiesRoute();
}

// Encaminha para o serviço que inativa uma entidade
export async function inactivateEntity(entity_id: string): Promise <{ success: boolean, message?: string }> {
    return await InactivateEntityRoute(entity_id);
}

// Encaminha para o serviço que cria uma entidade
export async function createEntity(entity: Entity): Promise<{ success: boolean, message?: string }> {
    return await CreateEntityRoute(entity);
}

// Encaminha para o serviço que atualiza uma entidade
export async function editEntity(entity: Entity): Promise<{ success: boolean, message?: string }> {
    return await EditEntityRoute(entity);
}
```

Figura 58 – *Controller* da gestão de entidades

Services

A componente dedicada aos serviços encarrega-se de estabelecer as conexões externas, através de um método específico para um determinado endereço *web*. Desta forma, a fim de colmatar esta necessidade, inclui-se o *Axios* que se trata de uma biblioteca cliente HTTP que permite fazer solicitações a um determinado *endpoint*.

O recurso ao *Axios* motivou-se por vários motivos no momento da procura pela vasta oferta de bibliotecas que estabelecem conexões:

- Possui bons padrões para trabalhar com dados JSON. Ao contrário de alternativas como a *API Fetch*, habitualmente não há a necessidade de definir os cabeçalhos.
- *Axios* possibilita executar chamadas HTTP de qualquer método, sendo flexível e apresentando poucas limitações.
- Diminui a quantidade código necessário para efetuar uma chamada a um serviço. Ao contrário da API *Fetch*, no caso do *Axios* para aceder aos dados JSON solicitados, basa utilizar a função de retorno `then()` (Barger, 2021).

```
export async function GetEntitiesRoute(): Promise<{ success: boolean, message?: string, sys_codes?: Entity[] }>{
    var company_id: string = sessionStorage.getItem("company_id") || "";
    var data = {
        comp_id: company_id
    };

    return await authAxios
        .post(`/listar-entidades`, data)
        .then(
            (response) => {
                if (response.status == 200 && response.data.success) {
                    var entities: Entity[] = response.data.body.entities;
                    entities.forEach((entity) => {
                        entity.state == 0 ?
                        entity.stateDesc = "Inativo" :
                        entity.stateDesc = "Ativo";
                    });
                    return { success: true, entities };
                }
                else if (response.status != 200) {
                    return { success: false, message: "SERVER_CONN_ERR" };
                }
                else {
                    var message: string = response.data.body.message;
                    if (message == "NOT_AUTHENTICATED") { //caso token passe do prazo
                        sessionStorage.removeItem("token");
                        sessionStorage.removeItem("user");
                        sessionStorage.setItem("isLogged", "false");
                    }
                    return { success: false, message };
                }
            })
        .catch(
            (error) => {
                var message: string = JSON.stringify(error);
                return { success: false, message };
            }
        );
}
```

Figura 59 – Implementação chamada *post* através do *Axios*

A estrutura da função exibida na Figura 59, exemplifica uma chamada *Axios* através do método `post`, onde se estrutura o cabeçalho do *request* com o identificador da empresa. Após a interação com a *API*, define-se o `then()` para aceder à informação retornada, e o `catch` atua em casos de insucesso iminente, efetuando o tratamento do erro.

5.1.2 Servidor

A implementação do lado do servidor resulta no recuso a funções que definem as diferentes lógicas resultantes da interação do utilizador com a aplicação. Este componente consulta e interage diretamente com a base de dados.

5.1.2.1 API

A inclusão da *API Gateway* surgiu com o intuito de resolver problemas de comunicação, por parte do cliente, com um sistema baseado em diversos serviços. Esta abordagem permite uma distribuição de responsabilidades eficaz.

No seguimento desta abordagem, a informação encontra-se distribuída por funções *Lambda* distintas, onde a API intervém como um ponto de entrada único. Os detalhes e as informações diferem de função para função, sendo que algumas pode não haver a necessidade de persistir a informação, e apenas possuir um comportamento de manipulação e transformação de dados.

A granularidade associada às API's disponibilizadas para este fim, é tendencialmente muito fina, ou seja, é comum os clientes necessitarem de interagir com componentes distintas do sistema, com o propósito de obterem resposta ao seu pedido (Richardson, Pattern: API Gateway / Backends for Frontends, 2017).

A *API Gateway* atua no sistema como ponto de entrada para os clientes distintos, e procede de com o redireccionamento para a lógica apropriada. Da mesma forma, executa a distribuição do pedido pelas funções necessárias para a satisfação do mesmo, utilizando vulgarmente como protocolo de comunicação solicitações HTTP.

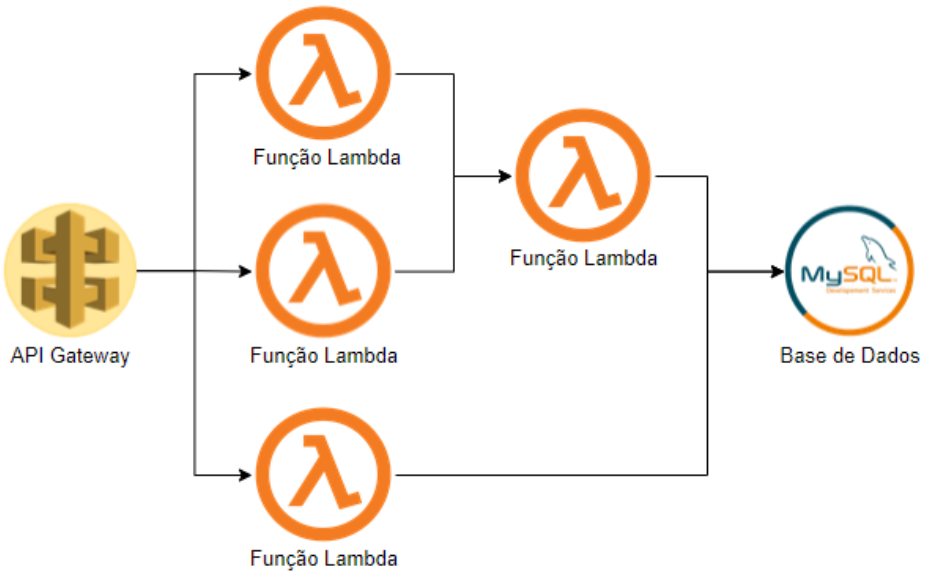


Figura 60 – Distribuição de responsabilidade da API *Gateway* pelas funções Lambda

Adicionalmente, a API trata questões de segurança, nomeadamente a autenticação, e restrições de utilização, através da filtragem do acesso a determinadas funcionalidades do sistema, inibindo assim a necessidade de aumentar a complexidade do sistema.

Solicitação HTTP

O HTTP funciona como um protocolo de requisição e resposta, tipicamente utilizado na presença de uma arquitetura cliente e servidor. A interação do utilizador com a aplicação, despoleta ações que submetem solicitações HTTP.

Desta forma, os métodos de solicitação HTTP são os ativos que indicam a ação específica desejada a ser executada em um determinado recurso. Cada método implementa uma semântica distinta, mas existem alguns recursos padrão compartilhados pelos vários métodos de solicitação HTTP.

A Tabela 20 descreve alguns dos métodos disponibilizados pelo protocolo HTTP, e que indicam a ação detalhada a ser realizada no recurso especificado.

Tabela 20 – Métodos de solicitação HTTP

MÉTODO	AÇÃO
POST	Cria um recurso associado a uma identidade
GET	Solicita uma apresentação do recurso especificado
PUT	Substitui todas as informações do recurso pelas enviadas na solicitação
DELETE	Exclui o recurso especificado na solicitação

O cliente envia uma solicitação HTTP ao servidor e, após decorrer a fluxo definido, o servidor envia de volta uma resposta. A resposta contém as informações necessárias, incluindo um código *status* no cabeçalho, que indica o sucesso ou insucesso do pedido.

Funções *Lambda*

As funções *Lambda* constituem um o serviço *serverless* disponibilizado pela *Amazon Web Services* (AWS), que permite criar funções, que suportam uma ampla gama de linguagens de programação, e permite configurar permissões e tempos de execução, de forma independente das restantes.

Adicionalmente, o *Lambda* destaca-se pela sua flexibilidade, sendo estas capazes de cumprir diferentes funções em contextos distintos, desde a integração com a API *Gateway*, a chamada de uma outra função *Lambda*, e a base de dados.

```
const AWS = require("aws-sdk");                          // Biblioteca AWS
AWS.config.update({ region: "eu-west-3" });              // Definir região do servidor

const { callFunction, verifyInput } = require("utils.js"); // Classe com funcionalidade complementares

exports.handler = async (event,context) => {
    try {
        var { comp_id } = event.body;                    // Identificador da empresa que chama serviço
        var token = event.params.header["x-access-token"]; // Token que identifica a autenticação do utilizador

        var payload = JSON.stringify({
                body: {
                    token
                }
            });

        // Valida o token, chamando uma outra função "authentication"
        var authData = await callFunction(payload, "authentication", "RequestResponse");
        if (!authData.success) {
            return authData;
        }
```

Figura 61 – Validação da autenticação numa função Lambda

A Figura 61 apresenta um excerto de que recebe a informação proveniente do cliente, e antes de realizar a chamada ao serviço, verifica se a assinatura do *request* permanece válida. A assinatura tem um prazo de expiração, obrigando o utilizador a autenticar-se novamente caso este seja ultrapassado.

Assim, cada função *Lambda* dispõe do seu próprio *container* no momento da execução, em que numa arquitetura *serverless* a alocação de recursos para o processamento da mesma, são alocadas de forma automática não sendo necessário alguma sem intervenção.

5.1.3 Base de Dados

Um sistema de gestão têxtil complexo contém diversas entidades e essas entidades contêm várias relações, isto exige a estruturação dos dados de uma forma organizada, o que inclui gravá-los numa área de armazenamento informática, designada base de dados.

5.1.3.1 Modelo de dados relacional

A construção de sistemas complexos que implicam a consulta e manipulação de dados, impõe a definição prévia do modelo de base de dados, a fim de colmatar problemas de múltiplas estruturas de dados arbitrárias, de maneira a garantir o desempenho ótimo do aplicativo (Oracle, 2020).

O modelo de dados relacional fornece uma visão global da estrutura de armazenamento de forma intuitiva, estabelecendo as relações entre as entidades que o compõe. A modelação da base de dados teve como foco os requisitos exigidos pelo setor de tinturaria e acabamentos.

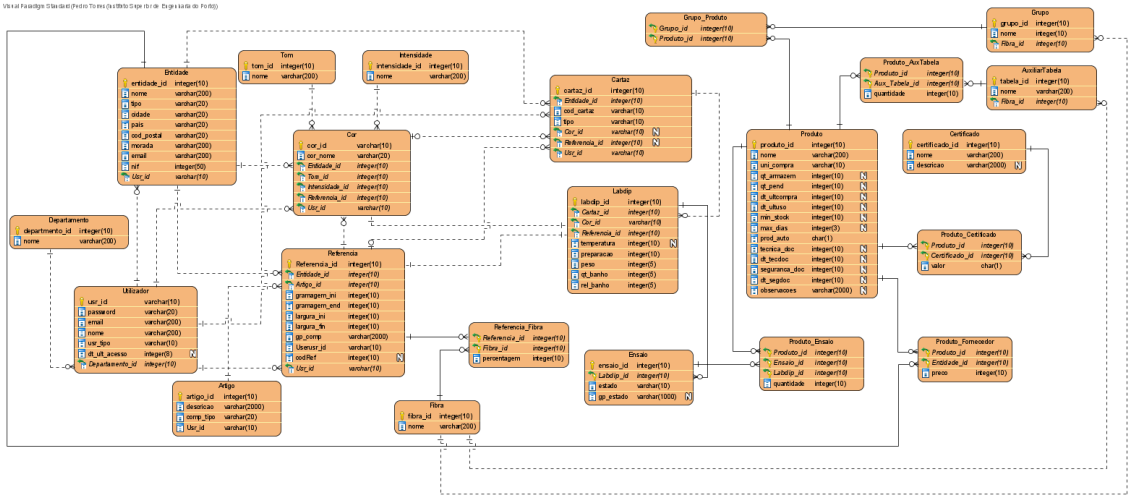


Figura 62 - Modelo de dados relacional do sistema

Como evidenciado na Figura 62, diversas relações anteriormente mencionadas na secção 4.2.1 resultam em tabelas intermédias por se tratar de ligações compostas de agregam múltiplos elementos de ambas os conceitos.

O cliente e o fornecedor, resultam na **entidade**, tratando-se assim de um único conceito, tendo em conta que partilham dos mesmos atributos que os definem, e por isso, especifica-se no tipo se se trata de um **cliente** ou **fornecedor**.

A relação estabelecida entre a **referência** e a **fibra** origina uma entidade intermédia, que especifica cada uma das **fibras** e as respetivas percentagens para uma **referência** em questão.

Este tipo de relação difere entre com o **artigo**, pelo facto de uma **referência** conter apenas um único **artigo**.

No que toca ao fornecimento de **produtos**, o mesmo produto pode ser proveniente de diversos **fornecedores**, em que cada um deles pratica o seu preço, da mesma forma que pode pertencer a **gamas** distintas.

Da mesma forma que acontece nos exemplos referidos anteriormente, um **certificado** pode estar presente em múltiplos **produtos**, mas a escala da relação entre os mesmos varia para cada um dos **produtos**.

Por último, os **ensaios** são definidos diversos **produtos** com quantidades específicas, compondo assim a fórmula do mesmo.

5.1.3.2 Função de Acesso à Base de Dados

A conexão a uma base de dados exige o uso de variáveis importantes, que devem ser encriptadas. Como se pode verificar na figura Figura 63 é utilizada a função `decryptSecret` para desencriptar as variáveis de acesso.

```
const AWS = require('aws-sdk');
AWS.config.update({ region: 'eu-west-3' });

const mysql = require('mysql');
const { decryptSecret } = require('utils.js');

exports.handler =  async (event, context, callback) => {

    var { sqlStatements, typeStatement } = event.body;
    var { functionName } = context;
    try {
        var DbHostEndpoint = await decryptSecret('DbHostEndpoint', functionName);
        var DbUsername = await decryptSecret('DbUsername', functionName);
        var DbPassword = await decryptSecret('DbPassword', functionName);
        var DbName = await decryptSecret('DbName', functionName);
    } catch (err) {
        console.error("Erro: " + err);
        return {
            success: false,
            body: { message: "DECRYPTION_ERROR" }
        };
    }

     var connection = mysql.createConnection({
        host: DbHostEndpoint,
        user: DbUsername,
        password: DbPassword,
        database: DbName
    });

    connection.connect();
```

Figura 63 – Fragmento de código que desencripta as variáveis de acesso

Para finalizar, o processo de conexão é necessário utilizar a biblioteca M*ySQL* própria de *JavaScript*, a qual fornece uma função designada `createConnection` que recebe como

parâmetros de entrada os dados desencriptados anteriormente. A partir desse momento torna-se possível executar *queries*, usando a função *query* e enviando por parâmetro a operação de base de dados construída.

```
connection.query(sqlStatements,  (error, results)=>{
    if(error){
        console.warn("Erro: " + JSON.stringify(error));
        connection.end();
        return reject({
            success: false,
            body: { message: error.errorMessage }
        });
    } else {
        connection.end();
        return resolve({
            success: true,
            body: { message: results }
        });
    }

});
```

Figura 64 - ANEXO

5.2 Testes

Na atualidade o software utilizado nas empresas é codificado por seres humanos ou é gerado através de ferramentas *low-code*. A segunda opção oferece uma maior segurança em termos de *bugs*, no entanto não garante que todos os requisitos do sistema estejam bem implementados. Os testes servem como instrumento de auxílio, aumentando a probabilidade de uma funcionalidade ter sido bem executada.

O desenvolvimento do produto foi acompanhado de testes, cuja tarefa é encontrar o maior número de erros possível com o menor esforço possível, assumindo um papel fundamental para ajudar a garantir o sucesso do produto, além de facilitar a sua manutenção e escalabilidade, e reduzir possíveis custos futuros.

O *Test-Driven Development* (TDD) foi a abordagem de desenvolvimento software aplicada no projeto têxtil, na qual são primeiro criados os casos de teste para cada uma das funcionalidades e só posteriormente é implementado o código, tornando o processo de codificação simples e livre de bugs. A Figura 65 representa o diagrama de estados deste tipo de desenvolvimento.

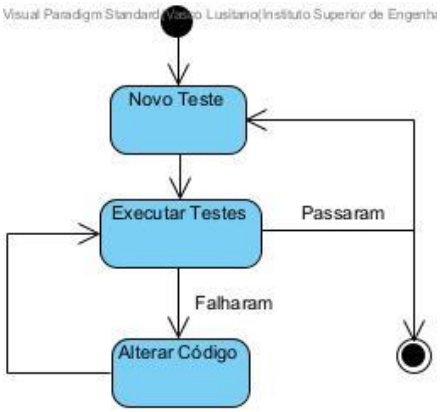


Figura 65 – Diagrama de estados do processo da abordagem TDD

O modelo de testes aplicado foi o *V-Model*, o qual dita que a execução dos testes ocorre no sentido inverso do seu planeamento correspondente. Sendo o planeamento correlacionado com a execução de testes da maneira que demostra a Figura 66.

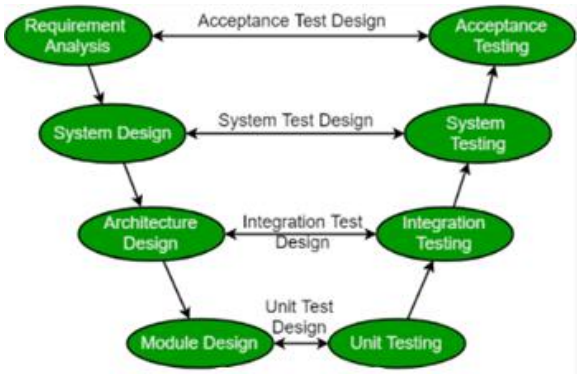


Figura 66 – Modelo de testes *V-Model*

5.2.1 Testes Unitários

Tal como o nome indica, este tipo de testes, foca-se em testar pequenas áreas e funções do sistema. Contribuindo particularmente para uma melhoria significativa da qualidade do código, pois segue as boas normas de programação por auxiliar na separação de responsabilidades (desacoplamento).

Na conjuntura do BMS, os testes unitários efetuados foram alcançados pelo meio de uma biblioteca de *JavaScript* designada *Jest*, que pode ser instalada pela linha de comandos utilizando o comando "npm install jest". No entanto este tipo de testes não cobre 100% da aplicação, devido a escassez de tempo.

```
test('login succeed', async () => {
    var loggedData: { success: boolean, message?: string } = await login('admin', 'admin');
    expect(loggedData.success).toBeTruthy();
});
```

Figura 67 – Fragmento de código de teste unitário

Assim como apresenta a Figura 67, os testes têm um tipo de função específica (*expect*) que avalia os dados de entrada segundo a função de asserção imediatamente a seguir (*toBeTruthy*). Existem diversas funções de asserção, sendo os mais reconhecidos:

- *toBe* – retorna true se o valor de entrada é igual ao valor esperado;
- *toEqual* – retorna true se o valor de entrada é igual ao valor esperado;
- *toBeNull* – retorna true se o valor de entrada for nulo;
- *toBeUndefined* – retorna true se o valor de entrada não for definido;
- *toBeDefined* – retorna true se o valor de entrada for definido;
- *toBeTruthy* – retorna true se o valor de entrada for true;
- *toBeFalsy* – retorna true se o valor de entrada for false;
- *toBeGreaterThan* – retorna true se o valor de entrada for maior que o valor esperado;
- *toBeLessThan* – retorna true se o valor de entrada for menor que o valor esperado;
- *toMatch* – retorna true se o valor de entrada coincidir com uma expressão regular;
- *toContain* – retorna true se o valor de entrada, sendo uma lista, contiver o valor esperado.

5.2.2 Testes Integração

O objetivo dos testes em análise é testar se os módulos de um software conseguem comunicar sem problemas entre si, podendo esses módulos pertencer a equipas de desenvolvimento diferentes.

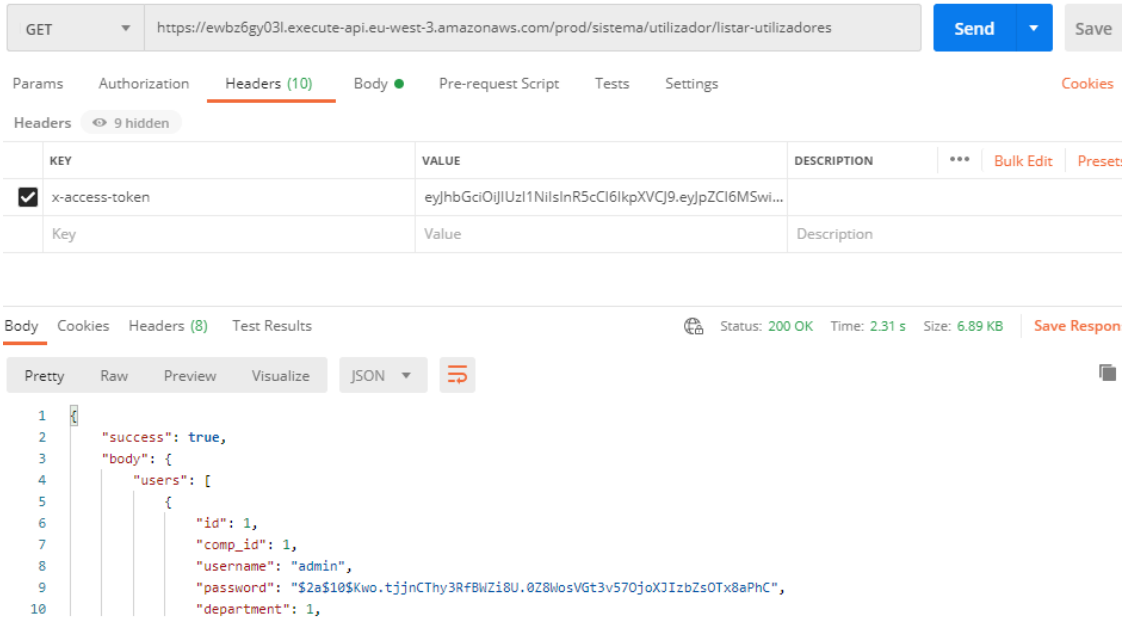
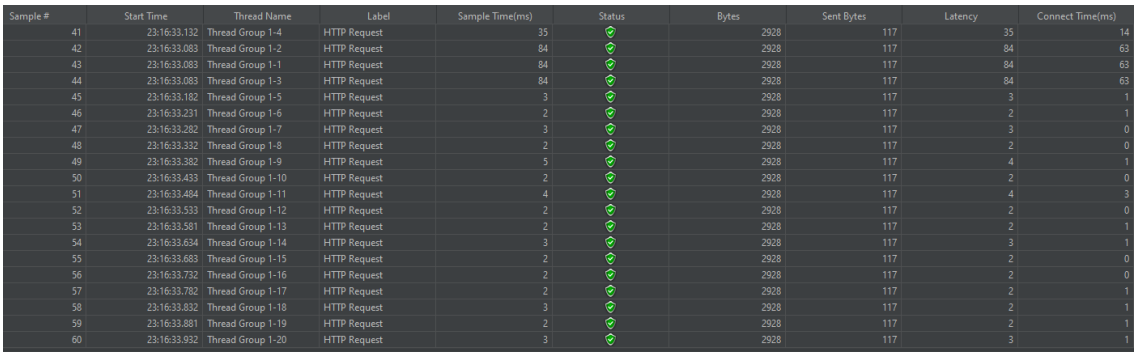


Figura 68 – Anexo

No contexto do projeto de gestão têxtil, foram realizados testes de integração do *backend*, a partir da ferramenta *Postman* como demonstra a Figura 68. Esta *framework* funciona como uma API cliente e permite a chamada de serviços REST através de dados *mock*, recebendo como input *headers* e um *body* e retornando os dados do serviço.

5.2.3 Testes de Sistema

O tipo de testes em questão procura testar o sistema como um todo, onde todos os módulos e as suas conexões são integrados de forma a verificar se o resultado é o esperado. São avaliados tanto os requisitos funcionais como não funcionais.



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
41	23:16:33.132	Thread Group 1-4	HTTP Request	35		2928	117	35	14
42	23:16:33.083	Thread Group 1-2	HTTP Request	84		2928	117	84	63
43	23:16:33.083	Thread Group 1-1	HTTP Request	84		2928	117	84	63
44	23:16:33.083	Thread Group 1-3	HTTP Request	84		2928	117	84	63
45	23:16:33.182	Thread Group 1-5	HTTP Request	3		2928	117	3	1
46	23:16:33.231	Thread Group 1-6	HTTP Request	2		2928	117	2	1
47	23:16:33.282	Thread Group 1-7	HTTP Request	3		2928	117	3	0
48	23:16:33.332	Thread Group 1-8	HTTP Request	2		2928	117	2	0
49	23:16:33.382	Thread Group 1-9	HTTP Request	5		2928	117	4	1
50	23:16:33.433	Thread Group 1-10	HTTP Request	2		2928	117	2	0
51	23:16:33.484	Thread Group 1-11	HTTP Request	4		2928	117	4	3
52	23:16:33.533	Thread Group 1-12	HTTP Request	2		2928	117	2	0
53	23:16:33.581	Thread Group 1-13	HTTP Request	2		2928	117	2	1
54	23:16:33.634	Thread Group 1-14	HTTP Request	3		2928	117	3	1
55	23:16:33.683	Thread Group 1-15	HTTP Request	2		2928	117	2	0
56	23:16:33.732	Thread Group 1-16	HTTP Request	2		2928	117	2	0
57	23:16:33.782	Thread Group 1-17	HTTP Request	2		2928	117	2	1
58	23:16:33.832	Thread Group 1-18	HTTP Request	3		2928	117	2	1
59	23:16:33.881	Thread Group 1-19	HTTP Request	2		2928	117	2	1
60	23:16:33.932	Thread Group 1-20	HTTP Request	3		2928	117	3	1

Figura 69 – Anexo

No âmbito do sistema têxtil, o software utilizado para a execução dos testes de sistema foi o *JMeter*, como asseverado na figura Figura 69. A ferramenta em apreço testa a performance de uma aplicação, estando os testes de carga e latência incluídos. Pode-se verificar que foram

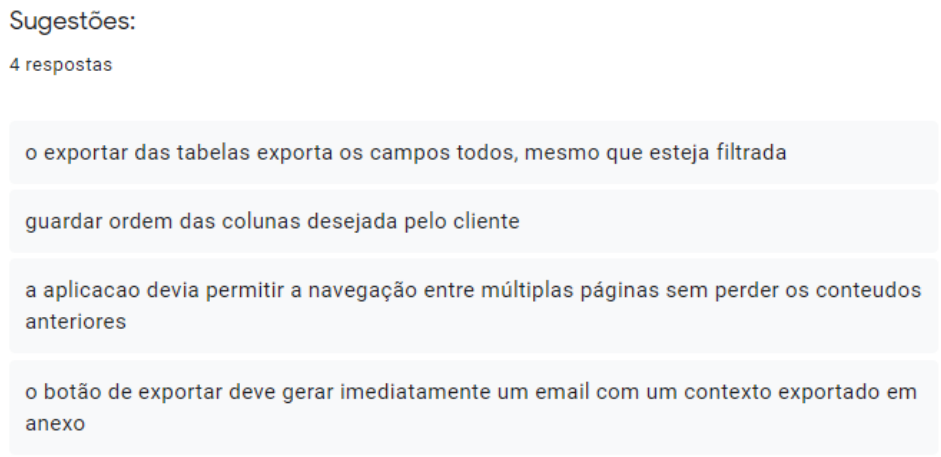
executados múltiplos pedidos ao mesmo tempo com sucesso, e também confirmar a baixa latência de cada um dos pedidos.

5.2.4 Testes de Aceitação

Ao contrário dos outros tipos de teste referidos, os testes de aceitação são realizados pelo utilizador final, ou alguém capaz de realizar o seu trabalho como substituto. Da mesma maneira que os testes de sistema, estes avaliam os requisitos funcionais e não funcionais.

No enquadramento do software de laboratório os testes de aceitação foram desempenhados por profissionais da área têxtil, que lidam com este tipo de sistemas no seu dia a dia, o seu feedback foi registado sobre a forma de inquéritos, como indicam as seguintes figuras.

Através do Anexo G pode-se verificar que a reação dos peritos foi positiva, sendo que a maioria preferiu o produto em progresso ao software que utilizam nas suas empresas. Ainda foram feitas sugestões de acréscimos *user-friendly* como se pode visualizar na figura.



Em suma, os testes de software atribuem valor ao sistema de gestão. Estes asseguram o bom funcionamento da aplicação, apesar de existirem desvantagens como os custos temporais no desenvolvimento do produto, o facto de oferecerem uma maior segurança e tornarem a capacidade de manutenibilidade mais eficiente torna a utilização desta metodologia gratificante em qualquer ambiente.

6 Experimentação e Avaliação

"Só porque os funcionários foram ensinados, não significa que eles aprenderam. É aí que entra a avaliação." (Peterson, 2019). A avaliação e experimentação servem como fonte de feedback das qualidades de um software, podendo retornar informação sobre a performance ou boa usabilidade das suas funcionalidades. O presente capítulo foca-se neste conceito. Primeiro são identificadas as hipóteses de teste e de seguida os indicadores e fontes de informação agregadas a elas, finalizando com a descrição dos métodos de avaliação a utilizar.

6.1 Hipóteses

Como referido anteriormente, o propósito desta dissertação trata-se da criação de um sistema de gestão têxtil, especificamente para as áreas de tingimento e acabamento. Este projeto é essencial para atingir o ponto de eficiência máximo das fábricas têxteis.

Para o produto ser bem-sucedido na entrada do mercado e permanecer sempre no cume dos sistemas de gestão, é necessário realizar manutenções e avaliações contínuas, para fortalecer a sua qualidade, e melhorar os seus resultados. Estes julgamentos podem ser executados através dos testes de hipóteses (Majaski, 2020).

O estudo da plausibilidade de uma hipótese designa-se de teste de hipóteses, tendo como alicerce uma amostra aleatória de algo em análise. Este método tem como objetivo principal o amontoamento de provas que admitam a veracidade da hipótese nula. Ainda existe a hipótese alternativa que, tal como o nome indica, é a opção inversa da hipótese nula. Posto isto, apenas uma das hipóteses pode ser fidedigna, enquanto a outra é refutada (Statistics How To, 2021).

No âmbito do software têxtil a hipótese nula e hipótese alternativa a considerar serão as seguintes:

- Hipótese Nula: Plataforma a desenvolver fiável, com bom performance e interface gráfica intuitiva;

- Hipótese Alternativa: Plataforma a desenvolver não cumpre com as expectativas dos clientes.

6.2 Identificação dos Indicadores e Fontes de Informação

Este subcapítulo clarifica e identifica os indicadores de avaliação e fontes de informação a dispor, no contexto das hipóteses denotadas.

6.2.1 Identificação dos Indicadores

Os indicadores de avaliação são algo imprescindível para validar se um programa funciona como deve e se corresponde às expectativas. Estes não só fornecem informações fulcrais com o intuito de se entender como tudo deve ocorrer, como também podem validar se existe algum tipo de anomalia, havendo a possibilidade de apontarem a razão do erro (CDC, 2016). Distinguem-se vários conjuntos de indicadores, no entanto os focados na análise do *software* têxtil são:

- Indicadores de *input* – incitam a maneira de como uma aplicação deve ser desenvolvida;

- Indicadores de *output* – validam se o *software* está a ter o comportamento esperado.

Os indicadores de input entram no campo do pré-negócio, ou seja, servem como precaução e não tanto como avaliação contínua, como por exemplo a informação obtida a partir de parceiros chave. Todavia, como o capítulo se trata de um prognóstico constante, os indicadores de output são os que evidenciam mais valor, tendo como exemplos:

- Requisitos funcionais desenvolvidos na integra;

- Objetivos mínimos dos requisitos não funcionais cumpridos com sucesso;

- Autoavaliação da equipa de desenvolvimento;

- Avaliação por possíveis clientes.

6.2.2 Fontes de Informação

“Como regra geral, o homem mais bem-sucedido na vida é aquele que providencia das melhores informações.” (Disraeli, 2016). A informação é a fonte do conhecimento e o conhecimento é a chave da sabedoria. Por outras palavras, a probabilidade de um produto ser

bem-sucedido aumenta se a informação obtida for verdadeira, daí a importância das fontes de informação.

Assim sendo, o empreendedor deve ser capaz de escolher sucintamente as suas fontes de informação. Para o projeto têxtil, tendo em conta as hipóteses de teste declaradas e os indicadores escolhidos, as fontes a ser utilizadas são:

- Testes de qualidade de *software* – comprovam o sucesso dos objetivos dos requisitos não funcionais, além de assegurarem o êxito da implementação dos requisitos funcionais;

- Inquéritos de satisfação – oferecem o *feedback* de indivíduos que interagiram com o produto em questão, que podem dar opiniões quanto à interface gráfica ou latência da aplicação.

6.3 Descrição da Metodologia de Avaliação

O presente subcapítulo apresenta e descreve as metodologias de avaliação referenciadas na secção 6.2.2, nomeadamente, testes de qualidade de software e inquéritos de avaliação. Estas táticas ajudam o fundador do produto a entender se a sua hipótese nula é legítima ou ilegítima.

6.3.1 Testes de Qualidade de Software

"Nenhuma quantidade de testes pode provar que um software está certo, um único teste pode provar que um software está errado." (Ghahrai, 2020). Os testes de qualidade de software não têm como foco comprovar se um sistema se encontra bem desenvolvido, mas sim identificar possíveis bugs. São distintos dois tipos de testes, os funcionais e os não funcionais, nos quais os primeiros validam se as funcionalidades de uma aplicação se comportam de acordo com as expectativas e os segundos avaliam a sua performance (The Economic Times, 2021).

Na conjuntura do projeto têxtil, ambas as categorias de testes foram aplicadas, a Tabela 21 indica quais os testes utilizados, incitando também a descrição e género de cada um.

Tabela 21 – Testes de Qualidade de Software

TESTE	DESCRIÇÃO	CATEGORIA	PROPÓSITO
Testes Unitários	Testa unidades individuais ou componentes de um software	Funcional	Fiabilidade
Testes de Integração	Testa a integração do *software* com um ambiente externo	Funcional	Fiabilidade
Testes de Aceitação	Teste não automatizado, no qual é o próprio utilizador a examinar	Funcional	Interface Gráfica, Fiabilidade

TESTE	DESCRIÇÃO	CATEGORIA	PROPÓSITO
Testes *Smoke*	Testa as funcionalidades do sistema através de eventos, automaticamente	Funcional	Fiabilidade
Testes de Carga	Testa se o sistema consegue aguentar quantidades excessivas de dados ou utilizadores	Não Funcional	Performance
Testes de Latência	Testa a velocidade do sistema, medindo o tempo de estímulo-reação	Não Funcional	Performance

6.3.2 Inquéritos de Avaliação

O método em questão é dos mais eficazes em termos de obtenção de *feedback*. A sua facilidade de implementação é elevada e disponibilizada por muitas das grandes organizações, como o exemplo da *Google* com a *Google Forms* (Keele University, 2014). Contudo, há que ter em atenção ao grupo de cidadãos ao qual é providenciado o questionário, de modo a manter uma apreciação viável, pois é preferível a qualidade à quantidade.

No enquadramento da plataforma têxtil, o conjunto de sujeitos relevantes a responder ao inquérito serão elementos de empresas têxteis que interagem diariamente com os softwares atuais do mercado e a própria equipa de desenvolvimento como autoavaliação.

Existe um vasto leque de categorias de questões às quais se pode recorrer num inquérito, sendo a escala de *Likert-type* a escolhida, isto porque, se trata de uma escala unidimensional. Por outros termos, há várias opções de resposta que variam entre o grau "concordo completamente" e "discordo completamente", como visível na Figura 70.

Fonte: (QuestionPro, 2020)

Figura 70 – Opções resposta da escala *Likert-type*

Outras opções de escalas afamadas foram refutadas pelas seguintes razões:

- Escala semântica - a avaliação é feita pela escolha de um número dentro de uma dada escala, ao contrário da de *Likert-type*, na qual os utilizadores afirmam se concordam ou discordam com a questão (Statistics How To, 2020);

- Escala de *Likert* - apresenta uma série de afirmações, enquanto a escala *Likert-type* apenas ostenta cinco ou sete possibilidades (Logframer, 2016);

- Escala de avaliações - corresponde a um parente da de *Likert*, consiste na seleção de uma entre várias opções de avaliação, em vez de focar num agrupamento de respostas à disposição num determinado espetro (Pollfish, 2020).

Em síntese, os meios de avaliação do software têxtil foram aplicados com sucesso, e os resultados obtidos a partir da sua utilização foram objetivos e positivos. O subcapítulo 5.2 descreve a forma como os métodos foram empregues no sistema de gestão.

7 Conclusão

O presente capítulo apresenta os resultados do sistema elaborado no âmbito da tese de mestrado, exibindo os objetivos realizados e conhecimentos agregados ao longo do seu desenvolvimento. Por fim, são denotadas as limitações e recomendações para o trabalho futuro.

7.1 Objetivos Concretizados

O principal objetivo do projeto foi realizar a análise, desenho e implementação de um software de gestão têxtil. Foram estudadas as possíveis tecnologias e arquiteturas de modo a se desenvolver uma solução soberana capaz de competir e extrapolar as atuais tecnologias do mercado.

Conforme um produto cresce as suas metas modificam. Estas mudanças são obrigatórias quando surgem condições internas ou externas irregulares. No contexto deste projeto, surgiram situações adversas que obrigaram à alteração dos objetivos inicialmente definidos, nomeadamente:

1. Arquitetura muito recente e diferenciada das suas concorrentes;
2. Tipo de desenvolvimento incomum e de curva de aprendizagem acentuada.

Assim, após uma análise intensiva numa fase embrionária do produto, decidiu-se dar foco ao módulo mais intrigante dos dois propostos, a gestão de laboratório. Contudo, segundo os dados obtidos pelas partes interessadas na secção REF, os objetivos foram completados com sucesso.

7.2 Limitações Encontradas

Todos os softwares têm as suas limitações por muito congruentes que sejam, essas barreiras dependem fundamentalmente de três fatores, a data de previsão de entrega do projeto, o orçamento disponível e as restrições de desenvolvimento.

No âmbito do produto criado, surgiram duas grandes limitações, uma devido ao orçamento do projeto, dado que não existiam recursos financeiros, e outra devido às restrições de desenvolvimento.

A limitação financeira diz respeito ao plano gratuito da *Amazon Web Services*, que limita a quantidade e qualidade dos serviços fornecidos. Esta adversidade teve impacto no desenvolvimento do sistema, principalmente nos serviços de encriptação, de desenvolvimento *serverless*, de *logs*, de base de dados e de comunicação a API's.

A restrição de desenvolvimento corresponde à dependência de internet em soluções baseadas em serviços AWS. Diversas empresas têxteis localizam-se em áreas rurais onde a conexão à internet é fraca ou até mesmo inexistente. De momento não existe um meio para resolver o problema em questão, dado a arquitetura da aplicação.

7.3 Trabalho Futuro

Um software presente no mercado nunca se encontra terminado, existe sempre margem para o acréscimo de novas funcionalidades e melhorias. Para um produto ser bem-sucedido e destacar-se dos seus concorrentes é obrigatória a sua manutenibilidade.

Um sistema de gestão têxtil engloba um grande número de módulos, cujo um deles é a gestão de laboratório, dado estas condições e para o software se distinguir no mercado, as restantes secções devem também ser implementadas. Havendo sempre a possibilidade de conceção de uma área financeira evoluindo de sistema de gestão para um sistema de recursos de organizações.

Na atualidade todos os softwares devem usufruir de uma página web que os patrocine e uma área de suporte, os quais são potenciais desenvolvimentos futuros. Isto oferece vantagens como abrir as portas no mercado ao produto, de modo a abranger um maior público, e a possibilidade de utilizadores proporem novas melhorias ou a correção de bugs que possam existir. Ainda existe a possibilidade de adicionar um *chatbot* ao sistema com o intuito de disponibilizar ajuda imediata aos clientes.

Para finalizar, como referido anteriormente na secção REF, existe o feedback de partes interessadas que responderam a um inquérito no qual existia um segmento próprio para sugestões, estando elas representadas na figura REF. Estas traduzem-se nos seguintes requisitos:

- Exportar apenas os campos não filtrados das tabelas;

- Armazenar na memória a ordem de disposição das colunas das tabelas;
- Permitir a manipulação de vários menus ao mesmo tempo;
- Opção de exportar diretamente ficheiro para um e-mail.

Referências

Adair, B. (2018). *The Difference Between EAM and CMMS*. Obtido de SelectHub: https://selecthub.com/cmms/difference-eam-cmms/

Agilites. (11 de 01 de 2017). *Pros and Cons of Using C# as Your Backend Programming Language*. Obtido de agilites: https://agilites.com/pros-and-cons-of-using-c-as-your-backend-programming-language.html

Airfocus. (2020). *Front End (In a Website)*. Obtido de Airfocus: https://airfocus.com/glossary/what-is-a-front-end/

Akiwatkar, R., & Arora, G. (15 de Maio de 2017). *The Drawbacks of Serverless Architecture*. Obtido de DZone: https://dzone.com/articles/the-drawbacks-of-serverless-architecture

Alexander, F. (20 de 11 de 2020). *Low-Code and No-Code: What's the Difference and When to Use What?* Obtido de outsystems: https://www.outsystems.com/blog/posts/low-code-vs-no-code/

AltexSoft. (23 de Julho de 2021). *Functional and Nonfunctional Requirements: Specification and Types*. Obtido de AltexSoft: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/

Amazon Web Services. (30 de 09 de 2020). *What is AWS Lambda?* Obtido de docs.aws: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Amazon Web Services, Inc. (18 de 01 de 2021). *https://aws.amazon.com/pt/dynamodb/faqs/*. Obtido de aws.amazon: https://aws.amazon.com/pt/dynamodb/faqs/

Anastasia. (18 de Abril de 2019). *Best Architecture for an MVP: Monolith, SOA, Microservices, or Serverless?* Obtido de RubyGarage: https://rubygarage.org/blog/monolith-soa-microservices-serverless

AngularJS. (2020). *What Is AngularJS?* Obtido de docs.angularjs: https://docs.angularjs.org/guide/introduction

AppSheet. (2019). *AppSheet Pricing*. Obtido de solutions.appsheet: https://solutions.appsheet.com/pricing

Appy Pie. (07 de 09 de 2016). *How Much Does an App Cost?* Obtido de appypie: https://www.appypie.com/app-builder/pricing-plan

B, B. K. (02 de 06 de 2020). *Low Code vs. Traditional Development*. Obtido de zoho: https://www.zoho.com/creator/decode/low-code-vs-traditional-development/

Barger, R. (2021). *freeCodeCamp.org*. Obtido de https://www.freecodecamp.org/news/how-to-use-axios-with-react/#what-is-axios

Basiel. (03 de Abril de 2021). *What are pantone colors?* Obtido de Merchandise-essentials: https://blog.merchandise-essentials.com/what-are-pantone-colors

Belani, G. (17 de 12 de 2020). *Programming Languages You Should Learn in 2020*. Obtido de computer: https://www.computer.org/publications/tech-news/trends/programming-languages-you-should-learn-in-2020

Bernardo, A. (04 de 2018). Fast MBA - Empreendedorismo, Negócios e Startups na Prática. *BMG Canvas - Estudo de Caso 2 - Gráfica Online - Parte 1*.

Bernardo, A. (04 de 2018). Fast MBA - Empreendedorismo, Negócios e Startups na Prática. *BMG Canvas - Estudo de Caso 2 - Gráfica Online - Parte 2*.

Berners-Lee, T. (2006). *A Framework for Web Science.* Now Publishers Inc.

Bernshteyn, R. (5 de Maio de 2018). *The Rise of BSM Ushers in New Era of Enterprise Tech Innovation*. Obtido de Cbronline: https://www.cbronline.com/opinion/rise-bsm-ushers-new-era-enterprise-tech-innovation

Bhatia, S. (09 de 04 de 2020). *PostgreSQL vs MySQL: Everything You Need to Know*. Obtido de hackr: https://hackr.io/blog/postgresql-vs-mysql

Blank, S. (31 de 08 de 2009). *The Customer Development Manifesto*. Obtido de steveblank: https://steveblank.com/2009/08/31/the-customer-development-manifesto-reasons-for-the-revolution-part-1/

Booth, J. (Setembro de 2021). *Mixpanel*. Obtido de https://mixpanel.com/blog/what-is-a-technology-stack/

Burnson, F. (3 de Março de 2016). *The Buffet of ERP Modules: Why You Should Build Your System à la Carte*. Obtido de SoftwareAdvice: https://www.softwareadvice.com/resources/erp-modules-benefits/

Butler, B. (07 de 04 de 2016). *What is Amazon cloud's Lambda and why is it a big deal?* Obtido de networkworld: https://www.networkworld.com/article/3053111/what-is-amazon-cloud-s-lambda-and-why-is-it-a-big-deal.html

Byrne, J. (2019). *What are the 7 stages of a new product development process*. Obtido de cognidox: https://www.cognidox.com/blog/7-stages-of-new-product-development-process

Campbell, C. (2017). *Charlie Campbell quote: Javascript is the duct tape of the Internet*. Obtido de storemypic: https://www.storemypic.com/image/R9cM

Carey, S. (13 de Agosto de 2019). *What is serverless computing and which enterprises are adopting it?* Obtido de ComputerWorld: https://www.computerworld.com/article/3427298/what-is-serverless-computing-and-which-enterprises-are-adopting-it.html

Cast. (2020). *Software Architecture*. Obtido de CastSoftware: https://www.castsoftware.com/glossary/Software-Architecture-definition-examples-explanation-tools-principle

CDC. (02 de Dezembro de 2016). *Indicators*. Obtido de cdc: https://www.cdc.gov/eval/indicators/index.htm

Cherry, B. (17 de Setembro de 2019). *What Is a Case Study?* Obtido de Bluleadz: https://www.bluleadz.com/blog/15-of-the-very-best-case-study-examples

Cloudflare. (2021). *Serverless Microservices*. Obtido de Cloudflare: https://www.cloudflare.com/learning/serverless/glossary/serverless-microservice/

Coder Academy. (10 de 06 de 2016). *Top 32 Sites Built With ReactJS*. Obtido de medium: https://medium.com/@coderacademy/32-sites-built-with-reactjs-172e3a4bed81

Colormetrix. (2015). *4TEX - Software de Gestão de Produção*. Obtido de Colormetrix: http://www.colormetrix.pt/software-de-gestatildeo-de-produccedilatildeo.html

Cooper, R. (Fevereiro de 1990). Stage-Gate Systems: A New Tool for Managing New Products. *Stage-Gate: A New Tool for Managing New Products*, pp. 2-6.

Crosett, L. (23 de 01 de 2019). *How to determine if Amazon DynamoDB is appropriate for your needs, and then plan your migration*. Obtido de aws.amazon: https://aws.amazon.com/pt/blogs/database/how-to-determine-if-amazon-dynamodb-is-appropriate-for-your-needs-and-then-plan-your-migration/

DA-14. (04 de 04 de 2017). *TOP 10 ADVANTAGES OF USING REACT.JS*. Obtido de da-14: https://da-14.com/blog/its-high-time-reactjs-ten-reasons-give-it-try

Darmon, Y. (21 de Outubro de 2016). *Création d'un modèle NoSQL avec Mongoose et NodeJS*. Obtido de Supinfo: https://www.supinfo.com/articles/single/2941-creation-modele-nosql-avec-mongoose-nodejs

Datamonplus. (2020). *Datamonplus*. Obtido de Datamonplus: https://datamonplus.com/

DB-Engines. (02 de 2021). *DB-Engines Ranking - Trend Popularity*. Obtido de db-engines: https://db-engines.com/en/ranking_trend

DB-Engines. (2021). *System Properties Comparison Amazon DynamoDB vs. Cassandra vs. MongoDB*. Obtido de db-engines: https://db-engines.com/en/system/Amazon+DynamoDB%3BCassandra%3BMongoDB

DB-Engines. (2021). *System Properties Comparison Microsoft SQL Server vs. MySQL vs. PostgreSQL*. Obtido de db-engines: https://db-engines.com/en/system/Microsoft+SQL+Server%3BMySQL%3BPostgreSQL

Deshpande, T. (2014). *Mastering DynamoDB.* Packt. Obtido de subscription.packtpub.

Disraeli, B. (13 de Maio de 2016). *"As a general rule, the most successful man in life is the man who has the best information." (Benjamin Disraeli)*. Obtido de developingsuperleaders: https://developingsuperleaders.wordpress.com/2016/05/13/as-a-general-rule-the-most-successful-man-in-life-is-the-man-who-has-the-best-information-benjamin-disraeli/

Dreverman, S. (12 de 01 de 2019). *Starting a Low-code application architecture*. Obtido de medium: https://medium.com/@stefan.dreverman/starting-a-low-code-application-architecture-13170fcd6fc7

Dugal, J. (17 de Junho de 2021). *An Overview To The Portugal Textile Industry: A Strong Industry Known For Its Craftmanship And Quality*. Obtido de FashionABC: https://www.fashionabc.org/overview-portugal-textile-industry-strong-industry-known-craftmanship-quality/

Duhigg, C. (14 de 08 de 2016). *Charles Duhigg Quote*. Obtido de azquotes: https://www.azquotes.com/quote/1467147

Durães, S. (2020). *Corantes Reativos Para Fibras Celulósicas.* Porto.

Ecosystem, E. (2019). *DJANGO HISTORY*. Obtido de education-ecosystem: https://www.education-ecosystem.com/guides/programming/django/history

Edelhoff, R. (5 de Setembro de 2019). *How serverless architectures change development process and project business*. Obtido de ITemis: https://blogs.itemis.com/en/how-serverless-architectures-change-development-process-and-project-business

Education Ecosystem. (2019). *REACT.JS HISTORY*. Obtido de education-ecosystem: https://www.education-ecosystem.com/guides/programming/react-js/history

EDUCBA. (06 de 01 de 2021). *Is MySQL Programming Language?* Obtido de educba: https://www.educba.com/is-mysql-programming-language/

Eeles, P. (15 de Fevereiro de 2006). IBM. *What is a software architecture?ti*, pp. 1-4.

Enfroy, A. (12 de Setembro de 2020). *Your ultimate guide to defining your product's value proposition*. Obtido de Qualtrics: https://www.qualtrics.com/experience-management/product/value-proposition/

ETM. (2021). *ETM*. Obtido de Empresa Têxtil da Maganha: https://textilmaganha.com/

Exadel Media. (12 de 11 de 2017). *Old Reliable: A History of MySQL*. Obtido de exadel: https://exadel.com/news/old-reliable-mysql-history/

Fedon, S. (10 de 03 de 2020). *The Right Medical Device Starts with a House of Quality Matrix*. Obtido de mindflowdesign: https://www.mindflowdesign.com/insights/medical-device-design-house-of-quality/

Fee, N. (28 de Agosto de 2020). *What Is Serverless Architecture? Key Benefits and Limitations*. Obtido de NewRelic: https://blog.newrelic.com/engineering/what-is-serverless-architecture/

Foust, J. (16 de fevereiro de 2021). *SpaceX launches Starlink satellites, but booster landing fails*. Obtido de spacenews: https://spacenews.com/spacex-launches-starlink-satellites-but-booster-landing-fails/

Fowler, M. (29 de Maio de 2019). *Is High Quality Software Worth the Cost?* Obtido de MartinFowler: https://martinfowler.com/articles/is-quality-worth-cost.html

G., D. (31 de 10 de 2019). *SQL Server vs MySQL: Overview, Similarities, Differences*. Obtido de hostinger: https://www.hostinger.com/tutorials/difference-between-mysql-and-sql-server

Gall, R. (23 de 05 de 2018). *Is web development dying?* Obtido de hub.packtpub: https://hub.packtpub.com/is-web-development-dying/

Gancarz, R. (23 de 07 de 2019). *An essential guide to the 2019 serverless ecosystem*. Obtido de techbeacon: https://techbeacon.com/enterprise-it/essential-guide-2019-serverless-ecosystem

GeneXus. (01 de 02 de 2020). *Pricing Plans - GeneXus Europe*. Obtido de genexus: https://www.genexus.io/pricing-plans/

GeneXus. (26 de 01 de 2021). *GeneXus ERP Connector for SAP*. Obtido de training.genexus: https://training.genexus.com/pt/aprendizagem/cursos/sap/genexus-erp-connector-for-sap/introducao-ao-desenvolvimento-de-aplicacoes-moveis-conectadas-ao-sap-erp-com-genexus

Ghahrai, A. (24 de Julho de 2020). *The Ultimate List of 100 Software Testing Quotes*. Obtido de abstracta: https://abstracta.us/blog/tools/ultimate-list-100-software-testing-quotes/

Gilbert, N. (2020 de 09 de 22). *AppSheet Review*. Obtido de reviews.financesonline: https://reviews.financesonline.com/p/appsheet/

Gnatyk, R. (3 de Outubro de 2018). *Microservices vs Monolith: which architecture is the best choice for your business?* Obtido de N-iX: https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/

Google. (02 de 02 de 2021). *Como escrever Cloud Functions*. Obtido de cloud.google: https://cloud.google.com/functions/docs/writing

Google. (18 de 01 de 2021). *Visão geral do Cloud Functions*. Obtido de cloud.google: https://cloud.google.com/functions/docs/concepts/overview

Goyal, A. (15 de 02 de 2017). *Node.js Vs Java and .Net*. Obtido de linkedin: https://www.linkedin.com/pulse/nodejs-vs-java-net-ajay-goyal

Gregg, M. (2006). *Hack the Stack.* Obtido de https://www.sciencedirect.com/topics/computer-science/presentation-layer

Gudelli, A. (19 de 10 de 2019). *History of AngularJs*. Obtido de angularjswiki: https://www.angularjswiki.com/angular/history-of-angularjs/

Hack Reactor. (18 de 11 de 2018). *What is JavaScript Used For?* Obtido de hackreactor: https://www.hackreactor.com/blog/what-is-javascript-used-for

Healey, D. (25 de 11 de 2020). *Low Code. What is no code? The pros and cons of no code for software development.* Obtido de codebots: https://codebots.com/low-code/what-is-no-code-the-pros-and-cons-of-no-code-for-software-development

Higginbotham, J. (20 de Novembro de 2016). *Why Do Microservices Need an API Gateway?* Obtido de DZone: https://dzone.com/articles/why-do-microservices-need-an-api-gateway

Hristozov, K. (19 de 07 de 2019). *MySQL vs PostgreSQL -- Choose the Right Database for Your Project*. Obtido de developer.okta: https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres

Huang, T. (7 de Março de 2018). Internacional Journal of Information Systems and Project Management. *Decision-making to switch your ERP system: empirical*, pp. 3-7.

IntexSoft. (15 de 07 de 2019). *Node.js vs Java: Why Compare?* Obtido de intexsoft: https://www.intexsoft.com/blog/post/java-vs-nodejs.html

Ismail, N. (11 de Abril de 2017). *Low-code technology: an emerging term that needs more definition*. Obtido de Information-age: https://www.information-age.com/low-code-technology-emerging-term-definition-123465654/

Jacobson, I. (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach.* Addison-Wesley.

Javatpoint. (2018). *C# History*. Obtido de javatpoint: https://www.javatpoint.com/csharp-history

Javatpoint. (2018). *History of Java*. Obtido de javatpoint: https://www.javatpoint.com/history-of-java

Joseph, N. (12 de 01 de 2016). *An Introduction to Node.js, the Server Side JavaScript*. Obtido de opensourceforu: https://www.opensourceforu.com/2016/01/an-introduction-to-node-js-the-server-side-javascript/

Josephson, M. (27 de Fevereiro de 2019). *How to Build Workflows*. Obtido de OutSystems: https://www.outsystems.com/blog/posts/low-code-workflow/

Joshi, P. (25 de 11 de 2020). *Best backend Quotes, Status, Shayari, Poetry & Thoughts*. Obtido de yourquote: https://www.yourquote.in/tags/backend/quotes

Keele University. (13 de Setembro de 2014). *Using Google Forms for Module Evaluations*. Obtido de keele: https://www.keele.ac.uk/hss/googleforms/

Khan, A. (15 de 05 de 2020). *10 Best Low-Code And No-Code Application Development Platforms in 2021*. Obtido de hackernoon: https://hackernoon.com/10-low-code-and-no-code-application-development-platforms-ew513y8q

Kiong, L. V. (2008). *Visual Basic 6 Made Easy.*

Koen, P. (2018). *Front End Innovation*. Obtido de frontendinnovation: http://www.frontendinnovation.com/fei

Kokemuller, N. (12 de Fevereiro de 2019). *What Is Customer Perceived Value?* Obtido de Smallbusiness: https://smallbusiness.chron.com/customer-perceived-value-23692.html

Kumar, B. (20 de 01 de 2021). *Low Code vs. Traditional Development*. Obtido de zoho: https://www.zoho.com/creator/decode/low-code-vs-traditional-development/

Lebeaux, R. (21 de Agosto de 2014). *Business service management (BSM)*. Obtido de Searchcio.techTarget: https://searchcio.techtarget.com/definition/business-service-management-BSM

Leblanc, M. (20 de 06 de 2018). *A user interface is like a joke*. Obtido de uxdesign: https://uxdesign.cc/a-user-interface-is-like-a-joke-955905e7865b

Leffingwell, D. (10 de Fevereiro de 2021). *Domain Modeling*. Obtido de ScaleAgileFramework: https://www.scaledagileframework.com/domain-modeling/

Lilly, R. (17 de 02 de 2021). *Quote By Ryan Lilly*. Obtido de goodreads: https://www.goodreads.com/quotes/5700351-make-something-people-want-and-sell-that-or-be-someone

Logframer. (11 de Novembro de 2016). *Likert type scale*. Obtido de logframer: https://www.logframer.eu/content/likert-type-scale

Looper, J. (19 de Março de 2019). *What is Low-Code?* Obtido de Progress: https://www.progress.com/blogs/what-is-low-code

MacWin. (2021). *ERP MacWin*. Obtido de Macwin: https://www.macwin.pt/

Magaji, J. (25 de 08 de 2020). *The History of Node.js*. Obtido de section: https://www.section.io/engineering-education/history-of-nodejs/

Magoulas, R., & Guzikowski, C. (12 de Novembro de 2019). *O'Reilly serverless survey 2019: Concerns, what works, and what to expect*. Obtido de Oreilly: https://www.oreilly.com/radar/oreilly-serverless-survey-2019-concerns-what-works-and-what-to-expect/

Majaski, C. (24 de Outubro de 2020). *Hypothesis Testing*. Obtido de investopedia: https://www.investopedia.com/terms/h/hypothesistesting.asp

Malhotra, A. (22 de 04 de 2019). *9 BEST REASONS TO CHOOSE JAVA FOR WEB DEVELOPMENT*. Obtido de xicom: https://www.xicom.biz/blog/9-best-reasons-to-choose-java-for-web-development/

Mansfield, D. (23 de Outubro de 2018). *Marketing theory: understanding customer value*. Obtido de Builtvisible: https://builtvisible.com/understanding-customer-value/

Mao, S. (2010). Cognitive Radio Communications and Networks. *Academic Press*, 201-234.

Maruti TechLabs Pvt Ltd. (23 de 11 de 2020). *No Code/Low Code Vs. Traditional Development - Which Team Should You Pick?* Obtido de marutitech: https://marutitech.com/no-code-low-code-vs-traditional-development/

Mashkowski, N. (31 de 03 de 2016). *Introducing Azure Functions*. Obtido de azure.microsoft: https://azure.microsoft.com/en-us/blog/introducing-azure-functions/

Meer, B. V. (Abril de 26 de 2018). *Four major disadvantages of no code that you must know*. Obtido de Moqod: https://moqod.com/no-code-disadvantages/

Microsoft. (30 de Ouubro de 2019). *Microservices architecture style*. Obtido de Microsoft: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

Microsoft. (29 de 01 de 2021). *Preços do Power Apps*. Obtido de powerapps.microsoft: https://powerapps.microsoft.com/pt-pt/pricing/

Mindfire Solutions. (24 de 04 de 2017). *Advantages and Disadvantages of Python Programming Language*. Obtido de medium: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121

Moenaert, R., Meyer, A. D., Souder, W., & Deschoolmeester, D. (1995). *R&D / Marketing communication during the fuzzy front-end* (Vol. 42). IEEE Transactions on Engineering Management. doi:10.1109/17.403743

MongoDB, Inc. (2021). *Our Mission To free the genius within everyone by making data stunningly easy to work with*. Obtido de mongodb: https://www.mongodb.com/company

MongoDB, Inc. (2021). *The database for modern applications*. Obtido de mongodb: https://www.mongodb.com/

MongoDB, Inc. (2021). *What is NoSQL?* Obtido de mongodb: https://www.mongodb.com/nosql-explained

Monus, A. (9 de Outubro de 2018). *What are microservices? The pros, cons, and how they work*. Obtido de RayGun: https://raygun.com/blog/what-are-microservices/

Murphy, S. A., & Kumar, V. (1997). *The front end of new product development: a Canadian survey.*

Nemer, J. (13 de Novembro de 2019). *Advantages and Disadvantages of Microservices Architecture*. Obtido de Cloud Academy: https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/

Norman, D. (1990). *The Difference Between UX and UI Design – A Beginner's Guide*. Obtido de Careerfoundry: https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide/

Oracle. (08 de Março de 2020). *What is a Relational Database (RDBMS)?* Obtido de Oracle: https://www.oracle.com/in/database/what-is-a-relational-database/

Ottinger, T., & Langr, J. (Abril de 2009). *Agile in a Flash*. Obtido de Agileinaflash: http://agileinaflash.blogspot.com/2009/04/furps.html

Pandit, N. (02 de 01 de 2020). *NoSQL Data Architecture Patterns*. Obtido de geeksforgeeks: https://www.geeksforgeeks.org/nosql-data-architecture-patterns/

Paulillo, J. (16 de Dezembro de 2020). *Conheça a equação que ensina como criar valor para o cliente*. Obtido de Agendor: https://www.agendor.com.br/blog/como-criar-valor-para-o-cliente/

Peixoto, A., & Mendes, L. (24 de Setembro de 2012). *Inovações para o enobrecimento têxtil*. Obtido de Textilia: http://www.textilia.net/materias/ler/textil/processo-e-tecnologia--acabamento-tingimento-estamparia-lavanderia/inovacoes_para_o_enobrecimento_textil

Perkins. (28 de Abril de 2020). *What is ERP? Key features of top enterprise resource planning systems*. Obtido de CIO: https://www.cio.com/article/2439502/what-is-erp-key-features-of-top-enterprise-resource-planning-systems.html

Perkins, B. (28 de Abril de 2020). *What is ERP? Key features of top enterprise resource planning systems*. Obtido de CIO: https://www.cio.com/article/2439502/what-is-erp-key-features-of-top-enterprise-resource-planning-systems.html

Peterson, K. (12 de Julho de 2019). *Measuring Training Effectiveness*. Obtido de northpass: https://www.northpass.com/blog/measuring-training-effectiveness

Phelps, O. (03 de março de 2021). *Guide to non-functional requirements: types and examples*. Obtido de boxuk: https://www.boxuk.com/insight/guide-to-non-functional-requirements-types-and-examples/

Pollfish. (03 de Março de 2020). *Likert scale questions: What are they and how do you write them?* Obtido de resources.pollfish: https://resources.pollfish.com/market-research/rating-scales-and-likert-scales/

PostgreSQL. (24 de 07 de 2014). *A Brief History of PostgreSQL*. Obtido de postgresql: https://www.postgresql.org/docs/8.4/history.html

Pratt, K. (Janeiro de 2020). *Low-code and no-code development platforms*. Obtido de Searchsoftwarequality: https://searchsoftwarequality.techtarget.com/definition/low-code-no-code-development-platform

Program-Ace. (14 de 08 de 2020). *How Good Is Node.js For Backend Development?* Obtido de program-ace: https://program-ace.com/blog/node-js-for-backend-development/

Protasiewicz, J. (04 de 07 de 2018). *What is Django Used For?* Obtido de netguru: https://www.netguru.com/blog/why-use-django

Purcell, W. (31 de Outubro de 2019). *The Importance of Innovation in Business*. Obtido de Northeastern: https://www.northeastern.edu/graduate/blog/importance-of-innovation/

Quality-One. (19 de 07 de 2018). *Quality Function Deployment (QFD)*. Obtido de quality-one: https://quality-one.com/qfd/

QuestionPro. (04 de Agosto de 2020). *What is a Likert Scale*. Obtido de questionpro: https://www.questionpro.com/blog/what-is-likert-scale/

Radley, D. (27 de Setembro de 2017). *What Is ERP? Definition and Introduction*. Obtido de TechnologyEvaluation: https://www3.technologyevaluation.com/research/article/what-is-erp-definition-and-introduction.html

Ratnakar, S. (2011). *Open the Windows.* Hay House India.

Rayome, A. D. (15 de Janeiro de 2019). *Most popular software development technologies*. Obtido de TechRepublic: https://www.techrepublic.com/article/top-10-most-popular-software-development-technologies/

Reid, S. E., & Brentani, U. D. (2004). The Fuzzy Front End of New Product Development for Dis-. *The journal of Product Innovation*, pp. 84-170.

Reynolds, J. (05 de 06 de 2020). *8 World-Class Software Companies That Use Python*. Obtido de realpython: https://realpython.com/world-class-companies-using-python/

Richardson, C. (15 de Junho de 2015). *Building Microservices: Using an API Gateway*. Obtido de NGinx: https://www.nginx.com/blog/building-microservices-using-an-api-gateway/

Richardson, C. (2017). *Pattern: API Gateway / Backends for Frontends*. Obtido de Microservices: https://microservices.io/patterns/apigateway.html

Richardson, C. (2017). *Pattern: API Gateway / Backends for Frontends*. Obtido de Microservices: https://microservices.io/patterns/apigateway.html

Romanowski, J. (19 de 08 de 2020). *The Most Popular Databases in 2020*. Obtido de learnsql: https://learnsql.com/blog/most-popular-sql-databases-2020/

Sam, S. (24 de 07 de 2018). *C# Language advantages and applications*. Obtido de tutorialspoint: https://www.tutorialspoint.com/Chash-Language-advantages-and-applications

Samadder, S. (02 de 11 de 2020). *Frontend vs Backend*. Obtido de geeksforgeeks: https://www.geeksforgeeks.org/frontend-vs-backend/

Schade, A. (4 de Maio de 2014). *Responsive Web Design (RWD) and User Experience*. Obtido de NNGroup: https://www.nngroup.com/articles/responsive-web-design-definition/

Schulz, J. (21 de 03 de 2018). *CASSANDRA USE CASES: WHEN TO USE AND WHEN NOT TO USE CASSANDRA*. Obtido de blog.pythian: https://blog.pythian.com/cassandra-use-cases/

Serverless360. (19 de 10 de 2020). *Microsoft Azure Functions*. Obtido de serverless360: https://www.serverless360.com/azure-functions

Shannon Bradshaw, E. B. (12 de 2019). *MongoDB: The Definitive Guide, 3rd Edition.* O'Reilly Media, Inc.

Sharma, A. (26 de 10 de 2018). *Difference between SQL and NoSQL*. Obtido de geeksforgeeks: https://www.geeksforgeeks.org/difference-between-sql-and-nosql/

Sharma, K. (2 de Março de 2015). *Impact of technology on international business.* Obtido de Slideshare: https://www.slideshare.net/kiransharma32/impact-of-technology-on-international-business

Shet, S. (3 de Maio de 2016). *Limitations of a Monolithic application and need for adapting Micro services Architecture*. Obtido de JavaCodeGeeks: https://www.javacodegeeks.com/2016/05/limitations-monolithic-application-need-adapting-micro-services-architecture.html

Silva, R. M. (2018). Preparação e Tinturaria. *Academia CITEVE - Preparação e Tinturaria*, (pp. 5-8). Famalicão.

Simform. (06 de 12 de 2020). *AWS Lambda vs Azure Functions vs Google Cloud Functions: Comparing Serverless Providers*. Obtido de simform: https://www.simform.com/aws-lambda-vs-azure-functions-vs-google-functions/#dependencies

Simform. (08 de 09 de 2020). *Node.js vs Django: Key differences, popularity, use cases and more*. Obtido de simform: https://www.simform.com/nodejs-vs-django/

Simplilearn. (2021). *Apache Cassandra Data Model Tutorial*. Obtido de simplilearn: https://www.simplilearn.com/cassandra-data-model-tutorial-video

Smallcombe, M. (19 de 05 de 2020). *SQL vs NoSQL: 5 Critical Differences*. Obtido de xplenty: https://www.xplenty.com/blog/the-sql-vs-nosql-difference/

Spacey, J. (13 de 07 de 2017). *4 Types of Commercialisation*. Obtido de simplicable: https://simplicable.com/new/commercialisation

Spendel, T. (25 de Maio de 2020). *Pros and cons of Low-code development platforms*. Obtido de Blog.Skyrise: https://blog.skyrise.tech/low-code-development-platforms

SQL & NoSQL Databases. (2019). Andreas Meier, Michael Kaufmann: Springer Vieweg.

Statistics How To. (11 de Dezembro de 2020). *Semantic Differential Scale: Definition, Examples*. Obtido de statisticshowto: https://www.statisticshowto.com/semantic-differential-scale/

Statistics How To. (04 de Março de 2021). *Hypothesis Testing*. Obtido de statisticshowto: https://www.statisticshowto.com/probability-and-statistics/hypothesis-testing/

Strategyzer. (09 de 03 de 2017). *Value Proposition Canvas*. Obtido de strategyzer: https://www.strategyzer.com/blog/value-proposition-canvas-a-tool-to-understand-what-customers-really-want

Stroustrup, B. (2000). *The most important single aspect of software development is to be clear*. Obtido de idlehearts: https://www.idlehearts.com/1782727/the-most-important-single-aspect-of-software-development-is-to-be-clear

Sullivan, D. (2015). *NoSQL for Mere Mortals.* Pearson Education20.

Suschevich, A. (16 de Janeiro de 2020). *What Is a Technology Stack? Choosing the Right Tech Stack For Your Web Project*. Obtido de Medium: https://medium.com/swlh/what-is-a-technology-stack-choosing-the-right-tech-stack-for-your-web-project-3f295cf60f10

Sweeney, J. (22 de Janeiro de 2018). *What Are Lab Dips? — How to Get the Perfect Color for Your Garment*. Obtido de Makersrow: https://makersrow.com/blog/2018/01/what-are-lab-dips-how-to-get-the-perfect-color-for-your-garment/

Swinhoe, D. (13 de Outubro de 2017). *GitHub CEO: "The future of coding is no coding at all"*. Obtido de IDGConnect: https://www.idgconnect.com/article/3578431/github-ceo-the-future-of-coding-is-no-coding-at-all.html

System Properties Comparison Microsoft SQL Server vs. MySQL vs. PostgreSQL. (2021). Obtido de db-engines: https://db-engines.com/en/system/Microsoft+SQL+Server%3BMySQL%3BPostgreSQL

TechMagic. (10 de 04 de 2020). *AWS Lambda vs Google Cloud Functions vs Azure Functions — What to Choose in 2020?* Obtido de medium: https://medium.com/techmagic/aws-lambda-vs-google-cloud-functions-vs-azure-functions-what-to-choose-in-2020-6d5340b79d98

TechTarget. (October de 2021). Obtido de https://searchsoftwarequality.techtarget.com/definition/3-tier-application

The Economic Times. (06 de Março de 2021). *Definition of 'Software Testing'*. Obtido de economictimes.indiatimes: https://economictimes.indiatimes.com/definition/software-testing

Thor, W.-M. (11 de 01 de 2018). *5 top programming languages to learn server-side web development*. Obtido de twm: https://twm.me/best-programming-languages-and-frameworks-for-server-side-web-development/

Toprak, M. (05 de 08 de 2020). *What is MSSQL?* Obtido de medium: https://medium.com/@toprak.mhmt/what-is-mssql-9a152d7d4ed0

upGrad. (13 de 11 de 2019). *8 Exciting Full Stack Coding Project Ideas & Topics For Beginners*. Obtido de upgrad: https://www.upgrad.com/blog/full-stack-coding-project-ideas-topics-for-beginners/

Usability. (09 de Outubro de 2013). *Use Cases*. Obtido de Usability: https://www.usability.gov/how-to-and-tools/methods/use-cases.html

Vega, M. A. (24 de Junho de 2019). *Quick Guide: Monolith vs Microservices vs Serverless*. Obtido de Hexacta: https://www.hexacta.com/quick-guide-monolith-vs-microservices-vs-serverless/

Venturi, A. (8 de Dezembro de 2014). *Impact of Technology on Management and Organizations*. Obtido de LinkedIn: https://www.linkedin.com/pulse/20141208111749-6038885-impact-of-technology-on-management-and-organizations/

Viescas, J. L. (2018). *SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL.* Pearson / Longman.

Visão geral do Cloud Functions. (18 de 01 de 2021). Obtido de cloud.google: https://cloud.google.com/functions/docs/concepts/overview

Vliet, V. v. (2019). *Stage Gate Process by Robert Cooper*. Obtido de Toolshero: https://www.toolshero.com/innovation/stage-gate-process-robert-cooper/

Warwick Manufacturing Group. (2007). Quality Function Deployment. *Product Excellence using Six Sigma*, pp. 6-24.

Wittmer, P. (31 de Dezembro de 2019). *Microservices Disadvantages & Advantages*. Obtido de TiempoDev: https://www.tiempodev.com/blog/disadvantages-of-a-microservices-architecture/

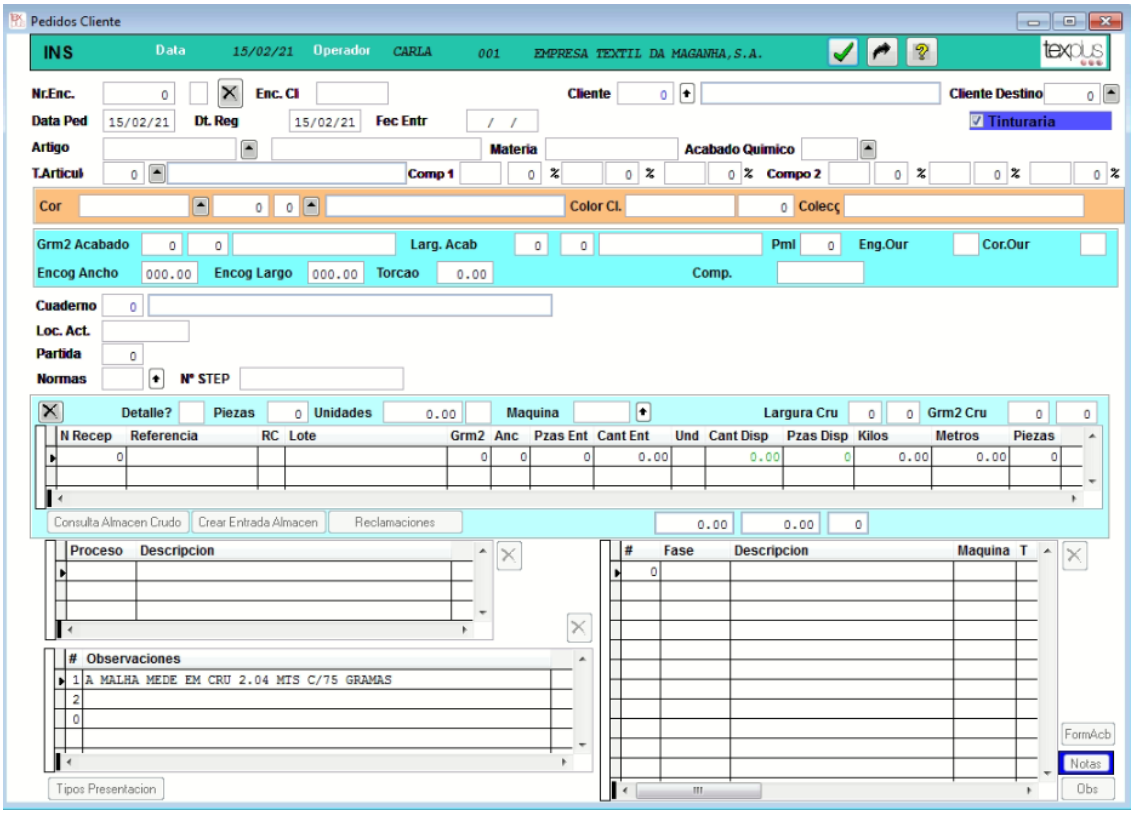
Wood, A. M. (2 de Outubro de 2020). *Visual FoxPro: You Need This Old School Programming Language In 2021.* Obtido de WhoIsHostingThis: https://www.whoishostingthis.com/resources/visual-foxpro/

Woods, E. (2011). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives.* Addison-Wesley.

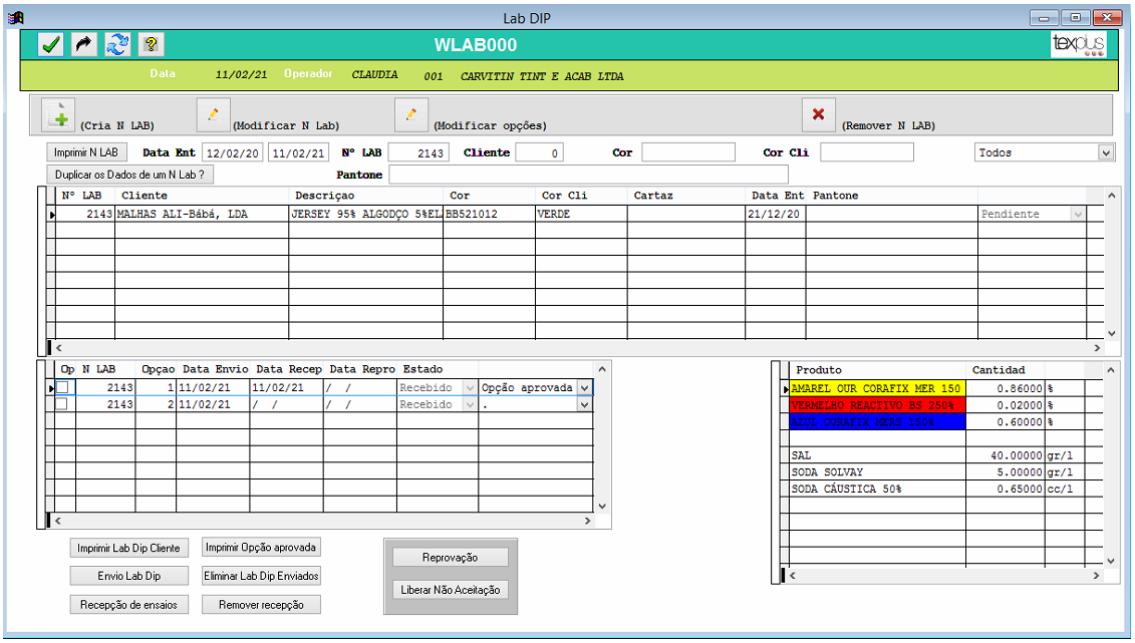
Zaforas, M. (09 de 01 de 2020). *Cassandra, the lady of NoSQL databases*. Obtido de en.paradigmadigital: https://en.paradigmadigital.com/dev/cassandra-lady-nosql-databases/

Anexo A: Interface Utilizador Texplus

Criação de Encomenda:

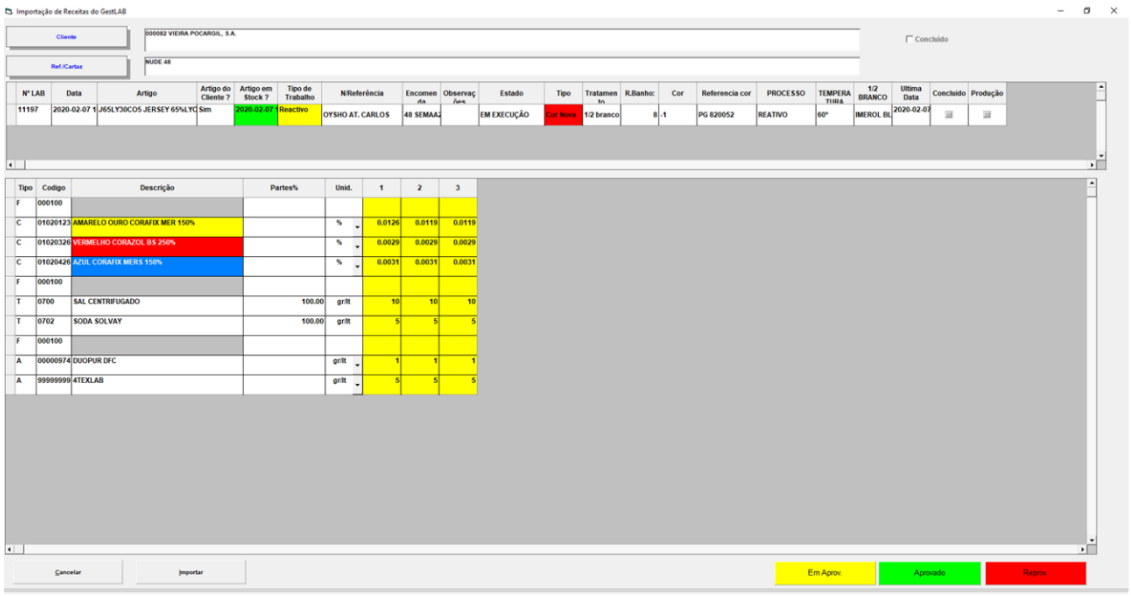


Gestão de Laboratório:



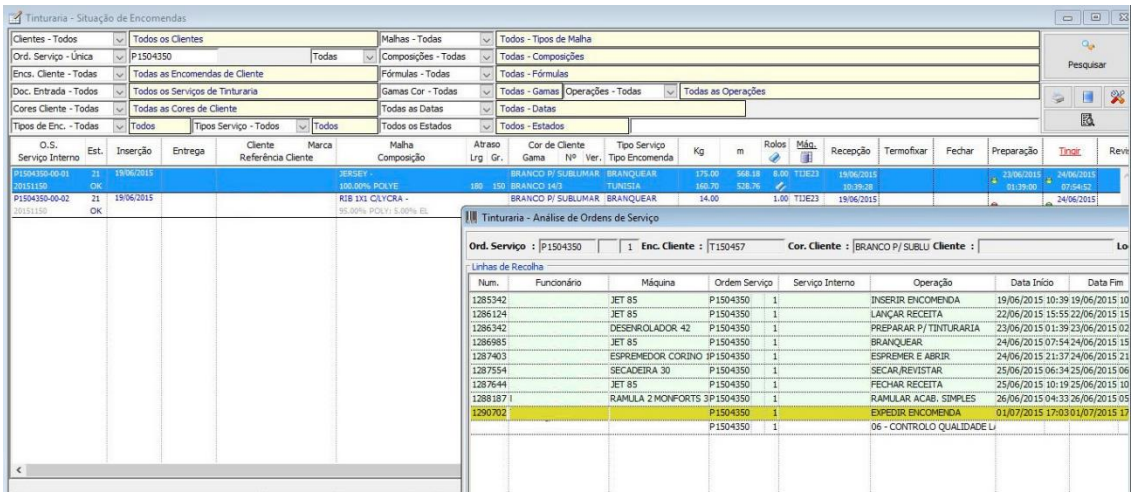
Anexo B: Interface Utilizador 4Tex

Gestão de Laboratório:



Anexo C: Interface Utilizador MacWin

Gestão de Encomendas:



Anexo D: Softwares de Gestão Têxtil Presentes na ITMA2019

NC STATE
Wilson College of Textiles

JTATM
Journal of Textile and Apparel, Technology and Management

Special Issue: ITMA, 2019

Supply Chain Management Software for Textile Networks at ITMA 2019

Kristin Thoney-Barletta
Wilson College of Textiles, NC State University
Raleigh, NC, USA

Introduction

At ITMA 2019, a variety of different types of textile software were showcased. There were many textile supply chain solutions among these. This paper focuses specifically on advanced planning and scheduling (APS) for supply chains. The software discussed in this paper was classified primarily in category 14.3.2, "Software systems for Supply Chain Management in textile networks". One of the nine companies listed in 14.3.2, Setex (https://www.setex-germany.com/), was not included. Although it seems to have good machine monitoring and predictive maintenance capabilities, the author determine that Setex is not an APS system.

All of the packages in 14.3.2 were also classified in category 14.3.3, "Software systems for Enterprise Resource Planning (ERP), Product Lifecycle Management (PLM) and Production Planning and Scheduling (PPS)", since most of the software reviewed could be classified as both ERP and PPS systems. Two of the companies listed in category 14.3.3 but not in 14.3.2 (Interlem GP Omega and Zeta Datatec) were also included in this paper because their software has similar capabilities as the software of the companies in 14.3.2. All of the companies discussed in this article, were also listed both in category 14.3.4, "Warehouse and logistics management systems", and 14.3.6, "Quality management systems", except Interlem GP Omega. PLM solutions were not specifically included, although some of the software reviewed in this article has this capability.

The software reviewed in this paper has a primary focus on the textile industry. This is in contrast to other APS systems on the market that do not necessarily focus on any specific industries. This focus on the textile industry can be a key advantage for a textile company looking for an APS system, since this often means that employees of the software company are much more familiar with textile processes. In addition, the software usually incorporates more of the functionality required to plan and schedule textile processes, which may result in less customization being necessary to successfully implement the system within a textile company. This can save a textile company significant money and time in getting the software fully operational.

The information in this article is based on discussions with software vendors at ITMA, promotional material that was handed out by them, and viewing the websites of the software companies. None of the software packages was tested. The companies are presented in the order in which they are listed in section 14.3.2 of the ITMA 2019 directory, followed by the two companies previously mentioned that are not in section 14.3.2. (When a company's website is in more than one language, the version in English is the one that is given.)

Computer House
Website: http://www.cho.it/index.php?ln=2
Computer House is an Italian company. Their software, PROTEX, contains an ERP solution with an Oracle database. The ERP solution consists of a sales module and a production module. Functionality within the production module includes cost control and quality control. Schedules are created that minimize a cost function. The user can assign different weights to different parts of this cost function, and the cost function can include meeting delivery dates. PROTEX also is a manufacturing execution system (MES), a warehouse management system (WMS), and a customer relationship management (CRM) system. It is not a PLM solution, but Computer House does collaborate with a specific PLM vendor if PLM functionality is needed by their clients. PROTEX can be used in spinning, knitting, warping, weaving, dyeing, finishing, and cut and sew operations. It has many applications in all segments of the textile industry, including apparel and home textiles.

According to the Computer House website, advantages of PROTEX ERP include that it is "developed entirely within Computer House", "totally integrated and easy to use", and is "multi-company and multi-language". The website also advertises that "This product is the existing most advanced on the market of textile scheduling, studied for years by the experience of our customers, [and] has 48 successful installations ranging in all areas of the textile industry."

Datamon
Website: http://datamonplus.com/, https://www.ruta4datamonplus.com/ (in Spanish)
Datamon Plus is a company based in Spain. Their primary software package is TEXPLUS. Their software is an ERP system that has costing and quality control. It plans production and performs scheduling with dispatching rules. The software has specialized solutions for spinning, weaving, dyeing, finishing, printing, and industrial laundries. The company currently sells their software in Spain, Portugal, and Latin America.

According to their website, Datamon Plus software offers control, improvement, and cost effectiveness. Its main benefits include increased profitability, immediate information for negotiations and decision making, inventory optimization, quick response to problems, and meeting customer quality specifications.

Datatex
Website: https://www.datatex.com/
Datatex has offices in Switzerland, Germany, India, Israel, Italy, and the United States. Their ERP system has sales and customer service (including CRM), planning and scheduling, production order management, quality management, inventory and warehouse management, purchasing, costing, and financial management functions. Their application-specific products include shop floor automation, fabric inspection, machine capacity management, machine queue management, and PLM. Datatex has many solutions including ones for fiber, spinning, knitting weaving, printing, dyeing, apparel, nonwovens, technical textiles, and carpet.

Shannon McCarthy, Vice President of Business Development at Datatex, said that are several advantages of Datatex. Datatex focuses exclusively on the textile industry (except for a few cases in which one of their textile customers has expanded into another industry), and the software has been developed for more than 20 years in response to their customers' needs. Not only does Datatex do detailed scheduling, it does load balancing among machines in different locations. According to the Datatex website, "Datatex is the world's leading supplier of IT software solutions to the global textile and apparel industry with the largest installed base of textile software with customers in 42 countries and 5 continents…Textile and apparel have very different and particular planning and scheduling requirements. Therefore the Datatex planning and the Datatex scheduling systems have proven to be one of the key success factors to our customers."

Inteos - Halo

Website: https://www.inteos.com/en/

Halo is an Austrian company. Its software package, Inteos, is an ERP, MES, and monitoring solution. Inteos also has quality control and warehouse management functions. In terms of planning and scheduling, it does capacity planning and reservation, rough and detailed planning, optimization, and simulation. It does not have PLM capabilities. Inteos can be used in weaving, knitting, finishing, printing, and apparel operations.

Stefano Sampietro, Salesman at Halo, said that he feels that that the main advantage of Inteos is that they are continuously developing and innovating. For example, they did not previously have tablet capability, and now they have it and are also moving into augmented reality. According to their website, Inteos provides "a product that can be integrated quickly and smoothly into current operations. The system is customised to suit your individual needs and the user-friendly and easy-to-understand platform allows for a minimum of training and guarantees maximum acceptance among users."

Intex

Website: https://www.intex-consulting.com

Intex is ERP and MES software. It is developed by Intex Consulting Group, a company based in Germany. Intex Consulting Group also has offices in other locations, including China, India, and Brazil. They partner with Oracle and also are a member of the SAP Extended Business Program. Modules within Intex include sales, production, inventory, purchasing, costing, and quality management. Scheduling is based on both meeting due dates and minimizing costs. Intex also contains a shipment model, and has CRM and PLM functionality. It allows the user to schedule sampling on machines and provides detailed tracking of sampling costs. Intex can be used in fiber, spinning, warping, weaving, dyeing, finishing, and cut and sew operations. Intex has been implemented in a variety of industries, including apparel and home furnishings.

According to Dileep Kumar, Senior Consultant at Intex Consulting Group, there are many advantages of Intex. It has integrated planning across the system and is a finite capacity scheduler. Actual costing is performed throughout the supply chain. Intex enables users to estimate accurate delivery dates during the course of a phone conversation and suggests similar products if the desired product cannot be produced on time. It also allows companies to determine who their best customers are from a profit point of view. According to the Intex website, "Including our co-operation partners and our international subsidiaries, more than 100 textile and software specialists are selling and implementing our solutions worldwide."

Porini

Website: https://www.porini.it/

Porini is a company based in Italy. Porini distributes its software through a channel of partners, including Arquiconsult. The Porini Suite for Microsoft Dynamics 365 consists of Porini 365 ERP, Porini 365 CRM, Porini 365 Apps, and Porini Analytics. Porini 365 CRM is a solution for fashion and retailing. Porini 365 Apps is for retail and includes point of sale (POS) management. Porini 365 ERP is for fashion, retail, footwear, and textiles. It has functionality for spinning, weaving, knitting, dyeing, finishing, printing, and cut and sew. Porini 365 ERP can be also used in technical textiles and geo-textiles. The scheduling module calculates requirements based on due dates and process constraints. The system suggests a schedule based on one of several different algorithms, including first in, first out (FIFO) and priority schemes. Porini takes into account quality control, and it blocks work in process (WIP) from advancing on its route and finished goods from shipping to the customer, if quality specifications are not met. The software has MES and warehouse management capabilities. Porini also has PLM software.

According to Luis Fernandes, Vertical Areas Lead at Arquiconsult, a major

advantage of Porini is its relationship with Microsoft. The software is cloud-based with a subscription fee and is accessed through browsers. Updates occur seamlessly. Porini integrates easily with Microsoft PowerApps, Flow, and PowerBI, as well as the standard Office suite. The Porini website states that they have 150+ projects and 25,000 active users.

Schaeffer Productique

Website: http://www.schaeffer-productique.com

Schaeffer Productique is a French company. Their software includes the ERP/CRM SOLINSyst and has capabilities in inventory management, costing, scheduling, quality control, and logistics. It can be used in knitting, weaving, dyeing, and cut and sew operations. It has been implemented in many industries, including apparel and home textiles. The software is used primarily in Europe. However, there are also some implementations in North Africa. The software is not currently used in China or India.

According to the Schaeffer Productique website, "…Schaeffer Productique is today the European leader in its sector. We integrate the changing needs of the International Textile Industry with the latest advances in digital technologies to support our Customers in their evolution. We regularly bring them the most relevant solutions, at the best conditions." Schaeffer Productique has 200 customers in 12 countries.

Up Solutions

Website: https://www.just-mes.com/en/

Up Solutions is a company based in Italy. Their Just MES software is not an ERP system, but is compatible with many ERP systems. Up Solutions is a Microsoft partner. Just MES consists of the just suite, which has the following four integrated solutions: just plm, just planning, just monitoring, and just quality control. There is one just suite for fashion, and a separate one for textiles and technical textiles. Companies only have to purchase the integrated solutions that they need. For example, carpet manufacturers often only use the just monitoring textile solution. With the textile just planning solution, there are specific programs for spinning, knitting, weaving, and dyeing. The just planning solution also has warehouse management functionality. Just MES does not yet have a program for cut and sew operations. The just planning solution incorporates the option to source production or schedule production. When the user decides to schedule it, they select the priority within scheduling heuristics. Just MES has been implemented in such industries as home textiles, apparel, shoes, and carpet. Most recently, it has be used in the automotive industry.

In describing Just MES, Up Solutions's website says that "… we have now achieved a fundamental goal for us: to offer a complete solution capable of managing and monitoring the entire manufacturing process of the company…Thanks to our consolidated experience in the textile and apparel sector, we are now able to satisfy all those production companies that need to monitor and improve their supply chain." Just MES currently has 160 customers in 30 countries.

Interlem GP Omega

Website:
https://interlemgpomega.it/?lang=en

Interlem GP Omega is an Italian company whose main focus is on small and medium sized enterprises. Their products include the Arianna Textile ERP solution, the Arianna Printing ERP solution for digital printing, and the Net@Pro MES solution. The software has functionality to be used in spinning, warping, weaving, dyeing, printing, and finishing. It is not currently for cut and sew operations. The software has been used for many end applications, including apparel, home textiles, and making carbon fiber. The software has finite capacity scheduling, and the user puts weights on different factors, including due dates and costs. It has CRM and WMS capabilities with order picking. It does not have PLM functionality. Interlem GP Omega sells primarily to the European

market. Its ERP is generally not sold outside of Italy.

Andrea Picone, Managing Partner at Interlem GP Omega, said that one key advantage of their system is that it is more flexible than others on the market. The Interlem GP Omega website says Net@Pro has more than 16,000 users in Italy and 200 user companies. According to the website, the benefits of using the software include speed improvement, production lead time reduction, reduction of waste, completeness of deliveries, punctuality, and resource optimization.

LOOMDATA (Zeta Datatec)

Website: https://en.zetadatatec.com

Zeta Datatec is a company based in Switzerland. Their software, LOOMDATA, is both an ERP and MES system. They deliver both the hardware and software. The system does plant monitoring, quality control, and warehouse tracking. It also can be used for planning. LOOMDATA did not historically have a lot of automated scheduling functionality. However, one of their new features is forward and backward scheduling of orders. Furthermore, the company is willing to customize their software for clients who require more scheduling functionality. Their software is specifically for weaving, knitting, and finishing. It is not used for spinning or sewing operations. The software also does not have PLM capabilities. Industries Zeta Datatec serve include textiles, technical textiles, plastics, pharmaceuticals, cosmetics, food, and paper.

Werner Zberg, Zeta Datatec Co-Owner, said that there are a lot of advantages to using LOOMDATA. He believes that LOOMDATA can more quickly integrated into companies. The software runs on an HP server with Sun Solaris, and he believes that this is a more stable environment. According to the Zeta Datatec website, "Having installed more than 550 systems in 50 different countries around the world, our over 35 years of experience in providing powerful monitoring and planning solutions for the textile industry has given us the ability to produce the most stable and reliable system in the world today."

Discussion

There are a lot of options available when choosing a supply chain APS system. Among the software discussed in this paper, there are significant variations in the software capabilities, types of textiles processes for which the software system was designed, and the final products of textile companies in which the software has been implemented. In addition, some of the software companies have a strong international presence, while others have focused on a particular region. Some of these regional companies may be looking for new opportunities to expand, while others may be content to be regional players.

Before evaluating different supply chain APS software, textile companies should

- clearly define the benefits that they expect to achieve from implementing the new software,
- determine how they would like the software to integrate with and/or replace existing software,
- decide how they would like the software to fit into their existing business procedures,
- and agree on the essential software functions necessary to achieve the desired benefits based on their business procedures.

This information, combined with textile company product and process details, can be used to narrow down which commercial packages seem to best match the company's requirements.

The remaining supply chain APS software can be further evaluated based on a more detailed assessment of both software capabilities and whether company constraints can be met. In terms of software capabilities, having an APS system that automatically generates detailed finite capacity schedules and outputs dispatching lists for individual machines might be very important to a company. If that is the case, can the software

provide that capability? If so, how are those schedules generated? Textile companies should also develop a comprehensive list of business and manufacturing constraints, in conjunction with production managers and machine operators. They should find out which if these constraints can be modeled in the base software system and which would necessitate customization. In addition to evaluating commercial APS systems, textiles companies should consider the alternatives of developing software in house and/or hiring a consultant experienced in supply chain planning and scheduling to develop specialized software. A cost-benefit analysis of each acceptable option should be performed based on a detailed cost breakdown and a well thought-out estimate of the anticipated financial benefits.

Conclusions

Many APS software packages for textile supply chains were presented at ITMA 2019. By carefully analyzing textile company requirements and constraints, in conjunction with software benefits and costs, textile companies can make the most informed decision as to whether one of these systems, a generic (non textile specific) APS system, developing software in house, or hiring an experienced consultant is the best decision for the textile company.

Anexo E – Ensaios pertencentes a um Cartaz enviados para aprovação

COLOR LAB DIPS

Ref: 08-0363/

To: _____

We are herewith submitting color lab dips as per your request for your approval.Please advise your approval and/or comments the soonest.

Date: 2008.1.14

Fabric				
Desc. / Color	Original	A	B	C
Grey flannel				
	Original	A	B	C
Navy				
	Original	A	B	C

SAMPLE

Anexo F – Descrição e Design dos User Stories

Tabela 22 – Descrição individual dos Casos de Uso

UC1: Gerir Cartaz

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC1.1)	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de cartazes para efetuar as operações CRUD da entidade cartaz.
	Consultar (UC1.2)	
	Editar (UC1.3)	
	Eliminar (UC1.4)	

UC2: Gerir Cor

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC2.1)	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de cores para efetuar as operações CRUD da entidade cor.
	Consultar (UC2.2)	
	Editar (UC2.3)	
	Eliminar (UC2.4)	

UC3: Gerir Tom

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC3.1)	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de tons para atribuir/desatribuir uma cor a um cartaz.
	Consultar (UC3.2)	
	Editar (UC3.3)	
	Eliminar (UC3.4)	

UC4: Gerir Intensidade

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC4.1)	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de intensidades para efetuar as operações CRUD da entidade ensaio.
	Consultar (UC4.2)	
	Editar (UC4.3)	
	Eliminar (UC4.4)	

UC5: Gerir Ensaio

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC5.1)	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de ensaios para
	Consultar (UC5.2)	
	Editar (UC5.3)	

	Eliminar (UC5.4)	efetuar as operações CRUD da entidade ensaio.

UC6: Gerir Grupos de Corantes

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC6.1)	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de cartazes para atribuir/desatribuir uma cor a um cartaz.
	Consultar (UC6.2)	
	Editar (UC6.3)	
	Eliminar (UC6.4)	

UC7: Gerir Tabela de Produtos Auxiliares

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC7.1)	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de produtos para efetuar as operações CRUD da entidade produto.
	Consultar (UC7.2)	
	Editar (UC7.3)	
	Eliminar (UC7.4)	

UC8: Gerir Tipo de Tingimento

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC8.1)	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de cores para efetuar as operações CRUD da entidade tipo tingimento.
	Consultar (UC8.2)	
	Editar (UC8.3)	
	Eliminar (UC8.4)	

UC9: Importar Leitura de Ensaio

Ator	Descrição
Gestor de Laboratório	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de cores para importar um ensaio para uma determina cor.

UC:10 Exportar Fórmula de Ensaio

Ator	Descrição
Gestor de Laboratório	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de ensaios para exportar dados do tipo da entidade ensaio.

UC:11 Calcular Preço de Ensaio

Ator	Descrição
Gestor de Laboratório	O utilizador registado no sistema com permissões de gestor de laboratório acede ao ecrã de gestão de ensaios para efetuar o cálculo do preço estimado de um determinado ensaio.

UC12: Gerir Entidade

Ator	Operações	Descrição

Gestor de Laboratório	Criar (UC12.1)	O utilizador registado no sistema com permissões de gestor de sistema acede ao ecrã de gestão de entidades para efetuar as operações CRUD da entidade tipo entidade.
	Consultar (UC12.2)	
	Editar (UC12.3)	
	Eliminar (UC12.4)	

UC13: Gerir Departamento

Ator	Operações	Descrição
Gestor de Laboratório	Criar (UC13.1)	O utilizador registado no sistema com permissões de gestor de sistema acede ao ecrã de gestão de departamentos para efetuar as operações CRUD.
	Consultar (UC13.2)	
	Editar (UC13.3)	
	Eliminar (UC13.4)	

UC14: Gerir Produto Químico

Ator	Operações	Descrição
Gestor de Químicos	Criar (UC14.1)	O utilizador registado no sistema com permissões de gestor de químicos acede ao ecrã de gestão de entidades para efetuar as operações CRUD da entidade tipo produto.
	Consultar (UC14.2)	
	Editar (UC14.3)	
	Eliminar (UC14.4)	

UC15: Gerir Notificações Stocks

Ator	Operações	Descrição
Gestor de Químicos	Criar (UC15.1)	O utilizador registado no sistema com permissões de gestor de químicos acede ao ecrã de gestão de produtos para ativar/desativar os vários tipos de notificação de stocks.
	Consultar (UC15.2)	
	Editar (UC15.3)	
	Eliminar (UC15.4)	

UC16: Gerir Certificação

Ator	Operações	Descrição
Gestor de Químicos	Criar (UC16.1)	O utilizador registado no sistema com permissões de gestor de químicos acede ao ecrã de gestão de certificações para efetuar as operações CRUD da entidade tipo certificação.
	Consultar (UC16.2)	
	Editar (UC16.3)	
	Eliminar (UC16.4)	

UC17: Gerir Utilizador

Ator	Operações	Descrição

Gestor de Sistema	Criar (UC16.1)	O utilizador registado no sistema com permissões de gestor de químicos acede ao ecrã de gestão de utilizadores para efetuar as operações CRUD da entidade tipo utilizador.
	Consultar (UC16.2)	
	Editar (UC16.3)	
	Eliminar (UC16.4)	

UC1.1 Criar Cartaz

Diagrama de sequência do sistema:

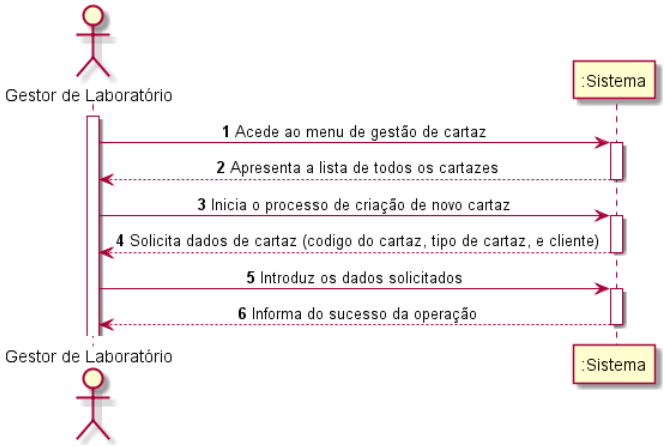


Figura 71 – Diagrama de sequência do sistema do UC1.1

Diagrama de sequência:

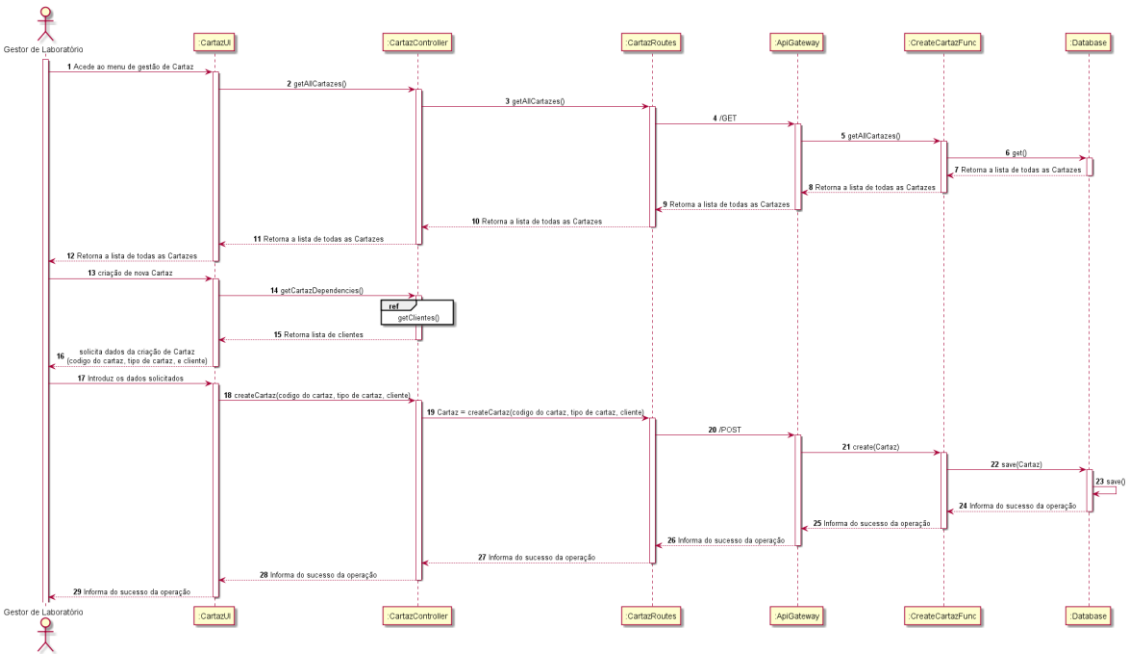


Figura 72 – Diagrama de sequência do UC1.1

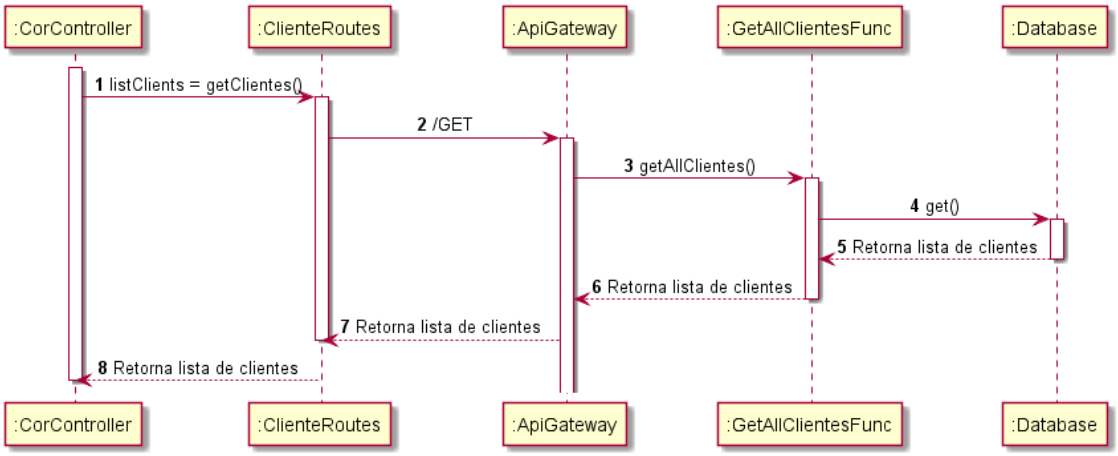


Figura 73 – Diagrama da dependência getClientes do UC1.1

UC1.2 Consultar Cartaz

Diagrama de sequência do sistema:

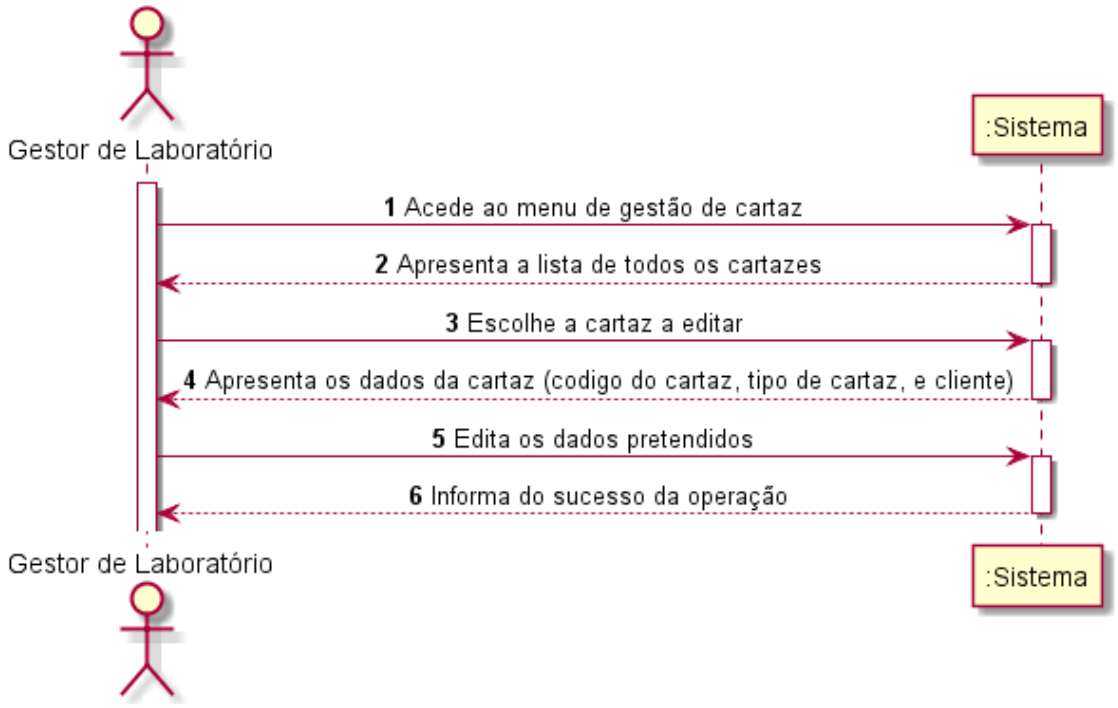
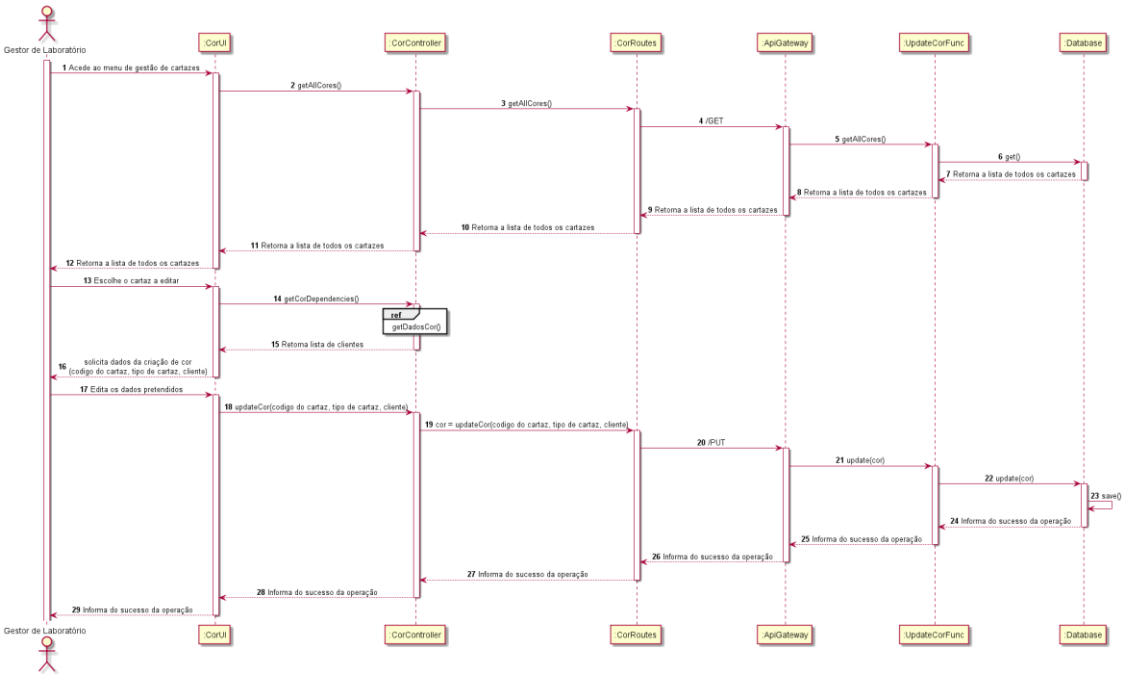


Diagrama de sequência:



UC1.3 Editar Cartaz

Diagrama de sequência do sistema:

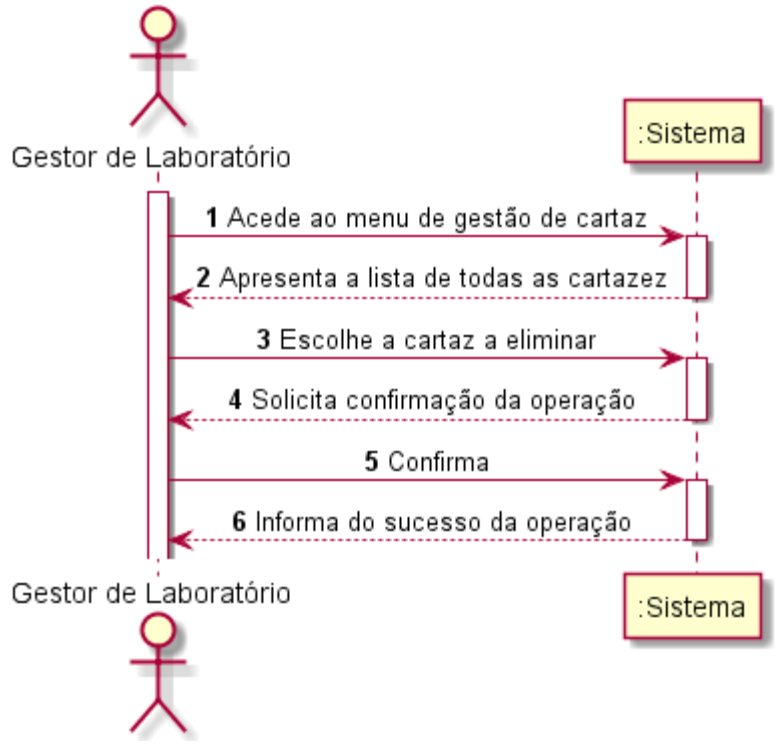
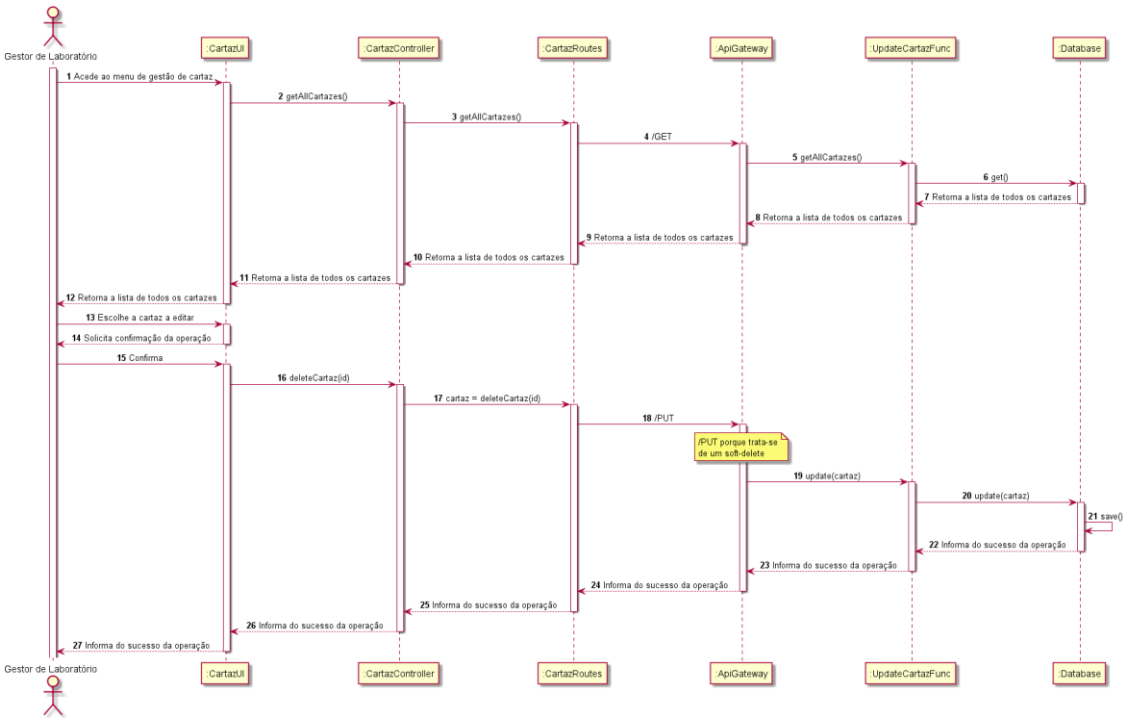


Diagrama de sequência:



UC1.4 Eliminar Cartaz

Diagrama de sequência do sistema:

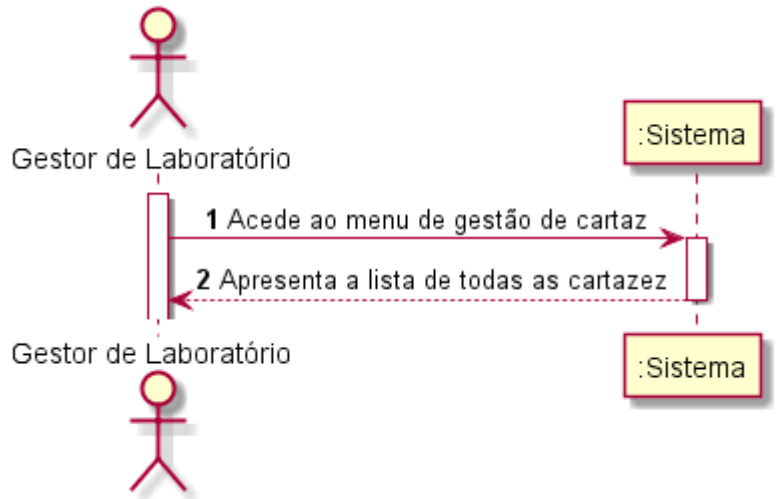
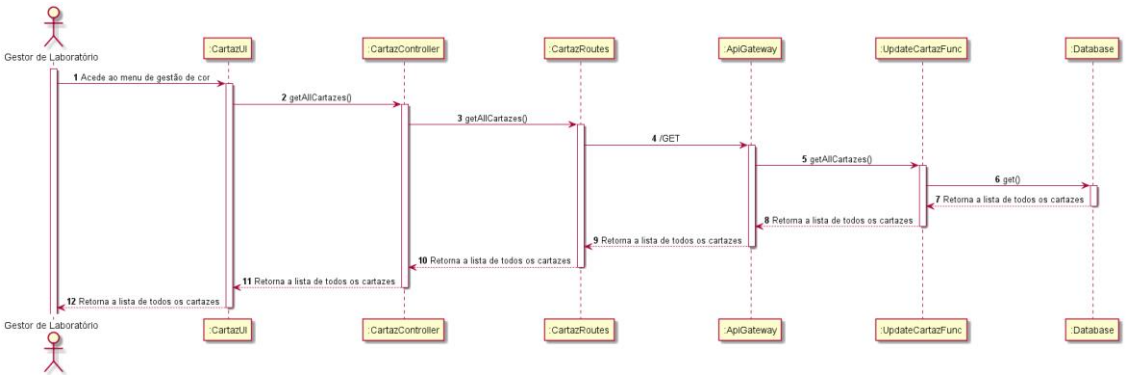


Diagrama de sequência:



UC2.1 Criar Cor

Diagrama de sequência do sistema:

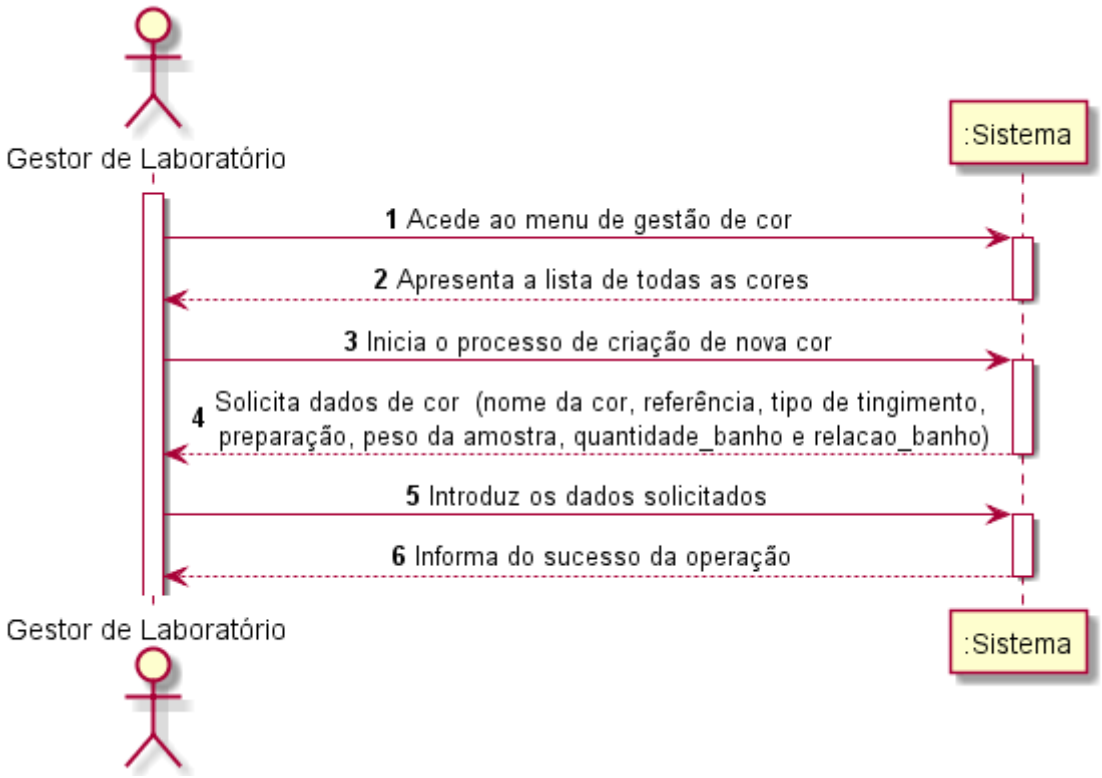


Diagrama de sequência:

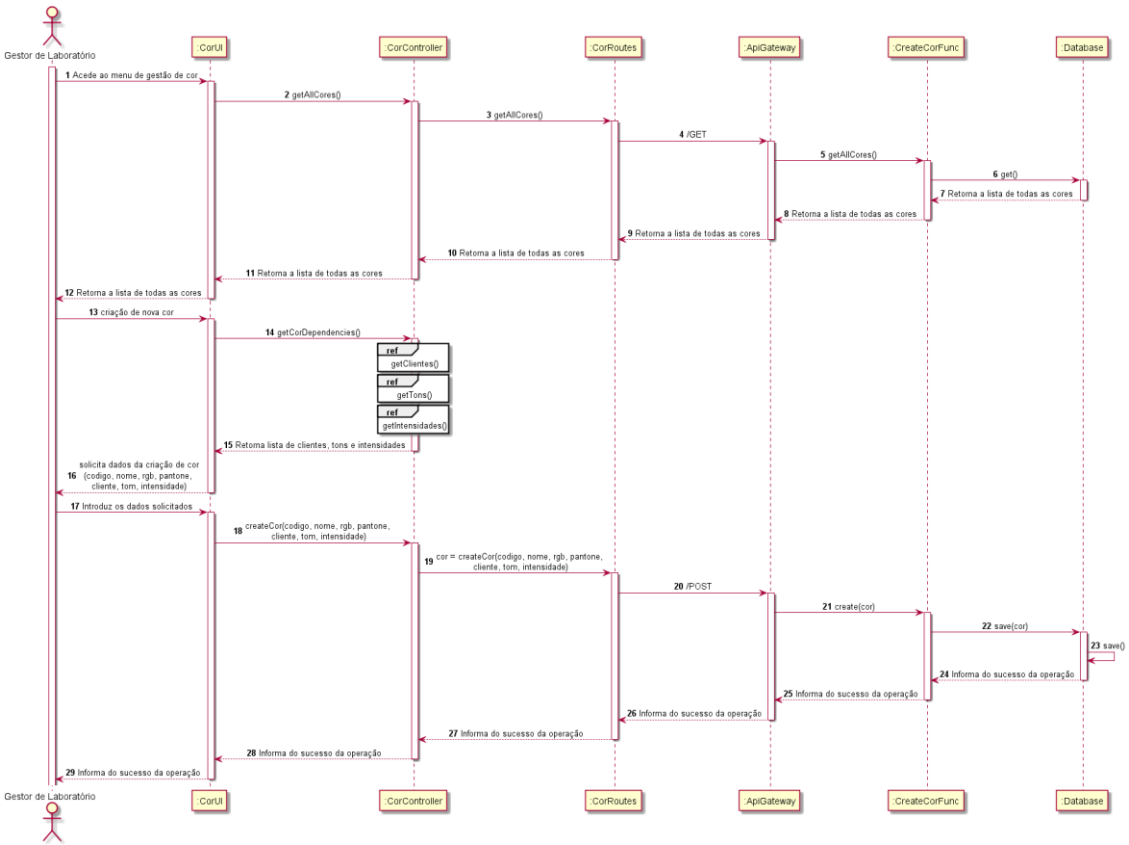


Diagrama da ação get:

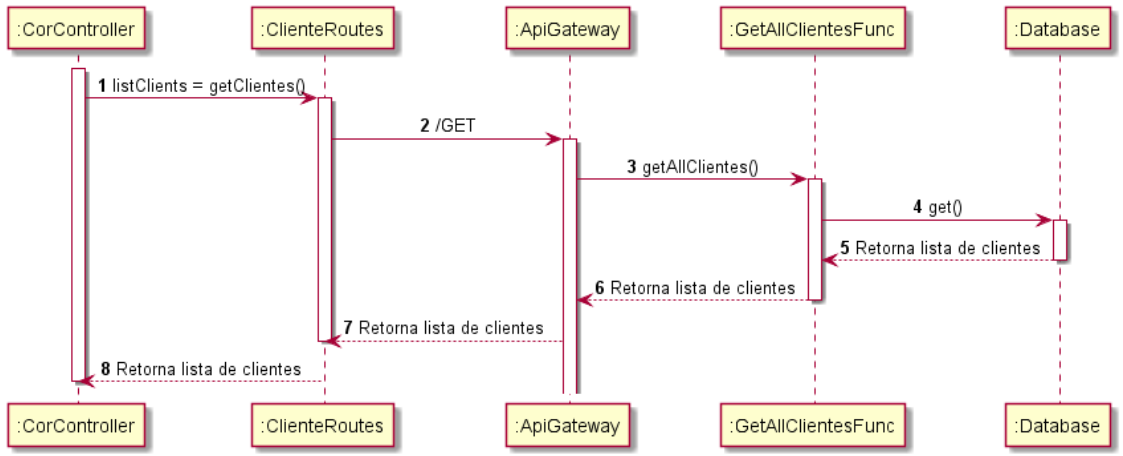


Diagrama da ação get:

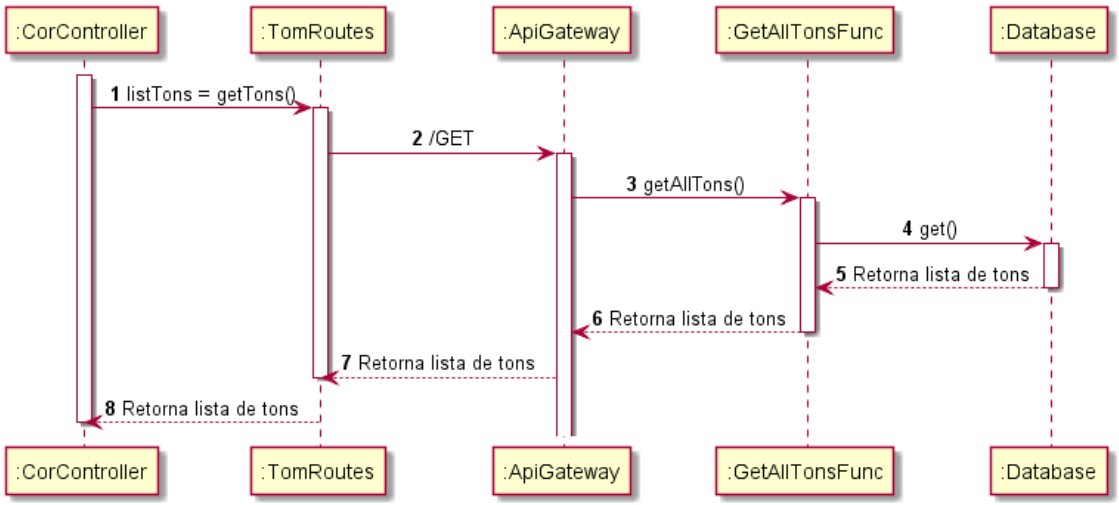
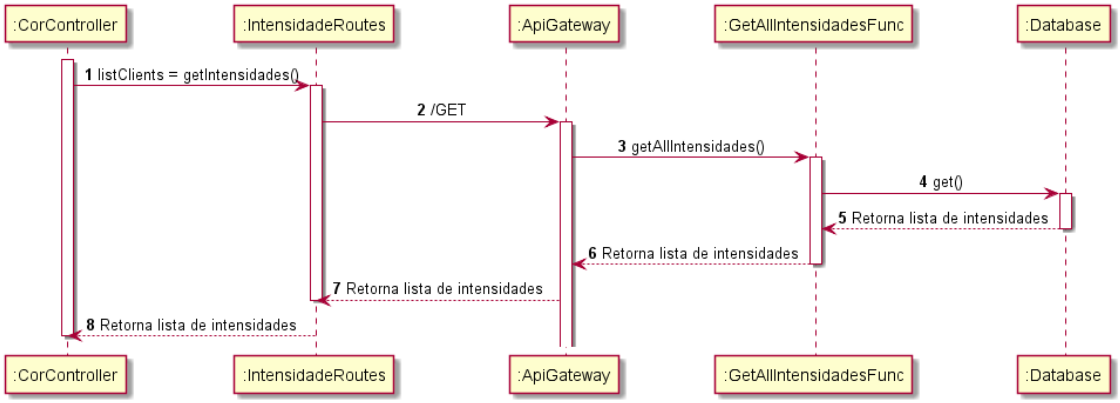


Diagrama da ação get:



UC2.2 Consultar Cor

Diagrama de sequência do sistema:

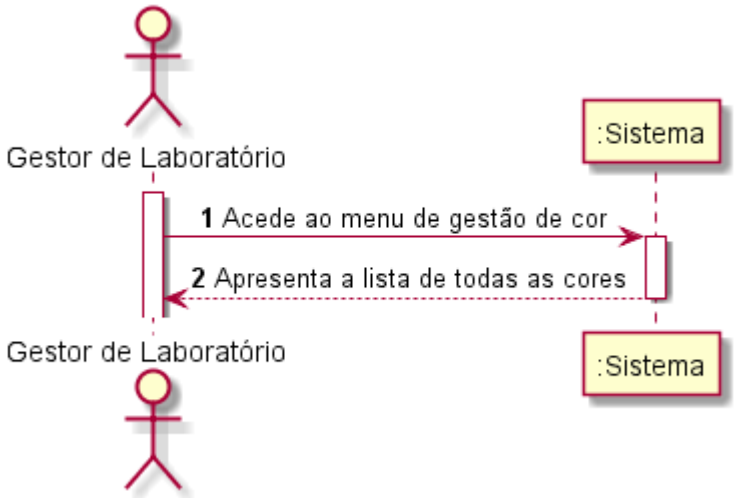
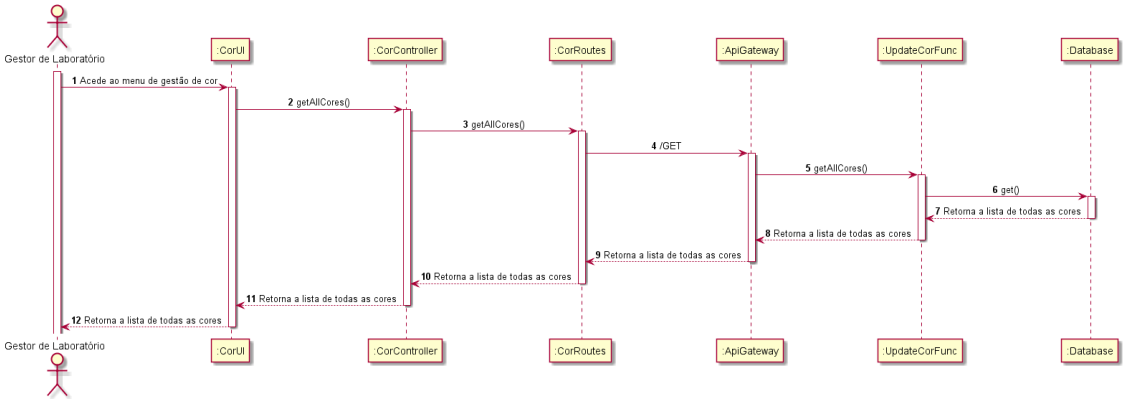


Diagrama de sequência:



UC2.3 Editar Cor

Diagrama de sequência do sistema:

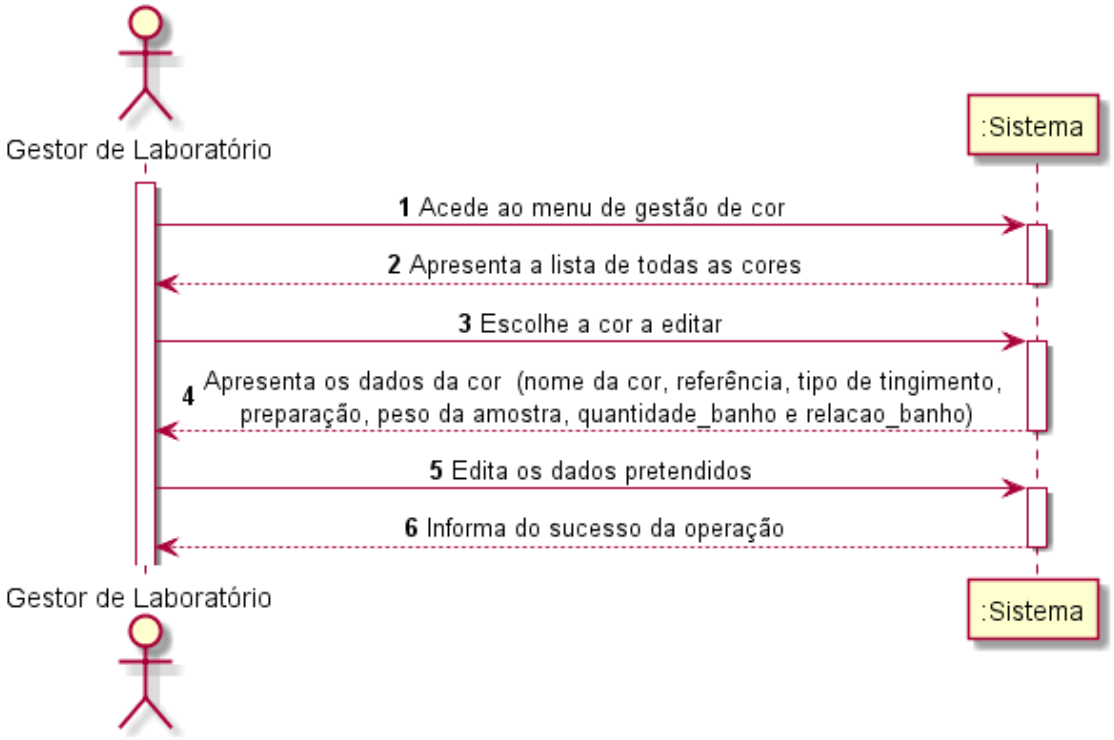
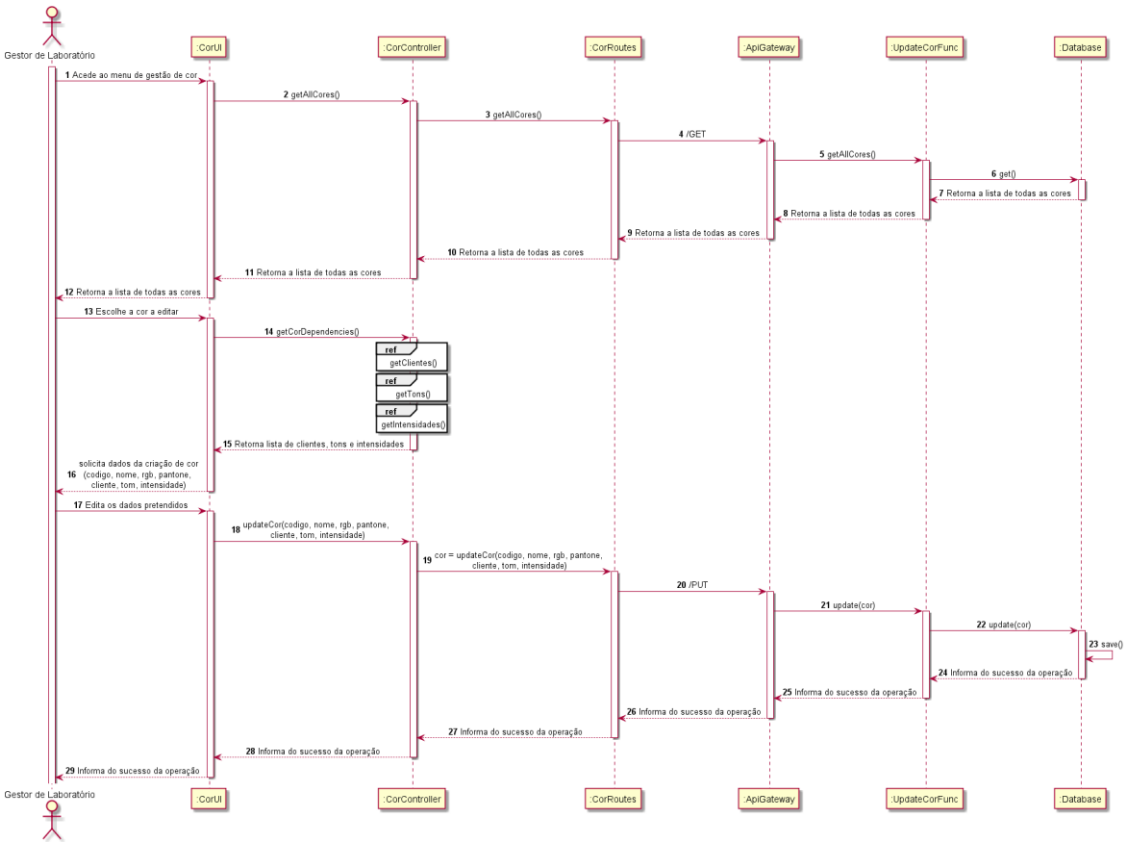


Diagrama de sequência:



UC1.4 Eliminar Cor

Diagrama de sequência do sistema:

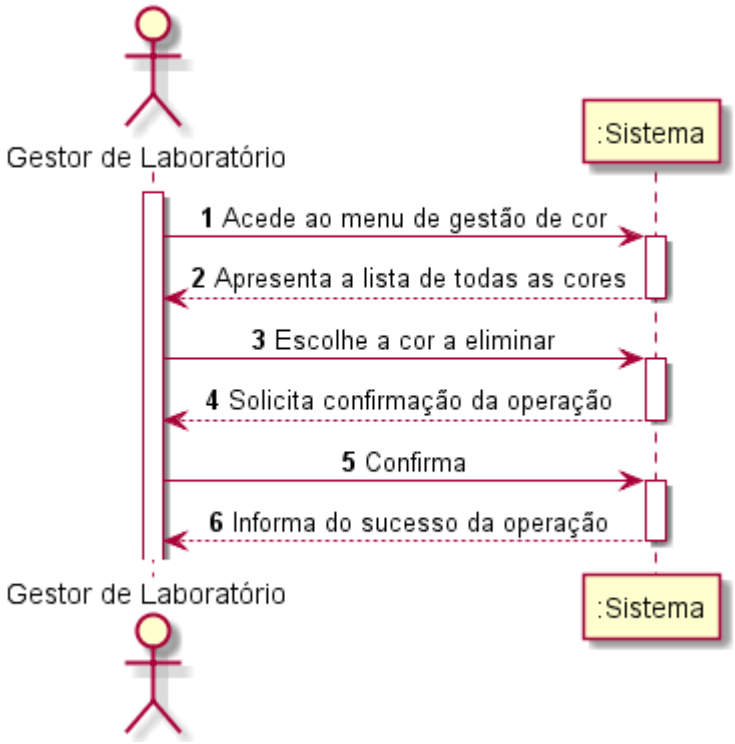
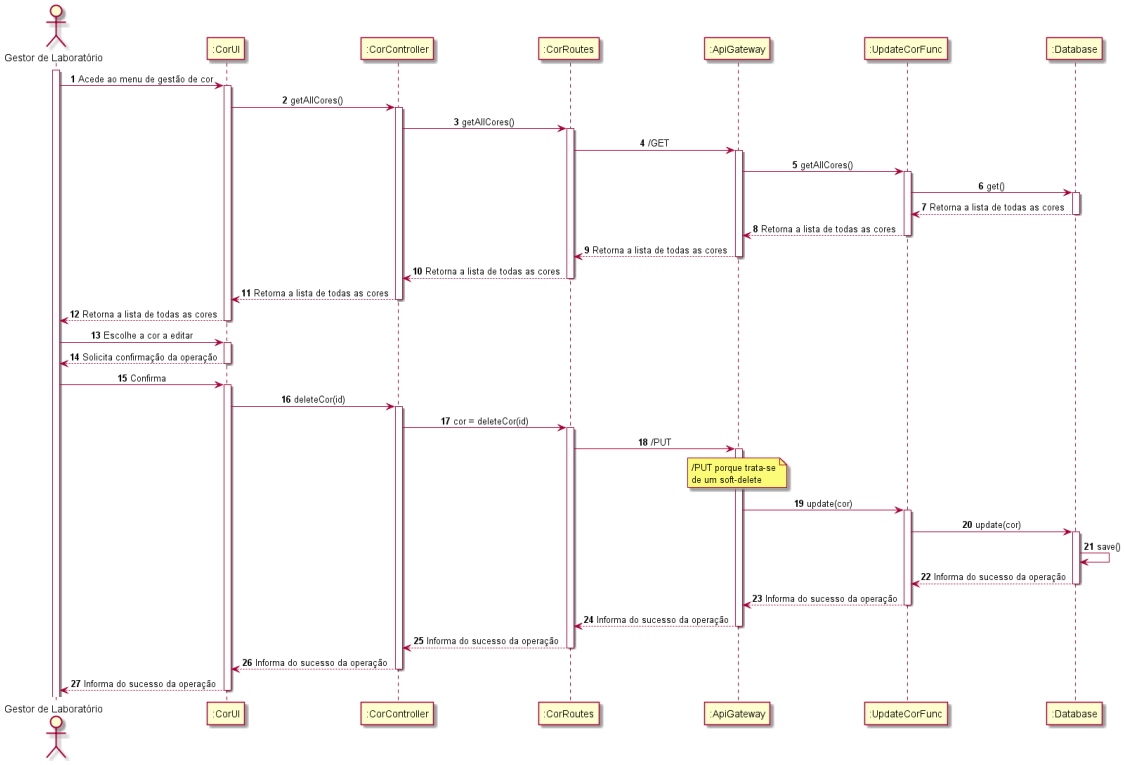


Diagrama de sequência:



UC3.1 Criar Tom

Diagrama de sequência do sistema:

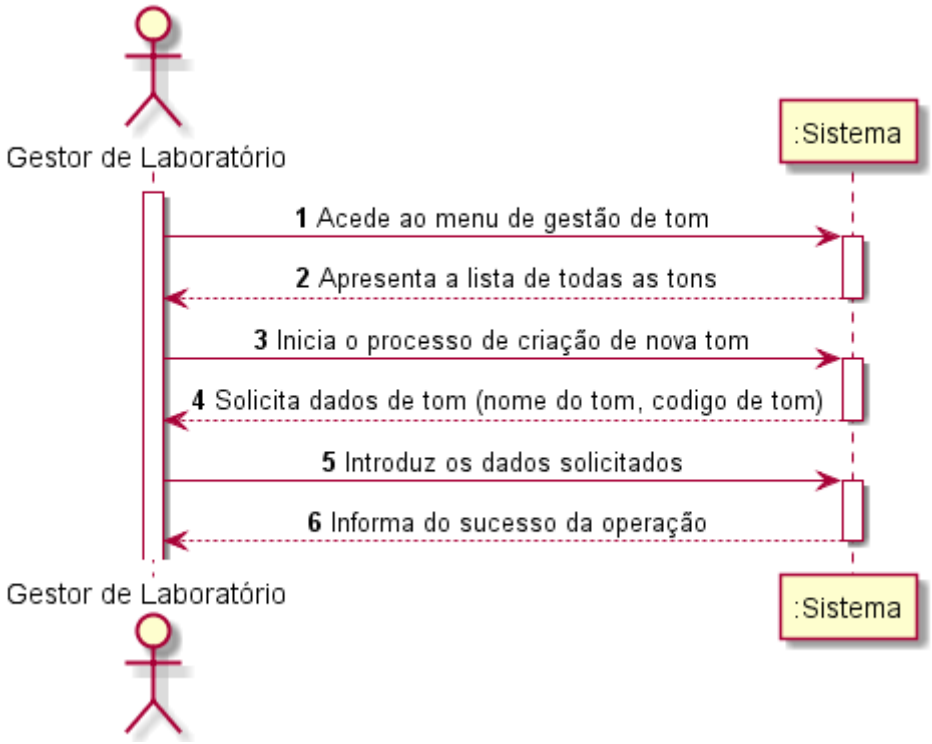
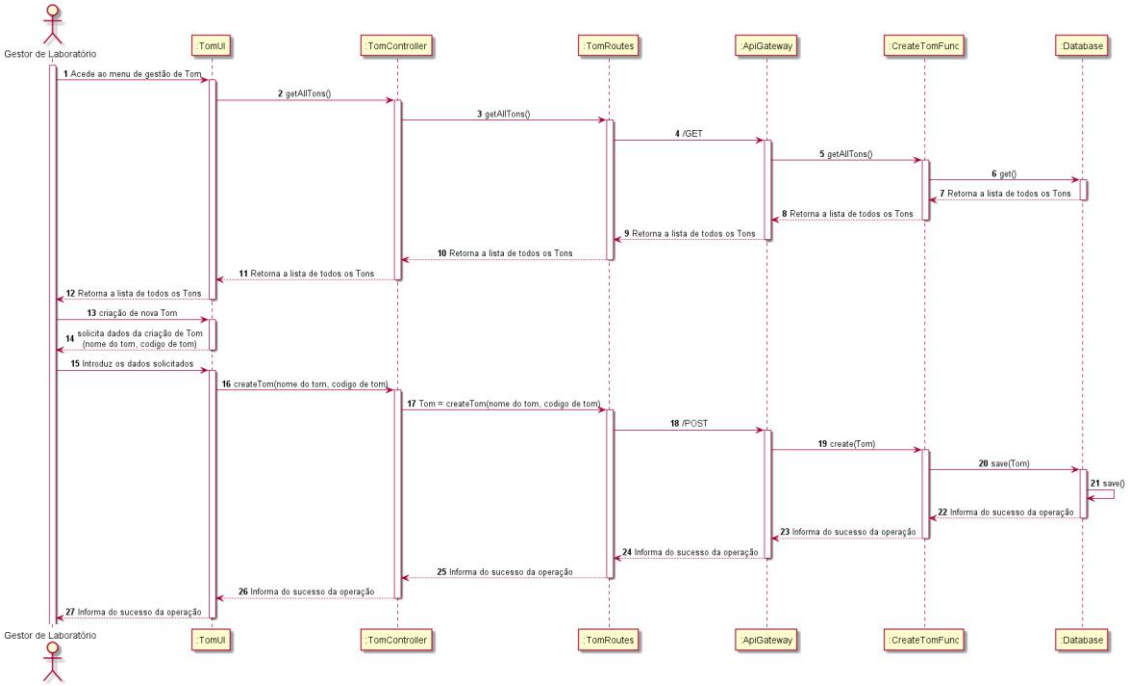


Diagrama de sequência:



UC3.2 Consultar Tom

Diagrama de sequência do sistema:

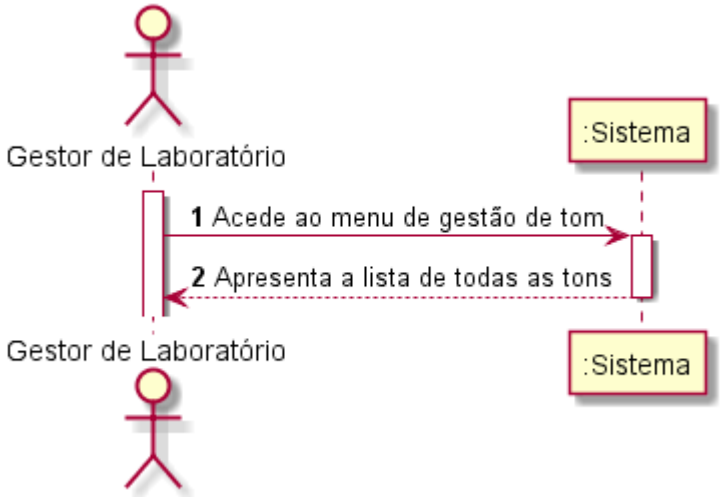
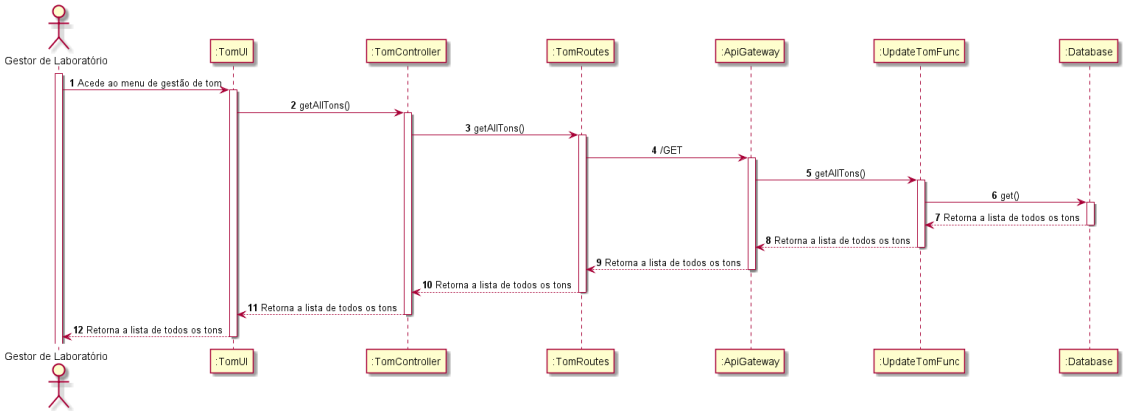


Diagrama de sequência:



UC3.3 Editar Tom

Diagrama de sequência do sistema:

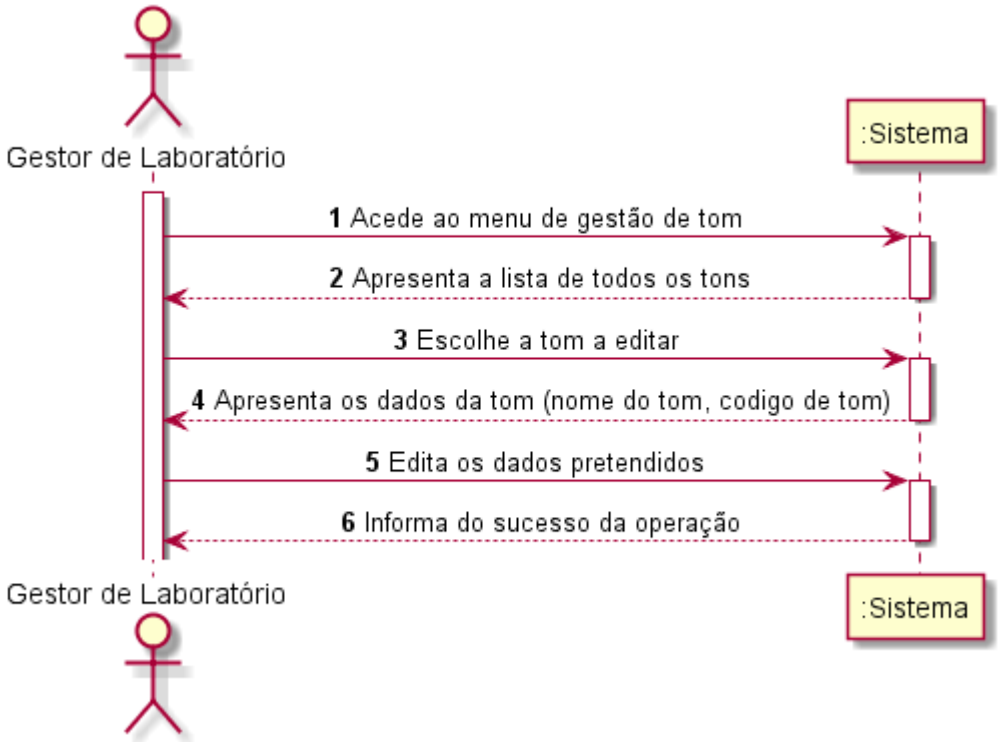
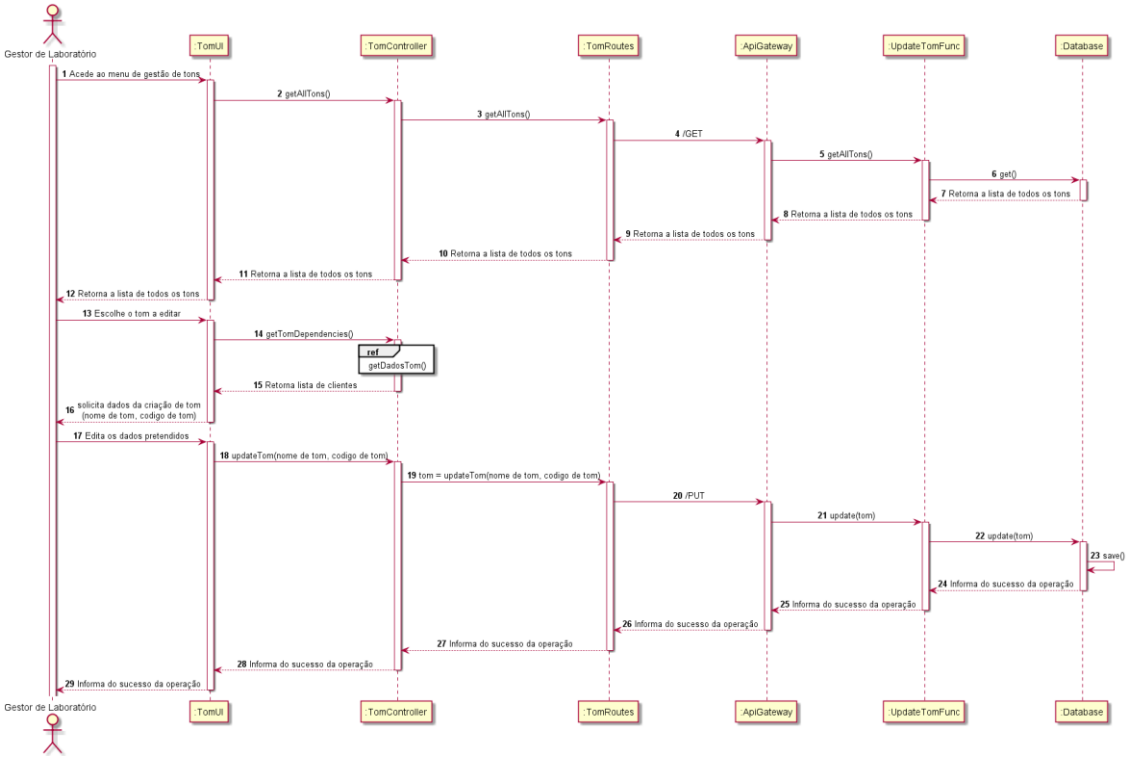


Diagrama de sequência:



UC3.4 Eliminar Tom

Diagrama de sequência do sistema:

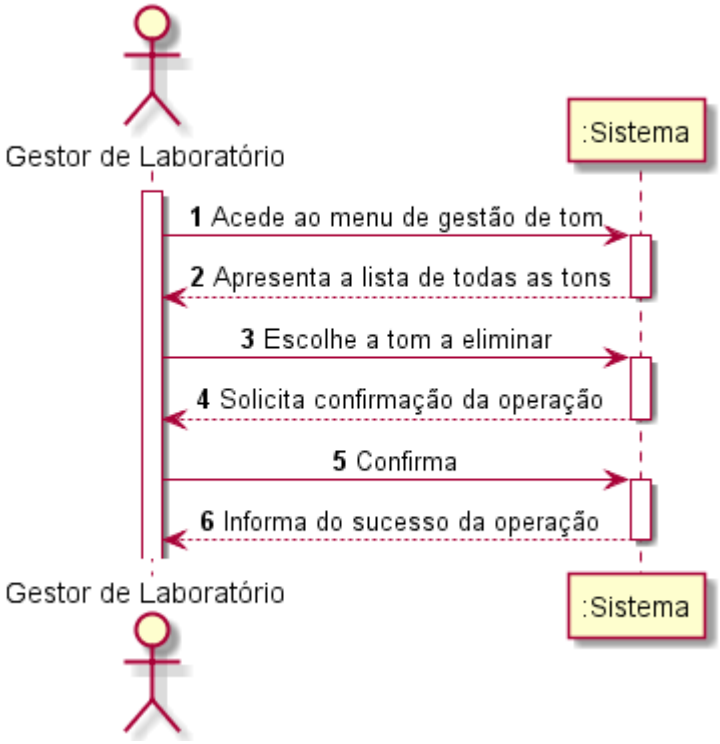
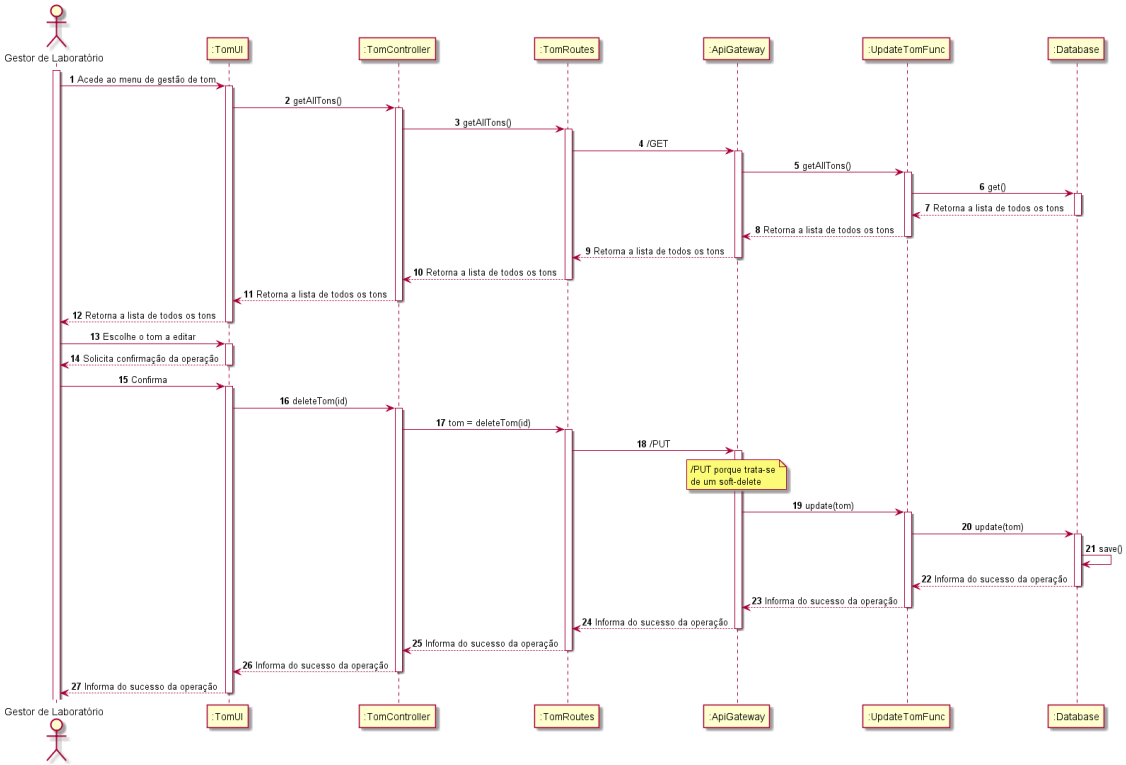


Diagrama de sequência:



UC4.1 Criar Intensidade

Diagrama de sequência do sistema:

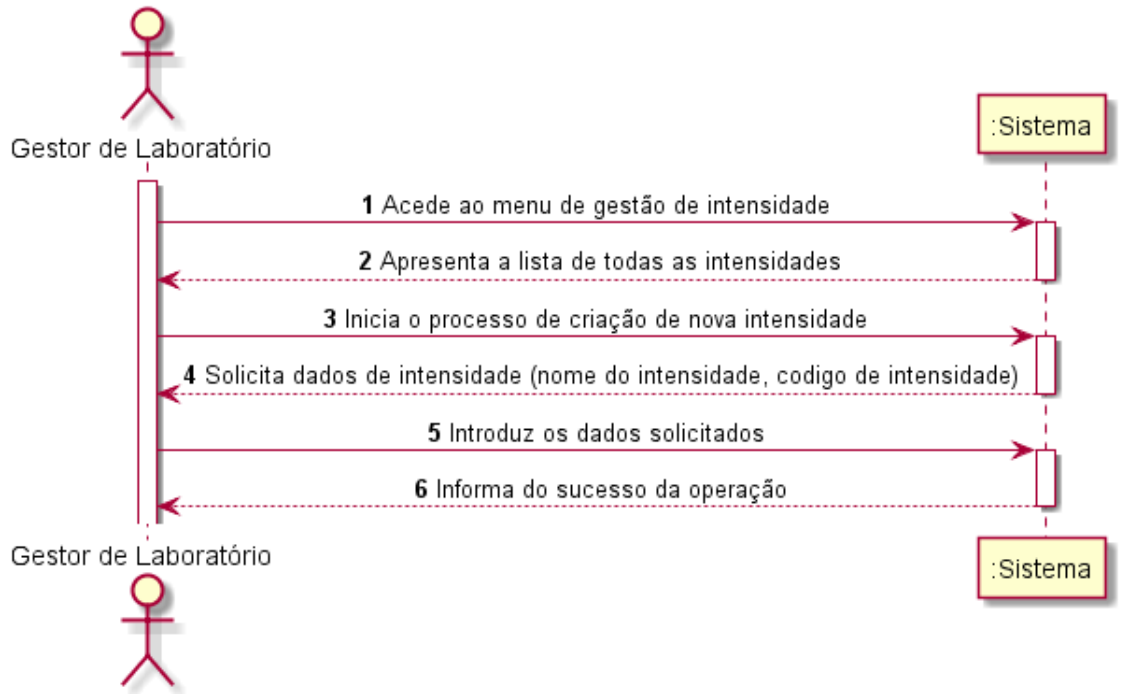
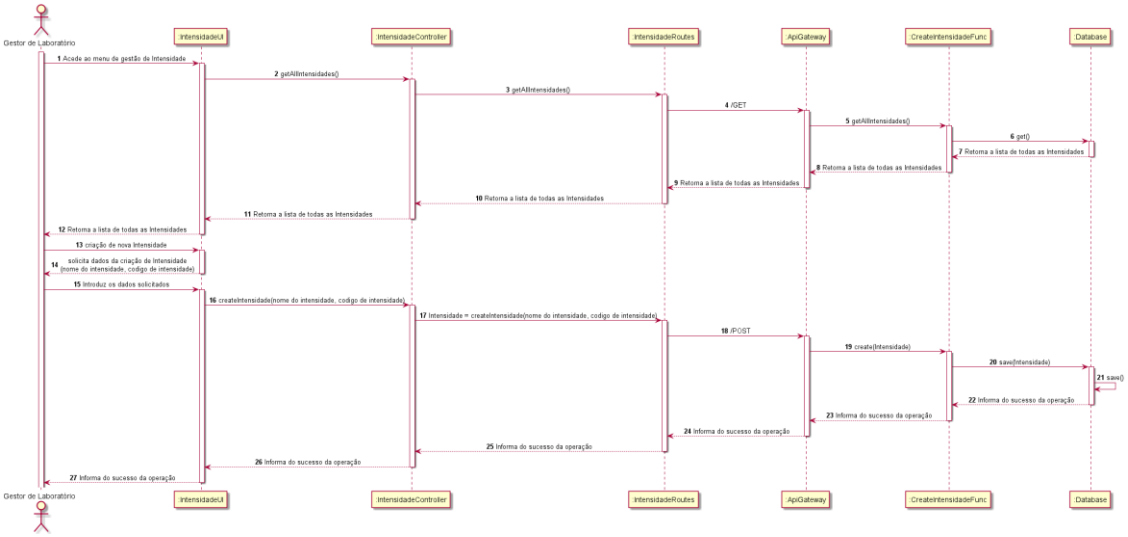


Diagrama de sequência:



UC4.2 Consultar Intensidade

Diagrama de sequência do sistema:

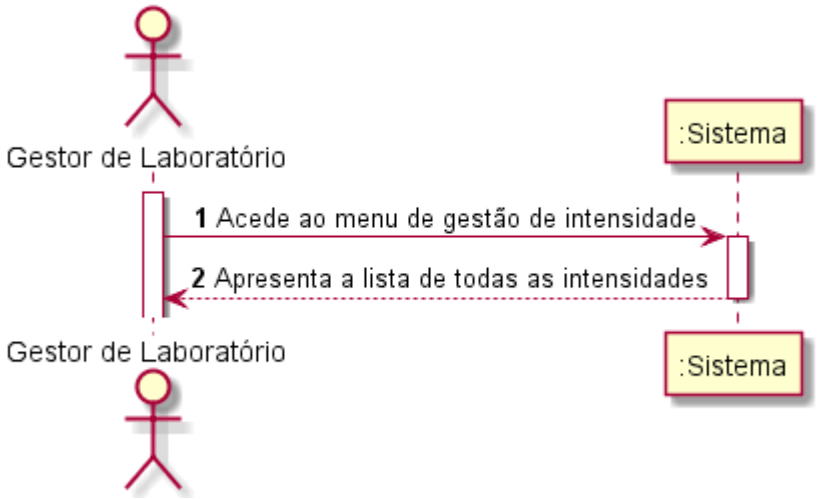
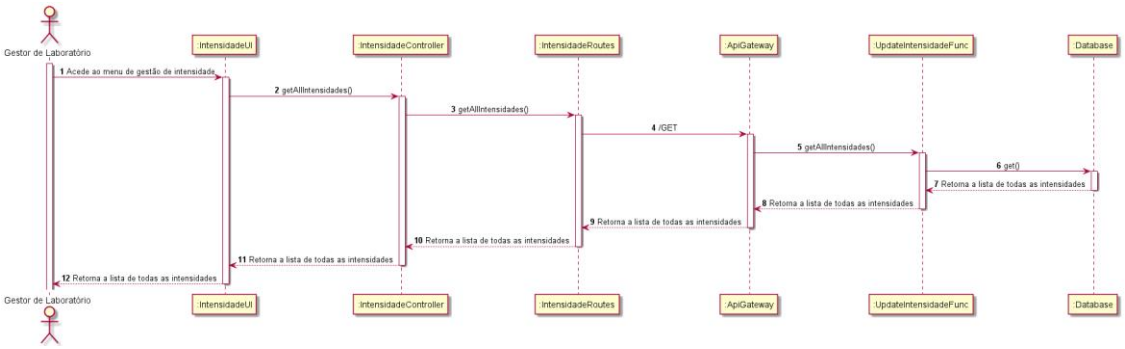


Diagrama de sequência:



UC4.3 Editar Intensidade

Diagrama de sequência do sistema:

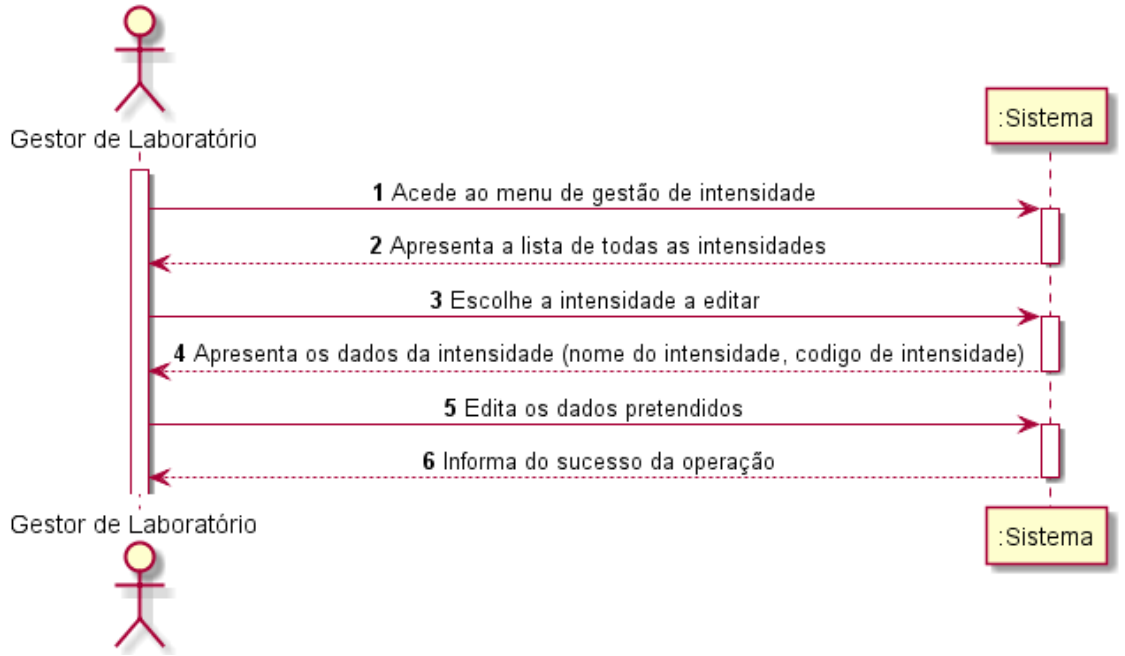
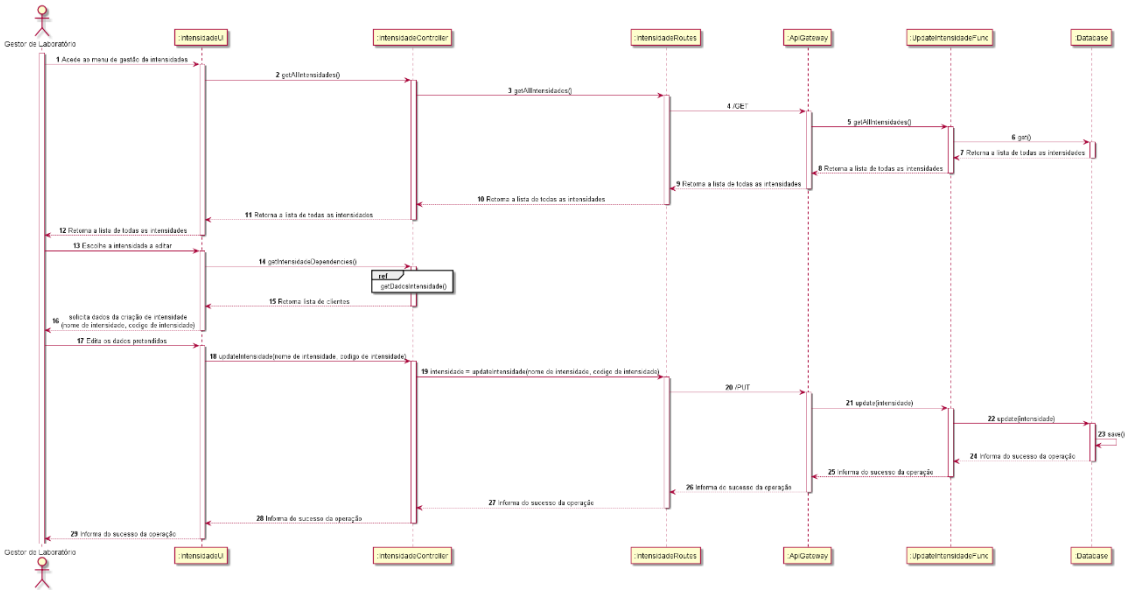


Diagrama de sequência:



UC4.4 Eliminar Intensidade

Diagrama de sequência do sistema:

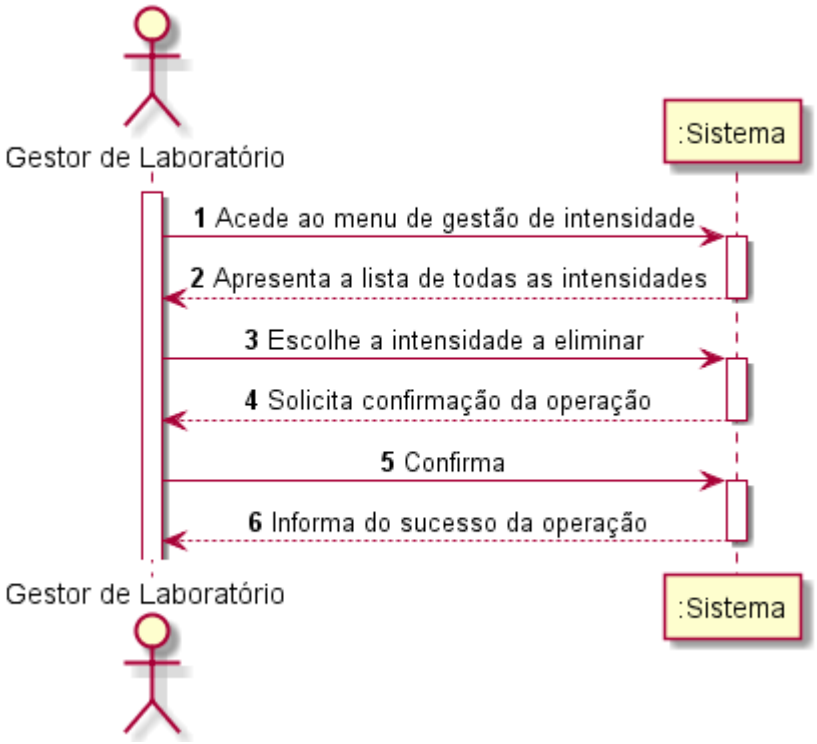
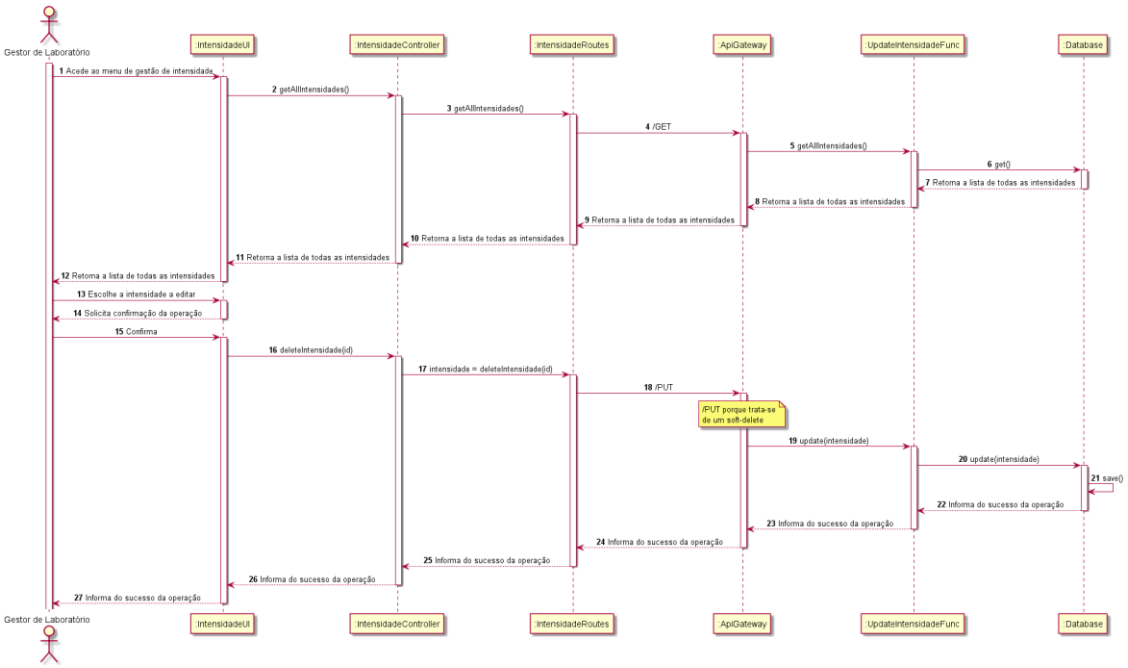


Diagrama de sequência:



UC5.1 Criar Ensaio

Diagrama de sequência do sistema:

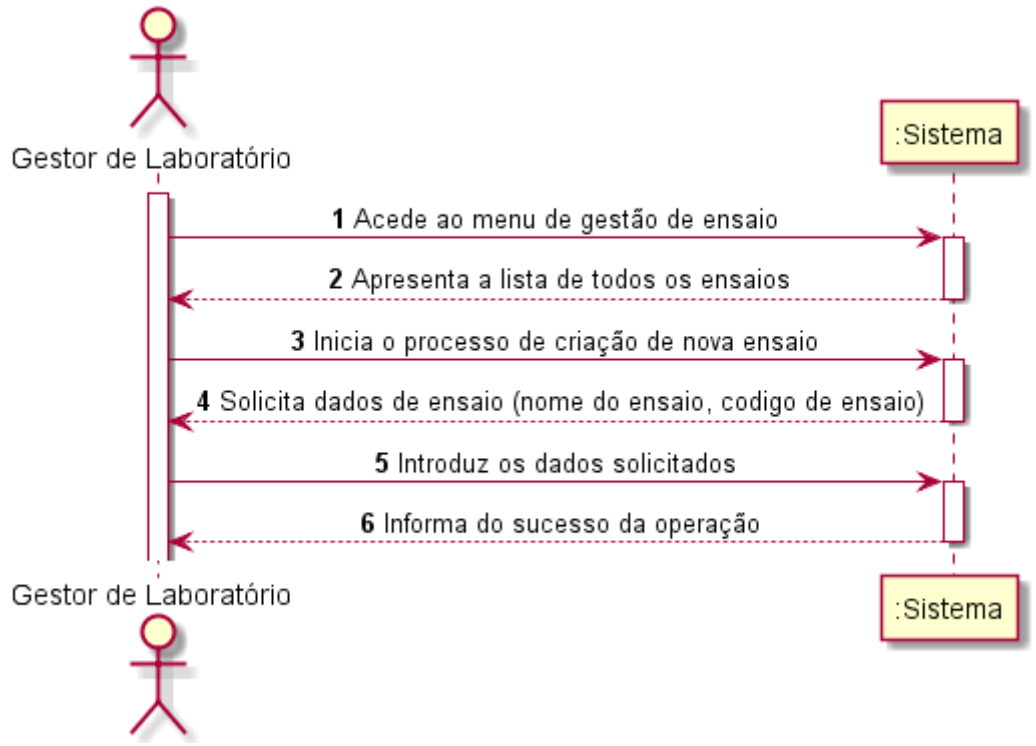


Diagrama de sequência:

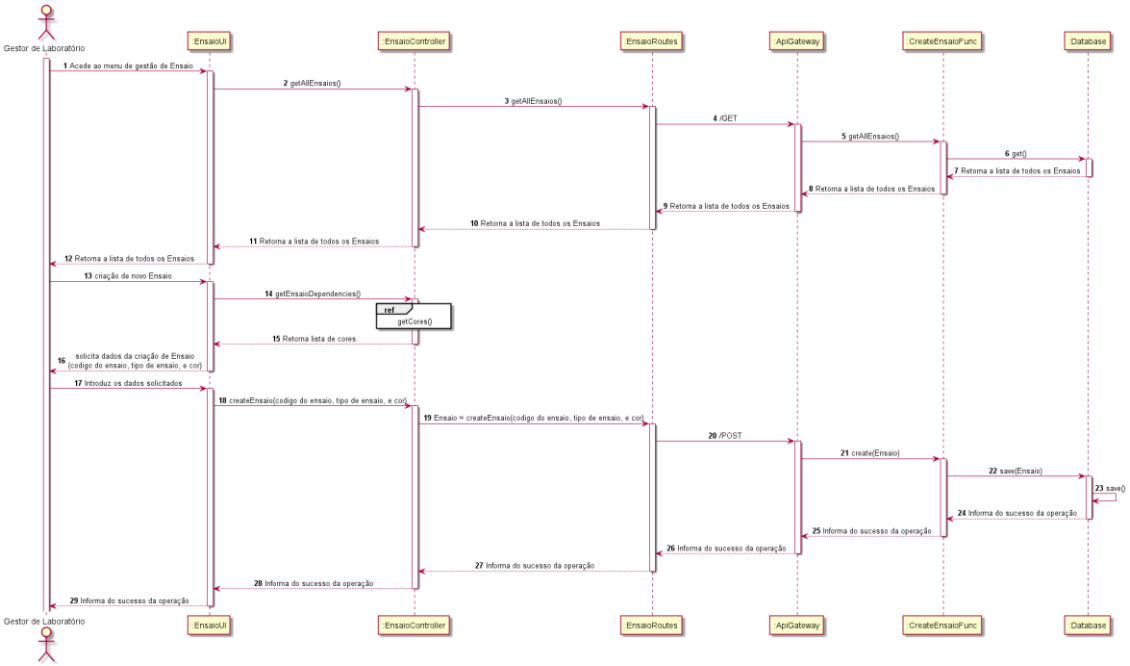
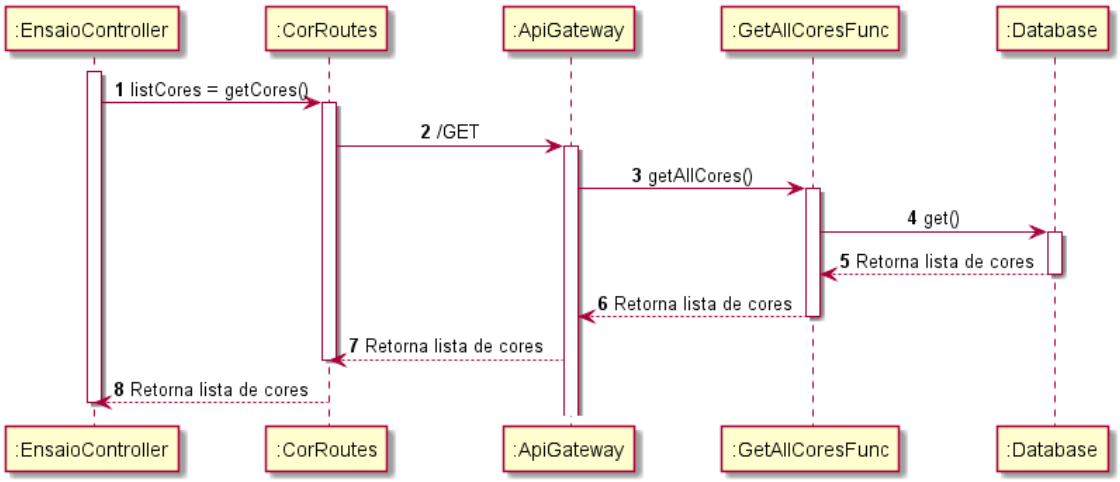


Diagrama da ação getCores:



UC5.2 Consultar Ensaio

Diagrama de sequência do sistema:

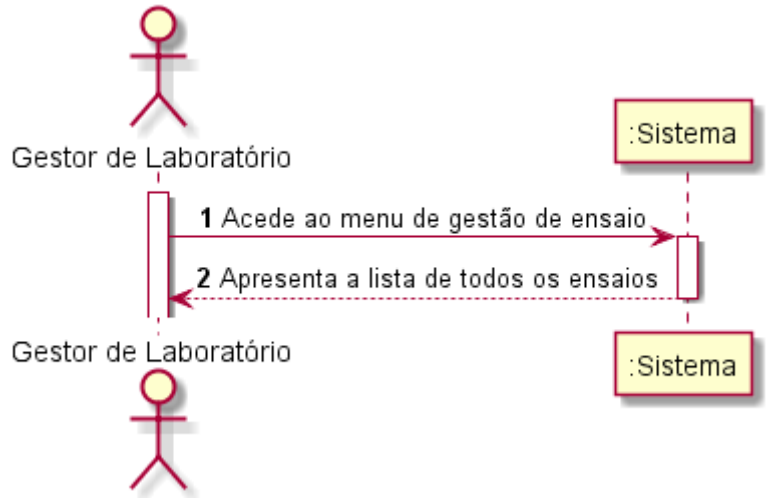
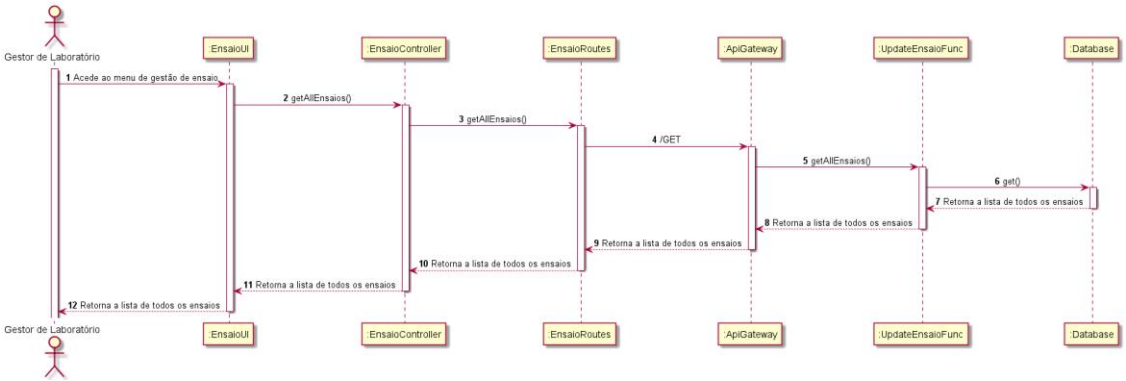


Diagrama de sequência:



UC5.3 Editar Ensaio

Diagrama de sequência do sistema:

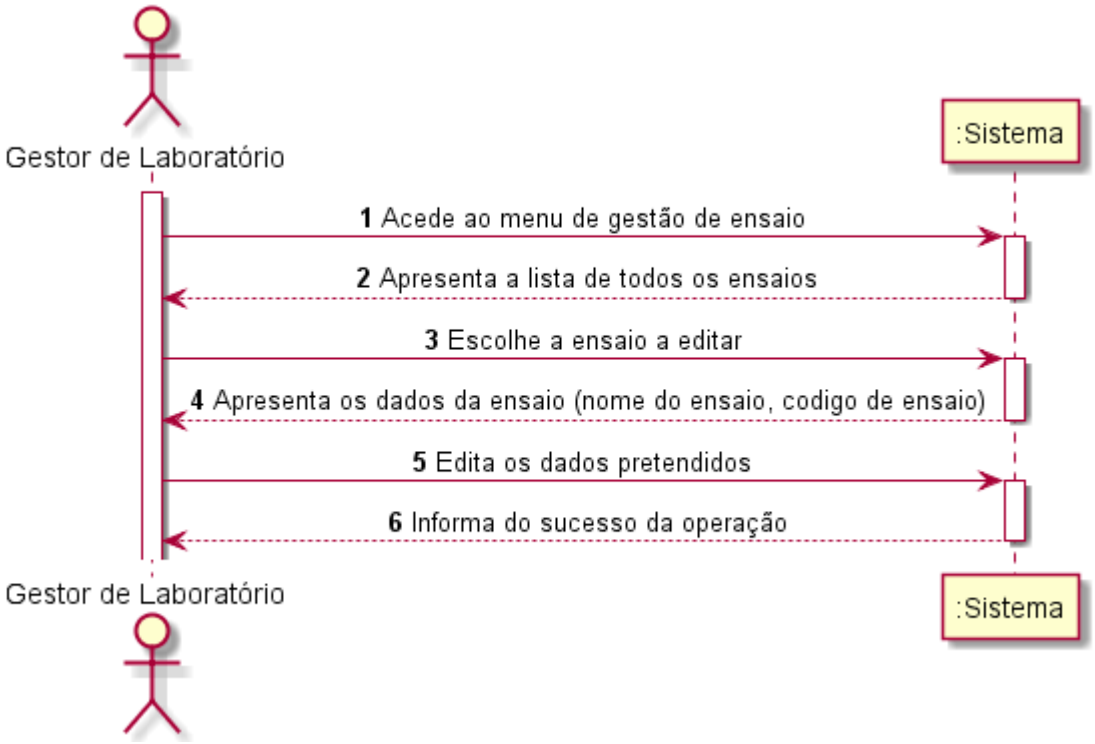
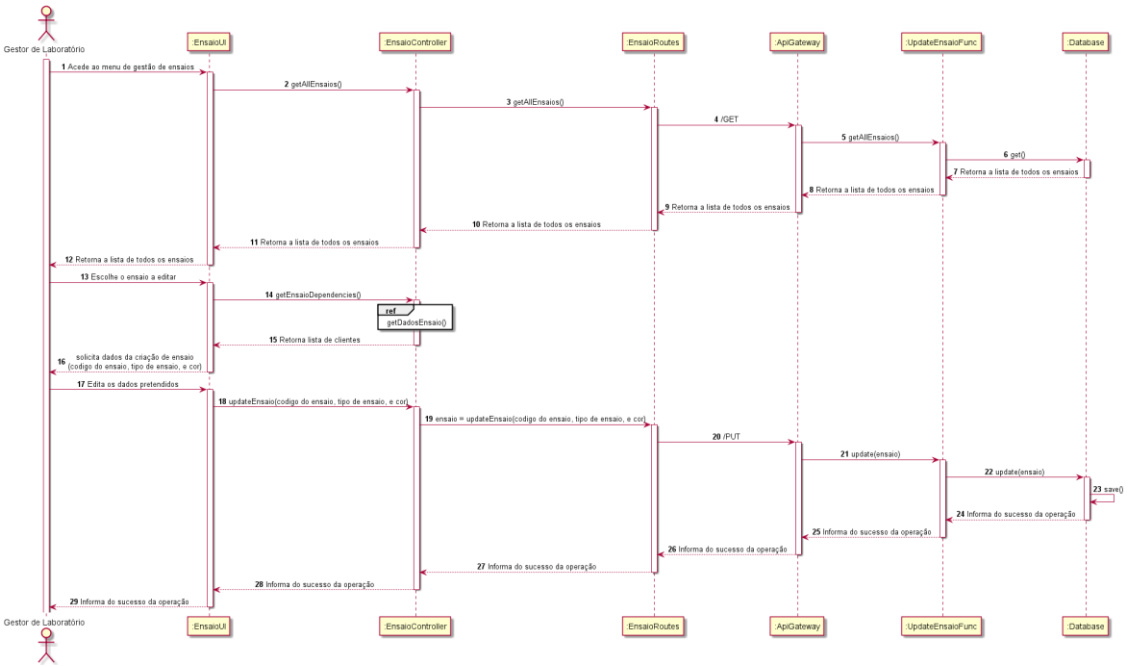


Diagrama de sequência:



UC5.4 Eliminar Ensaio

Diagrama de sequência do sistema:

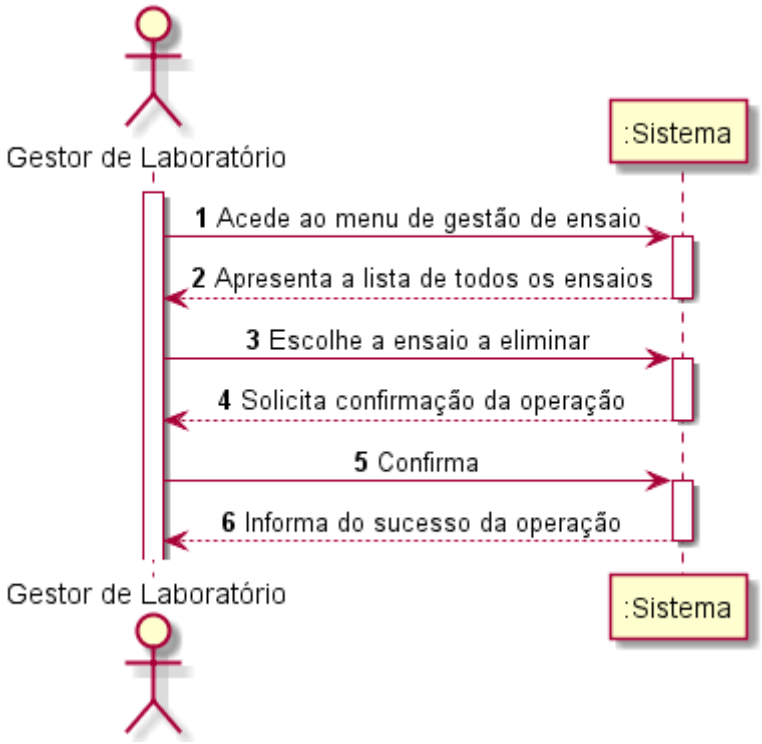
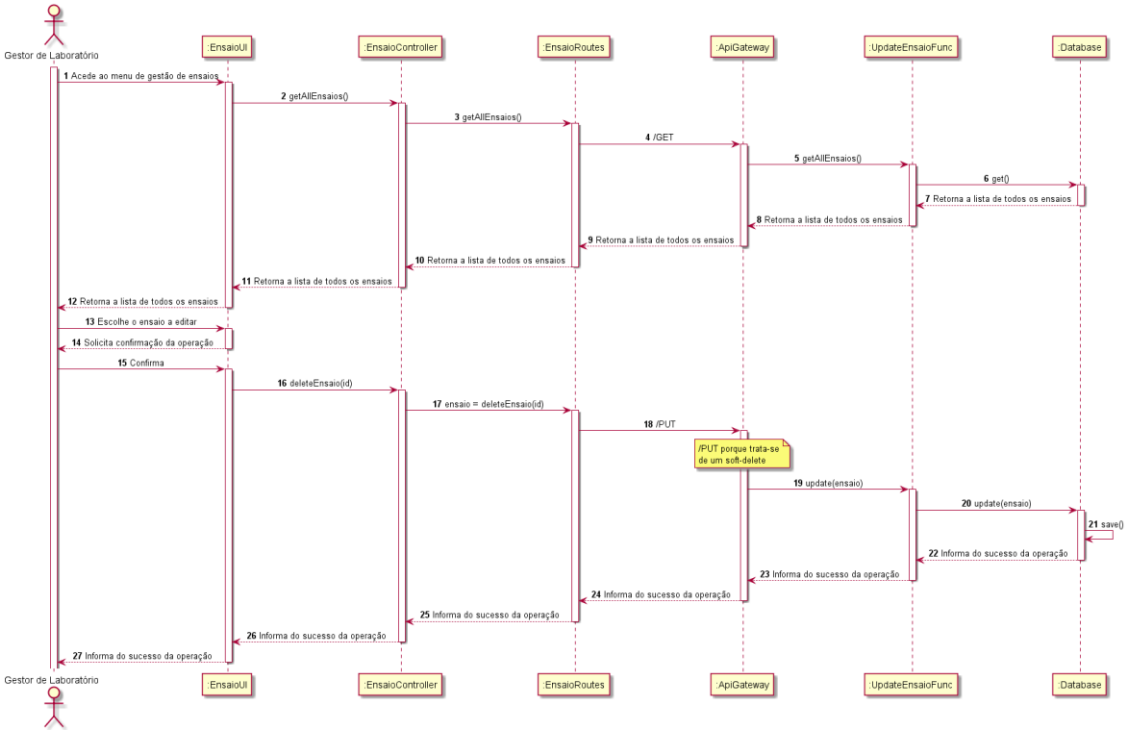


Diagrama de sequência:



UC6.1 Criar Grupos de Corantes

Diagrama de sequência do sistema:

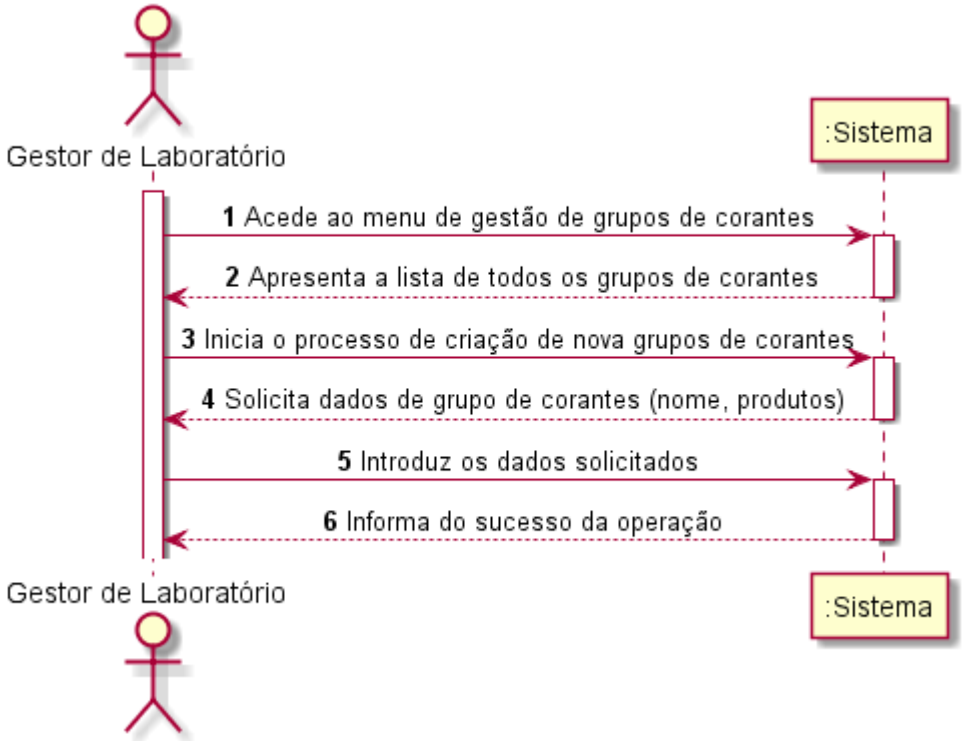
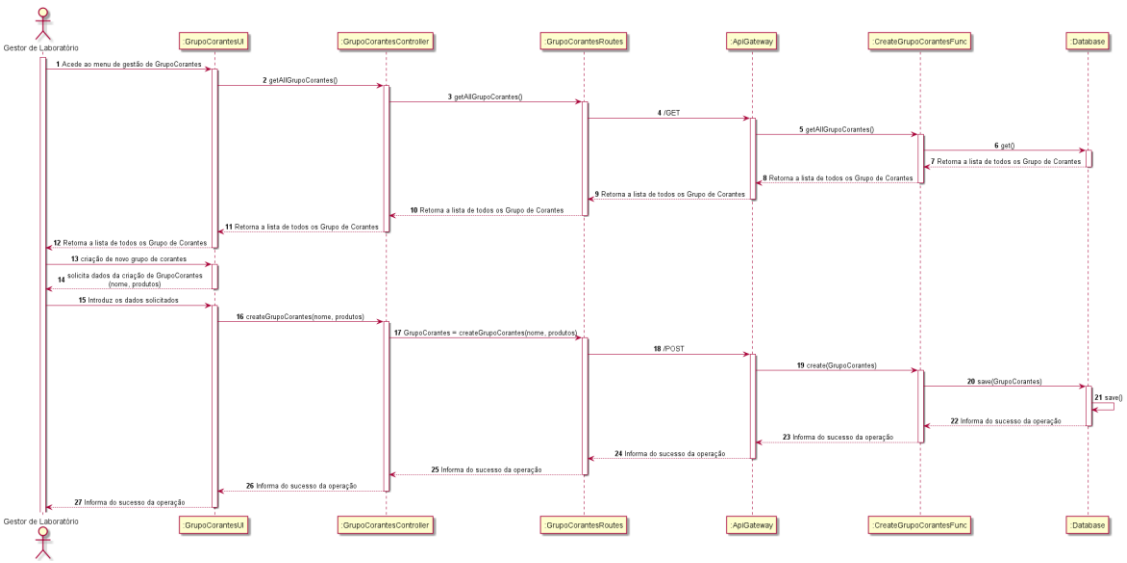


Diagrama de sequência:



UC6.2 Consultar Grupos de Corantes

Diagrama de sequência do sistema:

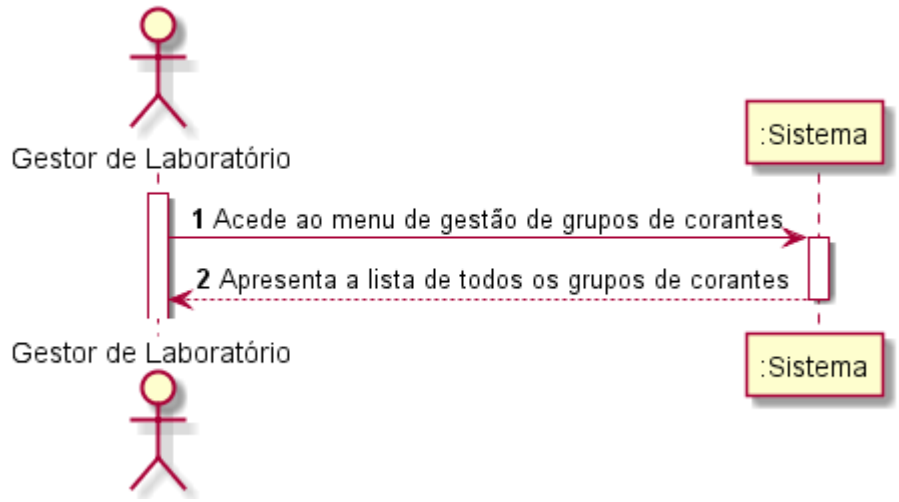
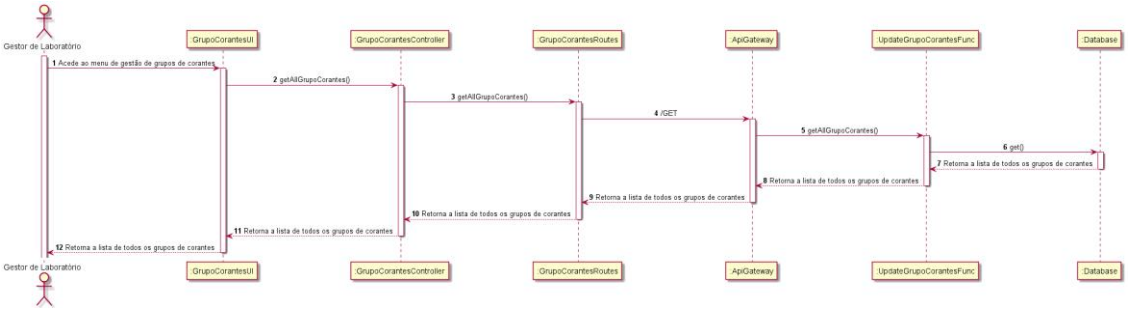


Diagrama de sequência:



UC5.3 Editar Ensaio

Diagrama de sequência do sistema:

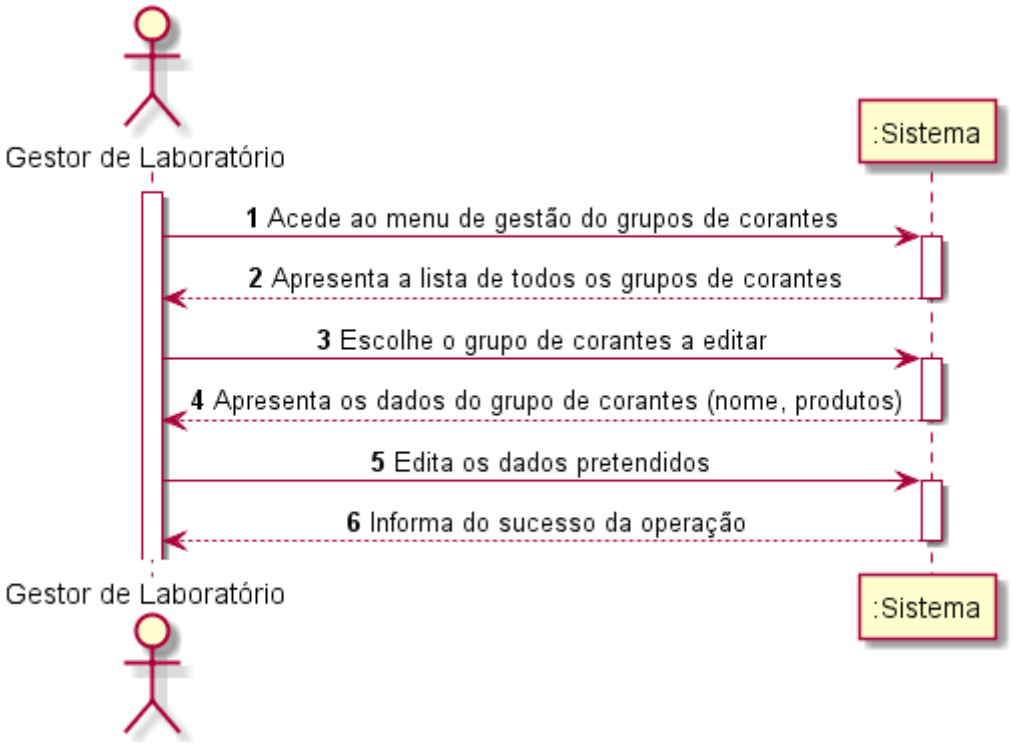
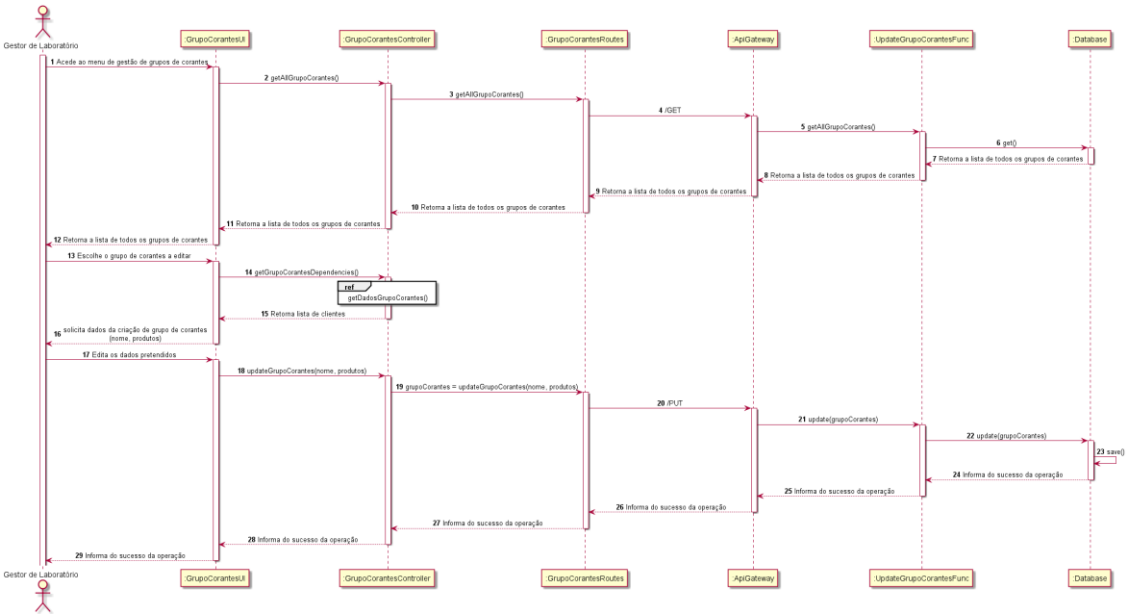


Diagrama de sequência:



UC5.4 Eliminar Ensaio

Diagrama de sequência do sistema:

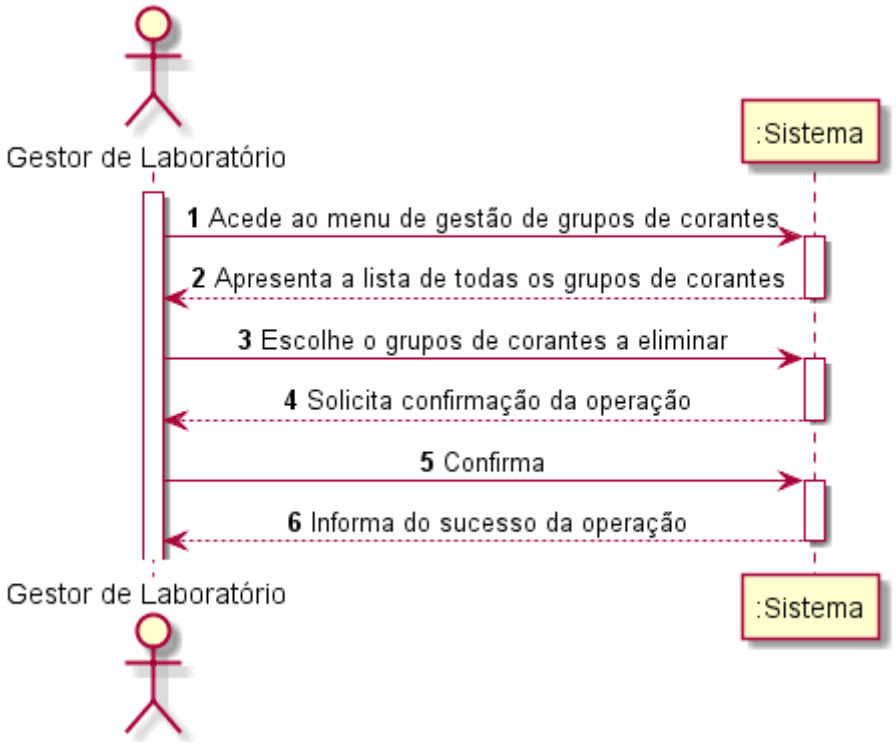
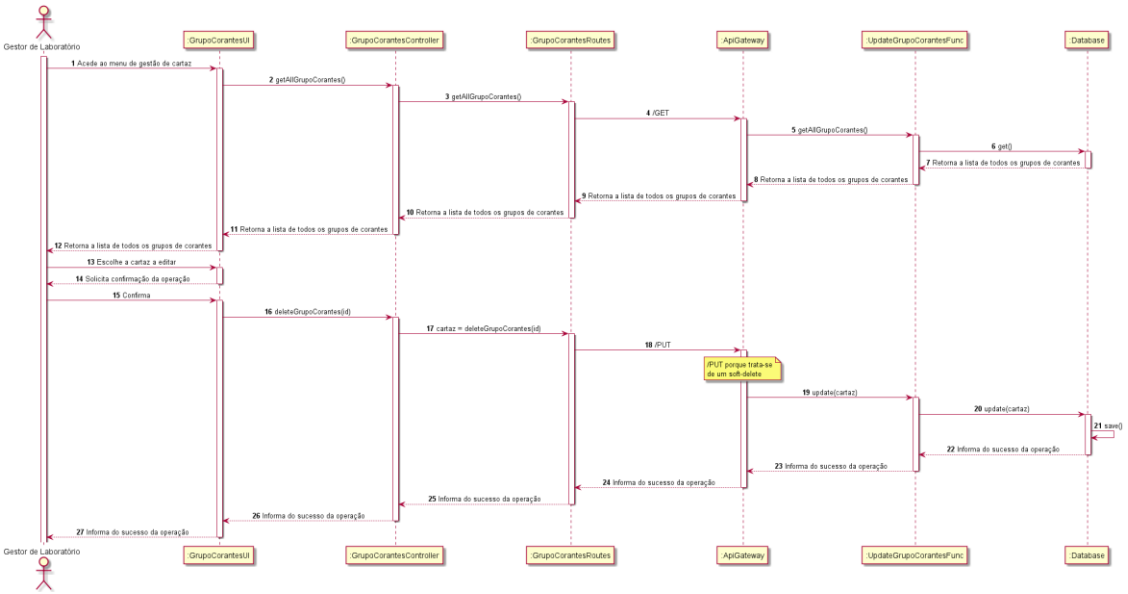


Diagrama de sequência:



UC5.1 Criar Ensaio

Diagrama de sequência do sistema:

Diagrama de sequência:

UC5.2 Consultar Ensaio

Diagrama de sequência do sistema:

Diagrama de sequência:

Diagrama da ação get:

UC5.3 Editar Ensaio

Diagrama de sequência do sistema:

Diagrama de sequência:

Diagrama da ação get:

UC5.4 Eliminar Ensaio

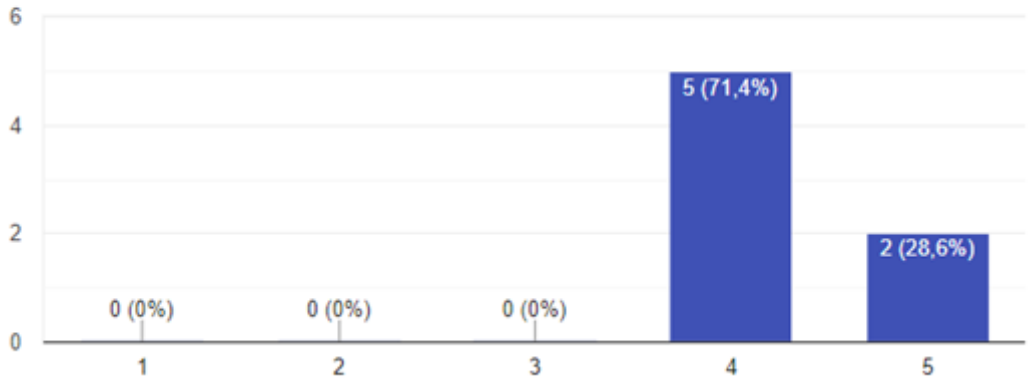
Diagrama de sequência do sistema:

Diagrama de sequência:

Anexo G – Questionário de Aceitação

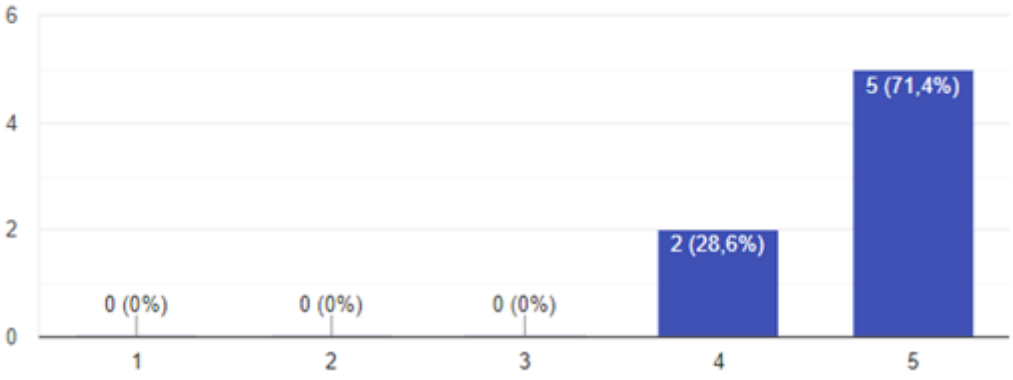
Qual o grau de satisfação da interface do utilizador na plataforma?

7 respostas



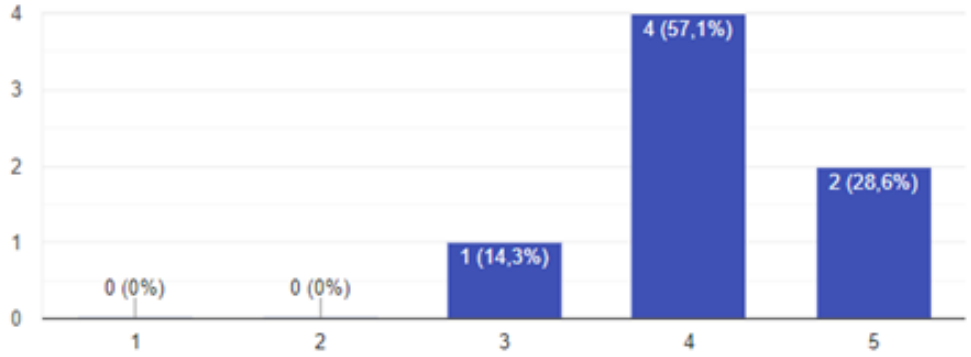
Qual o grau de satisfação do tempo de duração de um caso de uso (ex: criar uma referência)?

7 respostas



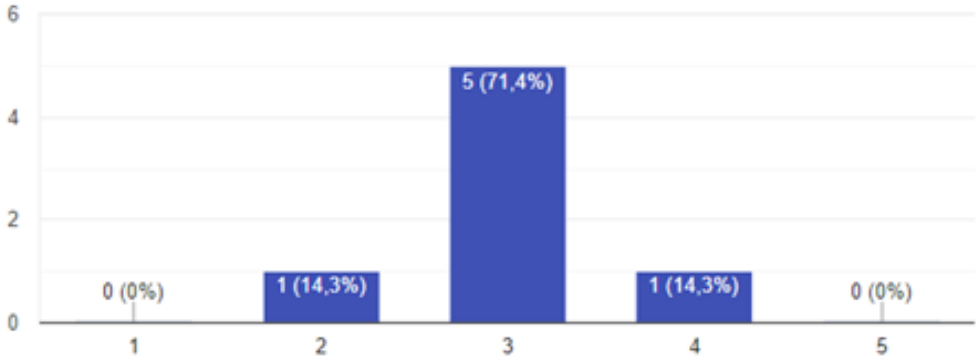
Qual o grau de satisfação que teria caso usasse esta solução em detrimento da que usa atualmente?

7 respostas



Qual o grau de satisfação em ter uma solução que depende 100% de ligação à internet numa empresa têxtil?

7 respostas



Qual de grau de satisfação na adaptação da interface a diferentes dispositivos?

7 respostas