INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de Visualização 3D e Ajuste Manual de Soluções de Paletização

JOÃO PEDRO ROCHA PINHO
Julho de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de Visualização 3D e Ajuste Manual de Soluções de Paletização

João Pedro Rocha Pinho

Dissertação para Obtenção do Grau de Mestre em

Engenharia Informática, Área de Especialização em

Sistemas Gráficos e Multimédia

Orientador: João Paulo Pereira

Coorientador: Luís Guardão

Porto, julho 2019

Dedicatória

“Ó profundidade das riquezas, tanto da sabedoria, como da ciência de Deus! Quão insondáveis são os seus juízos, e quão inescrutáveis os seus caminhos! Porque quem compreendeu a mente do Senhor? Ou quem foi seu conselheiro? Ou quem lhe deu primeiro a ele, para que lhe seja recompensado? Porque dele e por ele, e para ele, são todas as coisas; glória, pois, a ele eternamente. Amém.”

(Romanos 11:33-36)

Resumo

O desenvolvimento económico, e a evolução dos modelos de negócio, trazem cada vez mais exigência aos processos logísticos de armazenagem e movimentação de bens entre cliente e fornecedor. Estes processos de armazenagem, manipulação e envio de bens necessita de ser realizado atempadamente e com o menor custo possível. Um dos fatores mais importantes para atingir esses objetivos é o aproveitamento eficiente do espaço que consiste no processo de paletização. Este consiste na colocação e acondicionamento dos produtos numa ou mais camadas sobre paletes, cumprindo restrições dimensionais, de equilíbrio, de peso máximo, entre outras.

Este caso de estudo foca-se na conceção e manipulação de soluções de paletização. No âmbito deste trabalho foi desenhado e implementado um sistema de pré-visualização e manipulação dessas soluções (obtidas através de uma ferramenta de otimização que não foi alvo do trabalho desta tese). Este sistema apoia o utilizador na realização de eventuais ajustes manuais e preferências que se mostrem necessárias à posterior validação das soluções e à sua colocação em produção.

O sistema divide-se em dois subsistemas distintos, contudo complementares, com interfaces de utilizador para web: um primeiro, referente à criação e manipulação de soluções de paletização em duas dimensões, e um segundo, que permite a pré-visualização das soluções em três dimensões. Foram estudadas e selecionadas as melhores práticas e técnicas nas áreas da Usabilidade, Design Centrado no Utilizador e Experiência do Utilizador para adaptar a solução final ao utilizador, permitindo-lhe uma experiência produtiva e agradável.

Numa fase final do projeto, a solução foi avaliada qualitativa e quantitativamente através da realização de questionários com o fim de avaliar a experiência subjetiva dos utilizadores. Os resultados dessa avaliação mostraram que os objetivos foram atingidos, tendo-se respeitado na integra os requisitos definidos para o sistema.

Palavras-chave: Usabilidade, Design Centrado no Utilizador, Experiência do Utilizador, Paletização

Abstract

The economic development, and the evolution of business models, bring more and more exigency to the logistic processes of storage and movement of goods between client and supplier. These processes of storage, handling and shipment of goods need to be carried out in a timely manner and at the lowest cost possible. One of the most important factors in achieving these objectives is the efficient use of the space that consists of the palletising process. This involves placing and packing the products in one or more layers on pallets, complying with dimensional, balance and maximum weight restrictions, among others.

This case study focuses on the design and manipulation of palletizing solutions. Within the scope of this work, a system of pre-visualisation and manipulation of these solutions was designed and implemented (the optimization tool that produces these solutions is outside the scope of this thesis). This system supports the user in making any manual adjustments and preferences that may be necessary for the subsequent validation of the solutions and their implementation into production.

The system is divided into two distinct subsystems, however complementary, with user interfaces for the web: a first, referring to the creation and manipulation of palletization solutions in two dimensions, and a second, which allows the visualization of solutions in three dimensions. The best practices and techniques in the areas of Usability, User Centered Design and User Experience were studied and selected to adapt the final solution to the user, allowing a productive and pleasant experience.

In a final phase of the project, the solution was assessed qualitatively and quantitatively through questionnaires in order to assess the subjective experience of the users. The results of this assessment showed that the objectives were achieved, and the requirements defined for the system were fully respected.

Keywords: Usability, User Centered Design, User Experience, Palletization

Agradecimentos

Gostava de começar por mostrar um coração grato à minha família por terem nutrido em mim valores como honestidade, responsabilidade e perseverança que me ajudaram a levar a cabo e concluir este projeto com sucesso.

Quero agradecer à equipa do CESE, e em particular aos que me acompanharam mais de perto: Eng. Guilherme Torres, Eng. João Basto, Eng. Luís Costa, André Pereira, o Eng. João Costa, e em especial ao Eng. Luís Guardão, por que me terem acolhido de braços abertos e por terem tido paciência para me ensinarem e acompanharem ao longo do caminho.

Também, ao meu orientador Prof. Doutor João Paulo Pereira pela sua prontidão e a sua preocupação em ajudar e a indicar o melhor caminho a seguir, no que toca à produção deste documento.

E em especial à Íris Lima, minha noiva, futura Pinho e finalista do curso de Línguas e Estudos Editoriais da Universidade de Aveiro, que me acompanhou no processo desde o início e por me ter ajudado a criar nesta dissertação um discurso coerente e coeso de forma a transmitir melhor as minhas ideias.

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 3

2 Estado da Arte 5

2.1 Usabilidade 5

2.2 Interação Humano-Computador 8

2.3 Design Centrado no Utilizador 9

2.4 Experiência do Utilizador 11

2.4.1 Para Além do Instrumental 11

2.4.2 Emoção e afeto 13

2.4.3 Experiência 14

2.5 Tecnologias 15

2.5.1 Unity 3D e Unreal Engine 15

2.5.2 JavaScript e Modelação 3D 16

3 Análise de Valor 19

3.1 Modelo New Concept Development 19

3.1.1 Identificação da Oportunidade 20

3.1.2 Análise da Oportunidade 20

3.1.3 Génese da Ideia 21

3.1.4 Seleção da Ideia 21

3.1.5 Desenvolvimento do Conceito e Tecnologia 22

3.2 Proposta de Valor 22

3.3 Modelo de Negócio *Canvas* 22

3.4 Sacrifícios e Benefícios 23

3.5 Análise SWOT 24

4 Análise e Design 25

4.1 Análise de requisitos 25

4.1.1 Conceitos 26

4.1.2 Requisitos Funcionais 27

4.1.3 Requisitos Não Funcionais 29

4.2 Design 30

4.2.1 Diagrama de Implantação 31
4.2.2 Arquitetura Geral da aplicação 31
4.2.3 Arquitetura do Componente de Visualização 3D 33
4.2.4 Design do Componente de Manipulação 2D 36

5 Implementação 43

5.1 Visualizador de Soluções em 3D 43
5.1.1 Metodologia e Instalação 43
5.1.2 Estrutura 45
5.1.3 Modelação 45
5.1.4 Painel de Informação e Seleção de Camadas 53

5.2 Manipulador de soluções em 2D 55
5.2.1 Estrutura 55
5.2.2 Representação 56

6 Experiências e Avaliação 65

6.1 Metodologia de Avaliação 65

6.2 Cenário de Teste 66
6.2.1 *Feedback* dos Participantes 69
6.2.2 Discussão sobre os resultados 71

6.3 Inquérito de Satisfação 72

6.4 Cenário de Qualidade – QEF 80

7 Conclusões e Trabalho Futuro 83

7.1 Conclusões 83

7.2 Trabalho futuro 84

7.3 Apreciação Final e Pessoal 86

Lista de Figuras

Figura 1 – Pirâmide da hierarquia das necessidades humanas 12
Figura 2 – Modelo NCD (*New Concept Development*) 20
Figura 3 – Modelo de domínio 27
Figura 4 – Diagrama de fluxo da aplicação 28
Figura 5 – Diagrama de Implantação 31
Figura 6 – Diagrama de sequência AngularJS 31
Figura 7 – Diagrama de Componentes 32
Figura 8 – Diagrama de sequência do visualizador 3D 34
Figura 9 – Diagrama de classes – Mosaic3DViewer 35
Figura 10 – Maquete com painel de navegação à direita 37
Figura 11 – Maquete com painel de navegação à esquerda 38
Figura 12 – Ícones das transformações 39
Figura 13 – Diagrama de classes do manipulador 2D 40
Figura 14 – Exemplificação de iluminações e texturas diferentes 46
Figura 15 – Exemplo visual de um *Normal Map* 47
Figura 16 – Solução com *Interlayers* 48
Figura 17 – Comparação da textura da parede com diferentes composições 49
Figura 18 – Visão total da cena 50
Figura 19 – Câmara a intercetar uma caixa 50
Figura 20 – Vista de Topo da Solução 51
Figura 21 – Vista da solução 51
Figura 22 – Solução de exemplo com iluminações diferentes 52
Figura 23 – Cena com Sombras 53
Figura 24 – Representação do painel de informação 54
Figura 25 – Exemplo de uma camada selecionada 55
Figura 26 – Exemplo de mosaico no manipulador 56
Figura 27 – Secção do menu 57
Figura 28 – Secção dos painéis de navegação e seleção 57
Figura 29 – Exemplo de uma palete no visualizador 58
Figura 30 – Exemplo de guias de auxílio 58
Figura 31 – Modos de seleção em linha e coluna 59
Figura 32 – Painel de informações e controlo 59
Figura 33 – Modelo de uma caixa e dos pontos que a definem 61
Figura 34 – Explicação do algoritmo de distribuição das caixas 61
Figura 35 – Exemplo do ajuste da representação às dimensões da janela 62
Figura 36 – Exemplo do ajuste da representação às dimensões da janela 62
Figura 37 – Exemplo das linhas guias 63
Figura 38 – Representação de diferentes margens de ajuste 63

Figura 39 – Relação entre número de participantes e número de problemas de usabilidade detetados 67
Figura 40 – Exemplo de uma solução a replicar pelos participantes 68
Figura 41 – Passos da solução ideal do cenário de teste 69
Figura 42 – Os ícones dos controlos são representativos da ação realizada. 74
Figura 43 – Os controlos estão dispostos de uma forma confortável. 74
Figura 44 – Os controlos são intuitivos. 74
Figura 45 – As etiquetas dos controlos são úteis. 75
Figura 46 – O sistema de ajuste automático é intuitivo 75
Figura 47 – Consegui realizar as tarefas sem interrupções inesperadas. 76
Figura 48 – Os controlos disponíveis foram suficientes para atingir os objetivos. 76
Figura 49 – Sou capaz de usar o sistema de forma autónoma. 76
Figura 50 – Achei a interface esteticamente agradável. 77
Figura 51 – Eu senti-me realizado ao utilizar esta ferramenta. 77
Figura 52 – O controlo da rotação é intuitivo 78
Figura 53 – É fácil aceder a cada camada individualmente 78
Figura 54 – O Painel de informações mostra informações relevantes 79
Figura 55 – A iluminação da cena é adequada 79
Figura 56 – As texturas são adequadas ao contexto 79
Figura 57 – Achei a interface esteticamente agradável 80
Figura 58 – Exemplo de duas caixas que poderão não estar distribuídas simetricamente 86
Figura 59 – Componente de Manipulação 93
Figura A.60 – Ícones das transformações 94
Figura A.61 – Exemplo a replicar 95

Lista de Tabelas

Tabela 1 – Modelo de negócio Canvas..... 23
Tabela 2 – Tabela dos Sacrifícios e Benefícios 24
Tabela 3 – Análise SWOT..... 24
Tabela 4 – Requisitos Não Funcionais..... 29
Tabela 5 – Resultados obtidos das 25 experiências..... 71
Tabela 6 – Participantes mais eficientes..... 72
Tabela 7 – Opções do questionário..... 73
Tabela 8 – Questões sobre o manipulador de soluções 73
Tabela 9 – Questões sobre o visualizador de Receitas 78

Lista de Excertos de Código

Código 1 – URL de Referência para o Componente de Visualização 3D 44
Código 2 – Elemento que inclui o componente na página HTML 44
Código 3 – Método de inicialização do componente de Visualização 3D 44
Código 4 – Composição da textura da caixa 46
Código 5 – Excerto de código para a criação dos controlos 50
Código 6 – Método de Desenho 51
Código 7 – Excerto de Código para a criação de *eventListeners* 54
Código 8 – Exemplo de elemento HTML que representa um componente 56
Código 9 – Alinhamento ao extremo esquerdo da seleção 60

Acrónimos e Símbolos

Lista de Acrónimos

API	*Application Programming Interface*
CESE	Centro de Engenharia de Sistemas Empresariais
HTTP	HyperText Transfer Protocol, é o protocolo usado na World Wide Web que define como mensagens são formatadas e transmitidas, e como servidores e browsers devem responder a vários tipos de pedidos.
INESC TEC	Instituto de Engenharia de Sistemas e Computadores Tecnologia e Ciência
NCD	New Concept Development
UCD	User Centered Design
UX	User Experience
WWW	*World Wide Web*
WebGL	API do JavaScript baseada no OpenGL ES (Embedded Systems) 2.0, usado para desenhar gráficos em 2D e 3D em ambientes Web.

1 Introdução

1.1 Contexto

Os avanços tecnológicos têm elevado a humanidade para níveis de produção nunca vistos. Um aumento da procura leva, naturalmente, a uma necessidade de resposta com uma maior oferta e isto, por sua vez, implica a que o nível tecnológico atual nunca seja suficiente. Este é um ciclo que ano após ano, década após década, pode ser testemunhado pelas nações e comunidades afetadas. Desde os primórdios da era tecnológica, com a invenção do motor a vapor, que o ser humano tem vindo a criar oportunidades e incentivos para acelerar este processo de crescimento e inovação (Rüßmann, et al., 2015).

Com estes avanços tecnológicos surgem problemas logísticos. O âmbito da dissertação apresentada neste documento está intimamente relacionado com uma destas questões: o problema da paletização.

Os produtos, normalmente, acabam o seu percurso numa fábrica dentro de uma caixa, com o intuito de serem transportados. É do interesse de quem envia e de quem transporta estas caixas, que este processo seja o mais eficiente possível. Fruto desta necessidade, o problema da paletização surge (Abdou & Yang, 1994).

Este problema consiste na criação de uma configuração de caixas em cima de uma palete, de modo que uma certa carga seja atingida segundo um conjunto de restrições: que as caixas estejam arranjadas de forma ortogonal, isto é, que as arestas das caixas estejam posicionadas paralelamente às arestas da palete, e que não se possam intercetar umas às outras (Silva, et al., 2016).

Este trabalho vai incidir especificamente sobre a manipulação e representação destas configurações. Com isto em mente, os conceitos de “Usabilidade”, “Interação Humano-

Computador" e "Experiência do Utilizador" vão ser explorados de modo a determinar como estes deverão influenciar a forma como os objetivos serão atingidos.

1.2 Problema

A Paletização é o processo de organização de cargas em cima de paletes. A construção de soluções de paletização 3D é uma prática fundamental na logística de transporte e armazenamento de cargas, podendo ser, por vezes, um processo complexo.

Neste caso de estudo, a paletização é feita de forma automática através de um processo mecânico robotizado, dispondo-se a carga segundo um algoritmo. Estes processos encontram-se já tratados e, assim, fora do âmbito desta dissertação, a qual se focará na conceção e desenvolvimento de formas de representação e manipulação (2D e 3D) realistas das soluções de paletização, atualmente inexistentes e absolutamente fundamentais à operacionalização do sistema.

A manipulação das soluções de paletização, pelo operador especializado, terá de obedecer às mesmas restrições que o algoritmo de construção seguiu na elaboração da solução inicial, nomeadamente: limite e distribuição de peso, sequência de carga e descarga, etc.

Será igualmente necessário preparar toda uma arquitetura capaz de lidar com estas novas funcionalidades tanto no *backend* como no *frontend.* No entanto, esta dissertação irá incidir, essencialmente, na interface humano-computador, representação e manipulação das soluções geradas.

1.3 Objetivos

No contexto do projeto a decorrer na instituição, pretende-se apresentar uma solução ao problema exposto anteriormente. Para tal, foram definidos os seguintes objetivos a atingir:

a. Investigação de técnicas de representação espacial de objetos em ambiente web;

b. Desenvolvimento das ferramentas de manipulação das soluções em duas dimensões intuitivas, segundo um conjunto de restrições físicas como limite de peso, sequência de carga e descarga, tamanho da carga, etc.;

c. Desenvolvimento da camada de interface humano-computador para um conjunto de processos de manipulação de soluções de paletização;

d. Desenvolvimento do visualizador das soluções em três dimensões;

e. Integração das novas funcionalidades com serviços existentes.

1.4 Abordagem

O foco desta dissertação consiste na análise, desenho e implementação de uma solução experimental para o problema supracitado. Posteriormente, a experiência e os resultados obtidos pelo utilizador serão avaliados para garantir que estes correspondem às intenções iniciais e do negócio. Tal implica também que sejam exploradas técnicas para se atingir este fim.

O problema descrito neste documento será abordado de forma iterativa e incremental em grande proximidade com o cliente, para guiar o desenvolvimento do produto conforme as suas necessidades. Isto, naturalmente, resultará num levantamento de requisitos exaustivo, que deverá ser flexível a potenciais alterações conforme os desvios que forem achados estritamente necessários, tanto por falha de interpretação como alteração das necessidades iniciais.

Para que o processo de desenvolvimento possa ser iniciado e levado a avante de forma eficaz, terá de ser feita uma pesquisa extensiva de técnicas e tecnologias referentes à usabilidade e design centrado no utilizador, fazendo um levantamento de boas práticas.

Para controlar o progresso do desenvolvimento do projeto será usado o método de controlo de versões chamado de *Git*.

1.5 Estrutura do documento

A presente dissertação apresenta-se dividida em 7 capítulos distintos, cada um descrevendo os vários estágios do desenvolvimento do projeto. Estes capítulos serão brevemente apresentados de seguida.

No presente capítulo foi introduzido o âmbito do projeto, apresentando o problema a resolver e os objetivos a cumprir.

No segundo capítulo realizar-se-á um levantamento de conceitos chave para enquadrar os capítulos seguintes a nível teórico e tecnológico.

No terceiro capítulo será feita a proposta de valor do projeto, descrevendo, identificando, e analisando as oportunidades e ideias geradas. Será também apresentado o modelo de negócio *Canvas* e a análise SWOT para melhor sustentar o valor do projeto descrito no presente documento.

No quarto capítulo serão apresentadas as fases de Análise de requisitos e Design da arquitetura e da interface. Para a primeira serão enumerados os requisitos funcionais e não funcionais, e para a segunda serão apresentadas algumas situações com maior detalhe recorrendo a diagramas em notação UML (do inglês *Unified Modeling Language*) e as maquetes.

No quinto capítulo será sintetizado o processo de implementação do sistema descrito no capítulo anterior, assim como as boas práticas adotadas. As decisões mais relevantes para este processo também serão apresentadas e justificadas.

O sexto capítulo servirá para apresentar os resultados do trabalho descrito no capítulo anterior, tanto ao nível quantitativo da implementação, quanto ao nível qualitativo da experiência subjetiva dos utilizadores, através de um cenário de teste e de um questionário.

Por fim, no sétimo capítulo discutir-se-á sobre os objetivos alcançados e as limitações identificadas.

2 Estado da Arte

No presente capítulo serão explorados conceitos abordados no projeto de forma a contextualizá-lo globalmente. Este processo é crucial para fundamentar todas as decisões futuras no conhecimento de peritos nesta área.

2.1 Usabilidade

O termo Usabilidade é definido no ISO 9241-11 (ISO/TC 159/SC 4, 1998) como: a medida em que um sistema, produto ou serviço pode ser utilizado por utilizadores específicos de modo a atingir determinados objetivos com eficácia, eficiência e satisfação num determinado contexto de utilização.

Os termos chave nesta definição são:

- Eficácia: O utilizador consegue atingir os seus objetivos com precisão?
- Eficiência: Isto pode ser atingido com recursos mínimos: tempo, esforço mental e físico?
- Satisfação: Os utilizadores sentem-se confortáveis, ou felizes, em fazer isto?
- Contexto de utilização: Utilizadores, tarefas, equipamento (*hardware*, *software* e materiais), e os ambientes físicos e sociais em que um produto é usado.

Nielsen descreve a usabilidade como sendo o atributo de qualidade que avalia o quão fácil é a sua utilização. A palavra “usabilidade” refere-se aos métodos de melhorar a facilidade de uso durante o processo de design.

Segundo Nielsen, usabilidade é definida segundo estes cinco componentes (Nielson, 2012):

- Aprendizagem: O quão fácil é para os utilizadores completarem tarefas básicas durante a primeira vez que interagem com o sistema;
- Eficiência: Quão rápido conseguem os utilizadores realizar tarefas, a partir do momento de aprendizagem do design;
- Memorização: A capacidade de um utilizador voltar a usar um sistema com proficiência, após passar algum tempo sem o fazer;
- Erros: Quantos erros os utilizadores cometem, quão severos esses erros são e quão facilmente os utilizadores conseguem recuperar desses erros;
- Satisfação: Quão confortáveis estão os utilizadores em usar o sistema.

Ben Schneiderman e Catherine Plaisant identificam no seu livro (Shneiderman & Plaisant, 2004) cinco atributos da usabilidade que podem ser testados. Os conceitos anteriormente referidos como a eficiência e a eficácia são mais vagos e por isso mais difíceis de serem testados diretamente. De seguida serão enumerados e descritos estes cinco atributos:

- Curva de aprendizagem: Para um dado utilizador, o tempo (máximo, mínimo, médio, etc.) ou número de repetições para atingir um nível de desempenho, seja em termos de velocidade ou erros cometidos, para uma tarefa específica;
- Velocidade de execução: Para um dado utilizador, qual o tempo (máximo, mínimo, médio, etc.) para completar uma dada tarefa;
- Precisão: Para um dado utilizador, a proporção de erros cometidos;
- Capacidade de memorização: O nível de desempenho, seja em termos de velocidade ou de erros cometidos, passado um período de tempo;
- Satisfação subjetiva: A avaliação subjetiva do utilizador em termos gerais do *software*, normalmente através de um questionário.

Embora uma análise superficial destes elementos possa sugerir que o ideal seja maximizá-los para uma experiência mais positiva, Shneiderman e Plaisant apontam que estes estão todos interligados e dependem uns dos outros. Isto significa que qualquer investimento num destes atributos influencia, tanto positiva como negativamente, os outros e isto por sua vez resulta na necessidade de se encontrar um balanço.

Durante muitos anos, a satisfação do utilizador perante uma dada interface, tem sido ignorada de uma forma geral. No entanto, mais recentemente, outros aspetos como a motivação, confiança, gozo e envolvimento têm sido alvo de maior preocupação.

O conceito de usabilidade é de certa forma abrangente. Num nível mais baixo, há a disposição visual de informação e controlos num écran, ou num aparelho físico, e os seus comportamentos mais imediatos. Num nível mais alto, deve ter-se em conta todo o contexto organizacional e social: as pessoas que vão usar o sistema que está a ser desenhado, as suas crenças e valores, propósito e limitações do design (Blanton, et al., 2009).

Outro termo muito importante é o da utilidade, que remete para a funcionalidade do design. Este avalia se o design faz realmente o que o utilizador necessita. Nielson explica que os termos "usabilidade" e "utilidade" são igualmente importantes e juntos determinam se algo é útil: pouco importa se algo é fácil se não preenche uma necessidade. Também não importa se o sistema pode, hipoteticamente, fazer o que é necessário, mas o utilizador não consegue usufruir da funcionalidade devido à incompetência do design. Para estudar a utilidade de um design, pode usar-se os mesmos métodos de investigação dos utilizadores que melhoram a usabilidade. (Nielson, 2012)

Um sistema potencialmente útil pode ser inutilizável. Um sistema que é usável – um que é aprendido e manuseado facilmente – pode ser inútil, não conseguindo servir algum propósito reconhecido. Como esforços de pesquisa e projetos de desenvolvimento ambicionam suportar, e produzir, sistemas ou aplicações que são tanto úteis como utilizáveis, é, portanto, intuitivamente apelativo que as duas qualidades sejam consideradas em conjunto. Apesar disto, elas raramente são consideradas desta forma (Grudin, 1992).

Um movimento Britânico e Escandinavo apercebeu-se do potencial benefício da agregação do estudo da utilidade e da usabilidade (Bjerknes, et al., 1987). Ao trabalhar num contexto de desenvolvimento interno, estes investigadores e desenvolvedores reconheceram as limitações da metodologia designada por "cascata desenha-primeiro" (em inglês *design-it-first waterfall*). Eles aceitaram adiar decisões sobre funcionalidade o mais tarde possível, enquanto maximizavam o envolvimento do utilizador no design e no desenvolvimento.

Se adotada, esta abordagem culmina na consideração paralela da utilidade e usabilidade de um sistema. Todavia, para aplicá-la será necessário reestruturar substancialmente as práticas internas de desenvolvimento o que poderá ser difícil em empresas grandes. Apesar disto, é do interesse destas empresas que a utilidade e a usabilidade sejam tratadas em conjunto. Os utilizadores de grandes sistemas internos estão na expectativa de que estas interfaces sejam tão boas quanto as presentes em produtos caseiros, como um simples micro-ondas. Produtos de *software* comerciais, para reter uma vantagem, terão de ser adaptados a nichos específicos, requerendo um maior foco na utilidade. Estas forças estão a descentralizar o desenvolvimento de *software* das grandes empresas de sistemas informáticos para empresas de desenvolvimento mais pequenas. Tomando partido de sistemas abertos e ferramentas que facilitam o desenvolvimento de interfaces, estas empresas são as futuras fontes de progresso na área da Interação Humano-Computador (Grudin, 1992).

No entanto, há razões para tratar a utilidade e a usabilidade separadamente quando isto for feito sem pôr em risco o produto. Isto facilita a divisão do trabalho e a utilização de uma forma mais completa das capacidades especializadas de uma equipa. Para além disto, isolar aspetos da usabilidade promove a flexibilidade. Uma interface que pode ser modificada pode mais facilmente ser ajustada a plataformas mais avançadas de *hardware* e *software*, a audiências diferentes e a utilizadores individuais com base nos seus papéis, experiência, características físicas e preferências. Finalmente, considerar a interface de uma forma independente pode ajudar a promover consistência nas interfaces, reduzindo o esforço em aprender novas aplicações (e talvez reduzir o esforço no design) (Grudin, 1992).

2.2 Interação Humano-Computador

A Interação Humano-Computador, em inglês *Human-Computer Interaction* (HCI), é o estudo da forma como a tecnologia de computadores influencia o trabalho e atividades humanas. O termo "tecnologia de computadores" atualmente inclui a maioria da tecnologia de computadores mais comuns como ecrãs e teclados até telemóveis, eletrodomésticos, sistemas de navegação dos automóveis, e até sensores e atuadores embutidos como iluminação automática. A HCI tem uma disciplina de design associada, por vezes chamada de Design de Interação ou Design Centrado no Utilizador, em inglês *Interaction Design* e *User-Centered Design* respetivamente, focada em desenhar tecnologia de computadores para que seja, tanto quanto possível, fácil e agradável de usar. O aspeto chave da disciplina de design é a noção de "usabilidade", que é muitas vezes definida em termos de eficiência, eficácia e satisfação. Mas algo que é também muito importante considerar, para além da usabilidade, é a experiência dos utilizadores, em inglês *User Experience* (UX): a forma como estes se sentem ao usar o sistema (Blanton, et al., 2009). Os termos "Design Centrado no Utilizador" e "Experiência do Utilizador" serão aprofundados mais adiante.

As raízes da HCI podem ser encontradas desde o trabalho mais pragmático de Brain Shackel no fim dos anos 50, até ao trabalho inovador de Douglas Englebart no *Augmented Research Center* em Stanford no início dos anos 60 (Shackel, 1959), (Englebart, 1962).

No entanto, foi com a popularização da computação pessoal, no início dos anos 80, que a disciplina ganhou forma e conquistou o seu lugar no meio académico. Neste período inicial, os ramos académicos da psicologia, computação e ergonomia serviram de base para o trabalho desenvolvido. Ainda assim a maioria do trabalho foi fundado com base em fortes metodologias experimentais (Blanton, et al., 2009).

Notoriamente, o centro de Investigação Xerox PARC, baseado em Palo Alto na Califórnia, teve um papel crucial no que toca à HCI. Este centro desenvolveu o estilo de interface denominado por WIMP, *Windows-Icons-Menus-Pointer*, que eventualmente foi popularizado pela *Macintosh,* em 1984. Esta interface foi tão marcante que até aos dias de hoje é usada nos computadores pessoais. As interfaces têm hoje a mesma aplicação e sensação que tinham as primeiras

implementações do WIMP. No entanto, as interfaces hoje utilizam certas técnicas visuais, como o uso de sombras para simular a sensação de profundidade, tentando promover uma experiência mais agradável ao utilizador. Isto deve-se também ao avanço tecnológico no que toca à transição de ecrãs monocromáticos para policromáticos e em ferramentas muito avançadas para o desenvolvimento de interfaces WIMP. Estas interfaces são tão boas para casos de uso em *desktop*, que o campo de investigação está estagnado numa rotina confortável (van Dam, 1997).

Como já foi dito, a HCI é tanto uma disciplina académica, que estuda a forma como a tecnologia impacta a atividade humana, como também uma disciplina de design, com o objetivo de desenhar essa tecnologia com o máximo de eficácia. Baseia-se em muitos resultados de outras disciplinas como a psicologia e a sociologia, mas também nos seus próprios métodos e técnicas.

2.3 Design Centrado no Utilizador

O design centrado no utilizador, do inglês *User-Centered Design* (UCD), é um termo abrangente que descreve processos de design que envolve a participação e, por conseguinte, a influência dos utilizadores finais. É muito importante para este conceito que os utilizadores façam parte do processo, e há todo um espectro de maneiras de os envolver. Por exemplo, alguns tipos de UCD consultam os utilizadores acerca das suas necessidades, envolvendo-os em alturas específicas durante o processo de design. Tipicamente isto acontece durante a análise de requisitos e testes de usabilidade. No lado oposto deste espectro, há métodos de UCD nos quais os utilizadores finais do produto têm um impacto profundo no design, contactando diretamente com os designers ao longo de todo o processo. (Abras, et al., 2004)

O termo UCD teve a sua génese no laboratório de investigação de Donald Norman na Universidade de San Diego, Califórnia, nos anos 80 e popularmente usado depois da publicação de um livro que coautorou chamado "*User-Centered System Design: New Perspectives on Human-Computer Interaction*" (Norman & Draper, 1986). Norman edificou mais sobre o conceito UCD no seu livro "*The Psychology Of Everyday Things* (1988)*",* mais tarde editado para "*The Design of Everyday Things*" (Norman, 2013). Neste livro, Norman reconhece necessidades e interesses do utilizador e foca a sua atenção na usabilidade do design. Ele sugere quatro conceitos básicos (Norman, 2013, p. 188) que um design deve apresentar sempre:

- Tornar claro que tarefas estão disponíveis em todo e qualquer momento;
- Tornar coisas visíveis, incluindo o modelo conceptual do sistema, as ações alternativas e os resultados dessas ações;
- Tornar fácil a avaliação do estado atual do sistema;

- Seguir mapeamentos naturais entre: intenções e ações necessárias; ações e efeitos resultantes; informação visível e interpretação do estado do sistema.

Estas recomendações tornam o utilizador no centro do design. O papel do autor do design é de facilitar as tarefas ao utilizador e de garantir que este consegue utilizar o produto da forma pretendida com um esforço mínimo na aprendizagem.

O princípio basilar de que os produtos "devem ser intuitivos" é vago, pelo que é necessário obedecer a alguns princípios para atingir esse "objetivo subjetivo". Ben Shneiderman sugere oito regras "de ouro" (Shneiderman, 2004) para desenhar uma interface:

- Ambicionar a consistência: Ações similares deverão ser usadas em situações similares; os elementos da interface deverão ser consistentes desde a terminologia usada em janelas de diálogo, às cores usadas e à disposição dos elementos.

- Assegurar a usabilidade universal: Reconhecer as necessidades de diversos utilizadores e preparar o design do sistema para a plasticidade, facilitando a manipulação de conteúdo. Por exemplo, adicionar funcionalidades para novatos, como ajudas e explicações, e para experientes, como atalhos, que enriquecem o design da interface e melhoram a qualidade observada.

- Oferecer *feedback* informativo: Para cada ação do operador, deverá haver algum *feedback* da interface. Para ações frequentes e menos importantes, a resposta pode ser concisa, enquanto que para ações importantes e menos frequentes, a resposta deverá ser mais substancial. A apresentação visual do objeto de interesse fornece um ambiente conveniente para mostrar explicitamente qualquer alteração.

- Desenhar diálogos que indiquem o fim de uma ação: Sequência de ações deveriam estar organizadas em grupos com inícios, meios e fins. *Feedback* informativo na conclusão de um grupo de ações dá ao utilizador uma sensação de realização, um sentimento de alívio, um sinal para libertar os planos de contingência das suas mentes e um indicador que o prepare para o próximo grupo de ações. Por exemplo, websites de comércio eletrónico movimentam os seus utilizadores da seleção de produtos para a secção de *checkout*, terminando com uma página de confirmação clara da transação completa.

- Prevenir erros: Tanto quanto possível, desenhar o sistema de forma a que o utilizador não consiga produzir nenhum erro catastrófico; por exemplo, itens de menus que não são apropriados para a ação a ser executada deverão estar inativos, e campos de texto numéricos não deverão aceitar carateres alfabéticos. Se o utilizador cometer um erro, a interface deverá instruir o utilizador de forma clara, simples e construtiva de modo a recuperar em segurança. Por exemplo, o utilizador não deverá ter que reescrever um formulário inteiro se apenas um dos campos estiver errado. A interface deverá, no

entanto, guiar o utilizador a corrigir apenas o campo errado e dar instruções ao utilizador sobre como restaurar o sistema.

- Permitir que utilizadores recorrentes utilizem atalhos: Tanto quanto possível, ações deverão ser reversíveis. Esta funcionalidade reduz a ansiedade e a frustração porque dá aos utilizadores a garantia que erros podem ser revertidos e encoraja a exploração de opções desconhecidas. A unidade de reversibilidade poderá ser de uma única ação, uma entrada de informação, ou um grupo de ações completo.

- Manter o utilizador no controlo: Utilizadores experienciados querem pensar que estão no controlo da interface e que esta responde às suas ações. Estes utilizadores não querem surpresas ou alterações em comportamento familiar e, para eles, sequências de introdução de informação, dificuldade em obter informação necessária e a incapacidade de obter os resultados esperados são desagradáveis.

- Reduzir a carga da memória de curto prazo: A capacidade limitada dos seres humanos em processar informação em memória de curto prazo requer que desenvolvedores evitem interfaces que obriguem utilizares a decorar informação num painel e depois a usem noutro. Isto significa que, por exemplo, formulários extensos deverão ser compactados de forma a caberem apenas num écran.

2.4 Experiência do Utilizador

Na última década, o termo Experiência do Utilizador, no inglês "*User Experience*" (UX), tem-se tornado cada vez mais popular no campo da Interação Humano-Computador e Design de Interação, tanto por profissionais como por investigadores, (Hassenzahl & Tractinsky, 2006). Embora seja usado com frequência, é repetidamente criticado por ser vago, indescritível e efémero. Este termo está relacionado com uma variedade de significados, desde a tradicional usabilidade, até aos aspetos de beleza e afeção que advêm da utilização de tecnologia (Forlizzi & Battarbee, 2004).

Fazendo uma revisão da literatura sobre UX, como das Conferências "*Design and Emotion*" (McDonagh, et al., 2004), ou a literatura sobre Estética (Tractinsky & Zmiri, 2006), três perspetivas principais são reveladas: uma linha de pensamento lida predominantemente com as necessidades humanas para além do instrumental, do físico; a segunda linha dá ênfase aos aspetos afetivos e emocionais da interação; e uma terceira, lida com a própria natureza da experiência (Hassenzahl & Tractinsky, 2006).

2.4.1 Para Além do Instrumental

Desde os seus primórdios, a investigação na área da HCI tem sido focada exclusivamente na capacidade de atingir objetivos a nível comportamental, num ambiente profissional. A "tarefa"

tornou-se, no ponto mais importante na análise, centrada no utilizador e na avaliação de técnicas (testes de usabilidade). Garantir o valor instrumental dos produtos interativos tornou-se o foco de todo o campo de investigação.

No entanto, esta visão redutora sobre o instrumental tem sido desafiada repetidamente. Numa tentativa inicial de definir o termo UX, visto por exemplo em (Alben & Lauralee, 1996), a estética foi identificada como um aspeto da tecnologia de grande importância. A beleza é claramente um aspeto que vai além do instrumental. A estética torna-se importante por causa do seu valor intrínseco (Postrel, 2003), que ecoa o facto de que a beleza satisfaz uma necessidade geral do ser humano, principalmente na camada da autorrealização, no inglês *Self-fulfillment,* como se poderá ver na Figura 1 (Maslow, 1954).

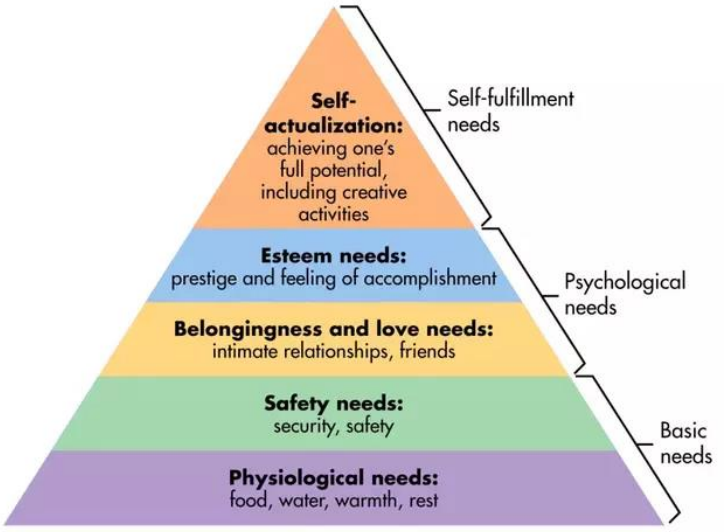

Figura 1 – Pirâmide da hierarquia das necessidades humanas[1]

Em (Gaver & Martin, 2000), (Logan, et al., 1994) e (Hassenzahl, 2003), vemos como aspetos mais abstratos e emocionais, como a intimidade e a surpresa, ou hedónicos, como a estimulação (p. ex. crescimento pessoal), a identificação (p. ex. a autoexpressão) e evocação (p. ex. memórias) não devem ser descartados numa tentativa puramente pragmática de ver a área da HCI.

Este modelo multidimensional relaciona atributos de um produto com necessidades e valores. Todas estas abordagens têm um objetivo comum: enriquecer modelos atuais de qualidade do produto com aspetos não instrumentais para criar uma HCI mais holística (Hassenzahl & Tractinsky, 2006).

[1] Disponível em: https://www.simplypsychology.org/maslow.html. Acesso em 2019

2.4.2 Emoção e afeto

A Investigação atual dá ênfase à importância de um sistema afetivo para um largo leque de processos centrais, tais como a tomada de decisões (Loewenstein & Lerner, 2003) ou o bem-estar subjetivo (Suh, et al., 1996). O projeto "*Affective Computing"* foi uma das tentativas pioneiras onde o afeto na HCI foi referido (Picard, 1997). Esta tentativa chamou a atenção para a importância do afeto e emoções. No entanto, a computação afetiva toma a perspetiva do computador. Ela lida principalmente com questões sobre como um computador conseguirá detetar afeto por parte do utilizador, como conseguirá adaptar-se a ele e como conseguirá demonstrar afeto em resposta ao utilizador. Por exemplo em (Cockton, 2002) vemos vários exemplos de como sistemas interativos podem ajudar utilizadores irritados, fazendo uma gestão das suas frustrações e prevenir outras emoções negativas. Neste paradigma, os investigadores preveem, por exemplo, brinquedos computorizados capazes de acalmar crianças que choram (Picard & Klein, 2002).

A área de estudo de UX toma o ponto de vista do utilizador, ao contrário da área anteriormente referida, que toma o ponto de vista do computador. A área de UX está interessada em compreender o papel do afeto como um antecedente, uma consequência e um mediador no uso de tecnologia. Também, em contraste com a abordagem do "*Affective Computing"* em que emoções negativas têm mais relevo, a área de UX foca-se principalmente em emoções positivas. O que é novidade, na investigação em UX, é o foco em emoções positivas como a alegria, diversão e orgulho.

Um exemplo que realmente tenta cultivar experiências emocionais positivas é a ferramenta *Gustbowl* (van der Hoog, et al., 2004). Esta ferramenta foi desenhada para conectar pais e crianças. A relação entre estes é predominantemente emocional e construída em cima de rituais afetivos. O objetivo desta ferramenta é o de estabelecer, de uma forma não intrusiva e contínua, uma comunicação puramente afetiva entre pais e filhos que estejam separados de alguma forma. O *Gustbowl* é uma taça que envia uma imagem do que lá for posto para uma outra taça emparelhada. Um exemplo é o de quando o pai pousa as chaves sempre que chega a casa. A taça emparelhada vibra notificando que as chaves foram pousadas. O *Gustbowl* explora o ritual do pai chegar a casa. Uma filha que viva fora de casa seria capaz de reviver estas memórias, invocando um sentimento de nostalgia, sem a necessidade de um ato explícito de comunicação.

Em (Desmet, et al., 2001) é demonstrado como o afeto se pode tornar num objetivo de design. A equipa tentou criar um telemóvel que estivesse adaptado à resposta afetiva preferida pelo utilizador. Todos os utilizadores tiveram preferência por uma resposta afetiva positiva, em níveis diferentes. Uns tiveram preferência por um telemóvel que evocasse uma resposta mais calma, enquanto que outros tiveram preferência por um que evocasse uma mais excitante. O resultado desta experiência foi atingido através de um processo gradual, em que ambos os grupos de participantes obtiveram um protótipo que ia de encontro às suas expectativas afetivas.

Em termos gerais há duas formas básicas de lidar com emoções em UX (Hassenzahl, 2006): uma linha de investigação tem como mais importante as emoções como consequências da utilização de um produto (p. ex. (Kim & Moon, 1998)); a outra linha concentra-se na importância de emoções como antecedentes à utilização de um produto e como este influencia o utilizador.

2.4.3 Experiência

A perspetiva experiencial sobre UX dá ênfase a dois aspetos da utilização de tecnologia: o contexto situacional e a temporalidade. Nesta visão, uma experiência é a combinação única de vários elementos, como os do produto e os estados internos do utilizador (o seu estado de espírito, expectativas, etc.), que estendem sobre um período, com um início e fim bem definidos. O experiencial assume que todos estes elementos estão interligados, que interagem e se alteram mutuamente. O resultado deste processo é a própria experiência. Um exemplo muito prático é, por exemplo, considerar a diferença entre "ter um tomate no frigorífico" e "o sabor de um molho de tomate numa pizza caseira", ou "um livro na prateleira" e "um *thriller* misterioso cuja história nos mantém acordados a noite inteira". Os produtos, neste caso o tomate e o livro, são usados numa situação muito particular, e que depois forma uma experiência.

Experiências têm vantagens sobre resultados materiais, porque têm um maior impacto positivo no bem-estar de um indivíduo (Van Boven & Gilovich, 2003). Estas experiências possuem qualidades afetivas que ajudam a transformar e a regular o estado afetivo de um indivíduo. Por consequência, será muito vantajoso dar ênfase ao experiencial em produtos interativos em detrimento do material.

Em (Forlizzi & Battarbee, 2004), os investigadores vão um passo mais à frente e fazem a distinção entre "Uma experiência" e "Experiência": a primeira pode ser articulada, tem um início e um fim e tende a resultar em alterações emocionais e comportamentais; a segunda é um fluxo constante do monólogo interno, que acontece quando interagimos com um produto. O primeiro termo estabelece o experiencial como algo complexo, único e por consequência algo de grande relevo e difícil de repetir. O segundo termo tem como importante o aspeto temporal da experiência.

As duas perspetivas levantam muitos desafios e questões sobre como será possível lidar com esta aparente complexidade ligada à experiência. Será que *designers* podem exercer controlo tal sobre todos os elementos relevantes, que uma experiência positiva seja certa? Ou será que devemos "desenhar para uma experiência", isto é, ter em conta aspetos experimentais enquanto desenhamos sem conseguir garantir uma certa experiência?

Em suma, cada uma destas três facetas da investigação sobre UX, contribui para o nosso entendimento sobre a área. Isto, no entanto, significa que nenhuma consegue encapsular na totalidade. UX é sobre tecnologia que cumpre mais do que necessidades instrumentais; é a consequência do estado interno do utilizador (predisposições, expectativas, necessidades,

motivações etc.), as características do sistema desenhado (p. ex. complexidade, propósito, usabilidade, funcionalidade, etc.) e o contexto no qual a interação ocorre.

O trabalho a realizar terá em vista os conceitos explorados ao longo destes subcapítulos, na medida em que não se dará apenas importância aos aspetos funcionais da aplicação, mas também à usabilidade. O cliente fará parte da fase de design, segundo os princípios do Design Centrado no Utilizador, garantindo o seu *input* nesta fase inicial que influenciará o desenvolvimento mais tarde.

2.5 Tecnologias

2.5.1 Unity 3D e Unreal Engine

O motor de jogo Unity é desenvolvido pela Unity Techonolgies baseada na Dinamarca. Este integra um motor de renderização com o motor de física PhysX da Nvidia e uma implementação *open-source* das bibliotecas .Net da Microsoft denominado por Mono.

Este motor de jogo tem várias vantagens que serão listadas de seguida, (Craighead, et al., n.d.):

- Tem uma documentação muito rica em exemplos e explicações para a API toda. Isto promove uma maior produtividade quando comparado com outros motores como o Unreal Engine que só fornece uma documentação parcial para clientes que não pagam.

- A comunidade é muito ativa o que promove a entreajuda especialmente entre utilizadores experienciados e utilizadores novos. Os desenvolvedores responsáveis pela manutenção do motor de jogo prestam atenção às necessidades da comunidade e desenvolvem funcionalidades em função dos pedidos recebidos. O *Unreal Engine* apresenta também uma comunidade ativa apesar de ser mais pequena fruto dos custos elevados associados às licenças disponíveis.

- Em termos de interface é também fácil e intuitivo de usar. Os objetos são inseridos em cena com ações de *drag-and-drop*. Os utilizadores conseguem facilmente acrescentar *scripts* a cada entidade para associar comportamentos interativos, criar interfaces ou simplesmente para gerir informação.

- O motor de física torna a implementação de características complexas como deteção de colisões, massa, resistência e elasticidade extremamente acessíveis e simples de introduzir.

- O motor apresenta uma distribuição multiplataforma permitindo a compilação dos jogos para OSX (MacOS), Windows ou para aplicações web, em semelhança ao Adobe Flash.

- Em termos de custo, este motor é bastante mais barato quando comparado a motores como o Unreal Engine. Licenças variam entre os 266€ por ano para desenvolvedores independentes e 1330€ por ano para profissionais.

Como se pôde verificar nos casos de estudo (Katz, et al., 2011), (Merlo, et al., s.d.) e (Wang, et al., 2010) o Unity3D é de facto uma ferramenta poderosa e pode ser usada num ambiente Web e oferece soluções flexíveis que não dependem de uma instalação demorada por parte do utilizador final.

O motor Unreal Engine é desenvolvido pela Epic Games, autora de títulos como o *Unreal Tournament* e o *Fortnite*. Este é uma ferramenta poderosa capaz de produzir desde experiências cinemáticas de alta qualidade a jogos em qualquer plataforma, como os referidos anteriormente.

Ao comparar ambos os motores, facilmente vê-se que são ferramentas poderosas. Mas não são iguais. Com o *Unreal Engine* consegue-se atingir resultados melhores, fruto de um leque mais alargado de funcionalidades disponíveis. A principal desvantagem deste motor são os custos associados reduzindo a sua acessibilidade (Petridis, et al., 2010).

2.5.2 JavaScript e Modelação 3D

JavaScript é uma linguagem importante porque é a linguagem dos *browsers*. Esta associação faz com que o JavaScript seja uma das linguagens mais populares da internet e ao mesmo tempo uma das mais desprezadas (Crockford, 2008).

Fruto do posicionamento do JavaScript no mundo do desenvolvimento em ambiente Web a quantidade de *frameworks* que se baseiam é deveras grande. Estimou-se, em 2017, que existiam mais de 10 milhões de desenvolvedores, sendo o número possivelmente maior na atualidade (Voss, 2018).

Neste âmbito, há muitas abordagens possíveis no que toca a tecnologias Web nativas. Como se pode verificar em (Slant, 2019), existem várias opções de onde escolher uma solução para desenhar gráficos ou modelar em três dimensões num ambiente Web baseando-nos no JavaScript.

WebGL

O WebGL é uma API (do inglês *Application Programming Interface*) sem direitos de autor e é uma multi-plataforma que traz o OpenGL ES 2.0 (uma adaptação do standard OpenGL) para a Web como um contexto de desenho tridimensional contido no HTML e é desenvolvida e mantida pelo Khronos Group, o grupo responsável por estabelecer standards que também governa outros sistemas como o OpenGL. Esta tecnologia é indicada para aplicações dinâmicas na Web, mais especificamente para a linguagem JavaScript, e é integrada na sua totalidade

pelos exploradores da internet (em inglês web *browsers*) mais populares (Parisi, 2012). Neste caso de estudo o WebGL é usado para gerar resultados tridimensionais complexos em tempo real tanto em computadores como em dispositivos móveis.

Os motores de jogo referidos anteriormente, o Unity 3D e o Unreal Engine, quando usados em ambiente web usam um compilador próprio para apresentar os resultados desejados através do WebGL. Este serve, portanto, como uma ponte muito importante no âmbito da modelação tridimensional em ambiente Web.

Há muitas *frameworks* de WebGL que estão disponíveis para abstrair as APIs de nível mais baixo. Esta abstração ajuda a fazer o desenvolvimento em WebGL mais fácil e mais produtivo. Uma *framework* popular será discutida de seguida.

Three.js

Esta ferramenta fornece vários modos de desenho e pode recuar das três para as duas dimensões se o WebGL não for suportado. O *Three.js* é uma biblioteca bem desenhada e intuitiva de se usar. Definições por omissão reduzem o trabalho inicial podendo ser customizadas consoante a experiência do desenvolvedor para soluções mais minuciosas.

O *Three.js* foi criado por Ricardo Cabello, e existe no *GitHub*, um repositório popular, desde 2010. A sua popularidade tem vindo a crescer ao longo dos anos e já conta com adições para a sua documentação de vários contribuidores.

Algumas das características mais relevantes do *Three.js* são as seguintes (Danchilla, 2012):

- Recua para um contexto de duas dimensões se o WebGL não for suportado;
- Operadores de vetores e matrizes nativos;
- Implementações de API de alto nível relativos a câmaras, luzes materiais e outras entidades relevantes à criação de cenas;
- Documentação rica em exemplos e explicações esclarecedoras.

Konva.js

O Konva.js é uma *framework* baseada também na linguagem do JavaScript cujo objetivo principal é de representar graficamente objetos definidos pelo utilizador.

Esta *framework* oferece uma grande performance a nível de animações, transições e manipulação de nós facilitando a representação de figuras geométricas em ambiente Web. O Konva.js apresenta dois tipos de telas (no inglês *canvas)*: o principal, representado pela etiqueta "<canvas>" na página Web onde todas as figuras são representadas e manipuladas, e outra que, sendo invisível, é usada para a deteção de eventos.

É neste primeiro *canvas* que encontramos o objeto principal desta *framework*: o “palco” (do inglês *Stage*). É a partir de este objeto que todos os outros (camadas, grupos e figuras) estão contidos, comparável aos nós de DOM (*Document Object Model*) numa página HTML (Lahti, 2016), (Anton, 2018).

3 Análise de Valor

Lawrence D Miles define a Análise de Valor no seu livro "Técnicas de Engenharia de Análise de Valor" como sendo um sistema de solução de problemas implementado pelo uso de um conjunto de técnicas específicas, um corpo de conhecimento e um conjunto de capacidades aprendidas. É uma abordagem criativa e organizada que tem como propósito a identificação eficiente de custos desnecessários, ou seja, os custos que não oferecem qualidade, utilização, vida, aparência, nem características ao cliente (Miles, 2015).

3.1 Modelo New Concept Development

Com o intuito de analisar o valor da plataforma proposta no presente documento será usado o modelo NDC (*New Concept Development*). Também será apresentado o modelo *Canvas* para discriminar as várias áreas de interesse para fundamentar a viabilidade de negócio para a comercialização do projeto.

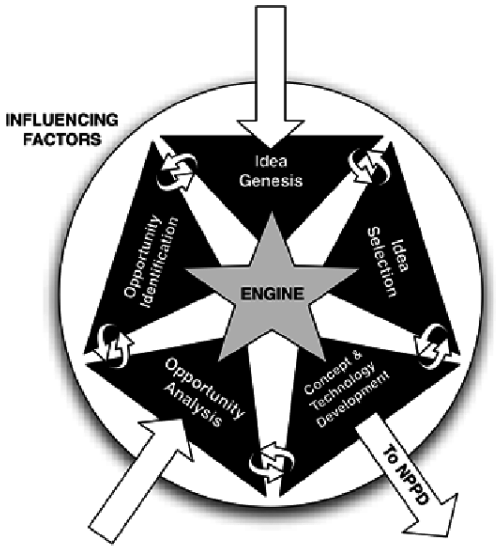

Figura 2 – Modelo NCD (*New Concept Development*)

Segundo Dewulf no modelo NCD, a componente do motor representa a liderança, cultura e estratégia de negócio de uma organização que impulsiona os cinco elementos chave que serão explorados de seguida (Dewulf, 2013).

3.1.1 Identificação da Oportunidade

A primeira atividade consiste em identificar oportunidades no negócio ou no âmbito tecnológico em que seja importante investir através de métodos mais ou menos estruturados. Segundo Koen estes métodos vão de ferramentas formais mais sistemáticas, como diagramas em forma de espinha de peixe, investigação do mercado e planeamento de cenários futuros a abordagens *ad hoc* como conversas informais e perspetivas individuais (Koen, et al., s.d.).

Neste caso, a oportunidade foi identificada através da pesquisa de mercado onde uma falha na presença de soluções de *software* de paletização tornou-se aparente. Estes ou eram incompletos nas suas funcionalidades, em específico na manipulação manual de caixas, ou não se aplicavam aos objetivos inicialmente delineados. Também, a integração com sistemas externos já existentes é um fator fulcral, tornando alternativas existentes no mercado pouco apelativas.

3.1.2 Análise da Oportunidade

Esta segunda atividade envolve uma análise mais profunda da oportunidade identificada por confirmar que de facto vale a pena investir nela. Para este fim, os mesmos métodos para a identificação de oportunidades podem ser usados, mas em maior detalhe.

Havendo sido identificada a oportunidade é necessário analisá-la do ponto de vista do cliente: é imprescindível apresentar uma solução desejável do ponto de vista do negócio. É importante conseguir demonstrar ao cliente que esta é uma oportunidade vantajosa ao negócio apresentando as mais-valias nomeadamente o de um controlo grande sobre as soluções produzidas automaticamente por módulos externos, e ferramentas de criação de soluções que sejam fáceis de usar e ao mesmo tempo completas reduzindo o tempo necessário para cada tarefa.

3.1.3 Génese da Ideia

Este é um processo iterativo e evolutivo onde o desenvolvimento e amadurecimento de ideias concretas ocorre. Ferramentas formais como secções de brainstorming podem ser usadas, mas processos informais como pedidos inesperados por utilizadores ou resultados de experiências podem moldar as ideias geradas.

Após a identificação e análise da oportunidade, duas alternativas foram identificadas quanto à forma de como a oportunidade vai ser aplicada: numa aplicação *Desktop* ou numa plataforma Web. O principal fator que distingue estas duas metodologias é a responsabilidade de executar os processos computacionais: no primeiro é necessário assegurar que todos os terminais de acesso ao sistema possuem da capacidade de lidar com toda a carga computacional sem pôr em causa a experiência do utilizador; no segundo o terminal é apenas um ponto de acesso a um computador central que detém toda a responsabilidade por computar todos os pedidos e operações, assumindo por isso que todos os terminais têm que estar ligados a este servidor para assegurar o correto funcionamento do sistema.

3.1.4 Seleção da Ideia

Depois de gerar ideias é necessário decidir quais suportar dado o tempo e orçamento limitados. Isto é feito por avaliar quais ideias maximizam o rácio valor/custo usando, tipicamente, processos formais como metodologias de portfólio e processos formais de seleção de procedimentos que solicitam a opinião a quem submete a ideia.

Nesta fase, o coordenador do projeto teve um papel crucial ao manter um contacto próximo com cada membro da equipa tomando em consideração tanto as sugestões relacionadas com as tecnologias como os custos associados ao próprio desenvolvimento.

Das duas alternativas definidas em cima decidiu-se seguir com a do desenvolvimento Web devido à grande vantagem de conseguir ter todos os pontos de interação em dependência de um servidor central. Nesta abordagem, estes pontos de interação não terão que lidar com cargas computacionais tão pesadas, pelo que apenas terão que apresentar o website, onde a plataforma estará disponível.

3.1.5 Desenvolvimento do Conceito e Tecnologia

Este sendo o estágio final do novo modelo de desenvolvimento do conceito, deve ser feito um caso muito forte para a proposição tecnológica ou de negócio, com informação qualitativa e quantitativa como objetivos, analise de fatores de risco e o tamanho da oportunidade.

Assim, o conceito do trabalho apresentado será o de desenvolver uma plataforma que servirá de interface do trabalhador com o chão de fábrica em que trabalha, oferecendo todos os controlos e dados pertinentes ao bom funcionamento do mesmo.

3.2 Proposta de Valor

Este projeto consiste de uma plataforma de gestão de chão de fábrica e em particular um módulo que permite a criação e manipulação de soluções de paletização, denominadas por mosaicos. Esta plataforma vai permitir aos seus utilizadores que tenham uma visão detalhada sobre todos os aspetos da fase de paletização centralizando toda gestão das entidades pertinentes ao processo, desde criação e edição de paletes e produtos, controlo dos robôs e tapetes rolantes, gestão de alertas e geração, manipulação e pré-visualização de mosaicos.

Este último aspeto relativo à manipulação e pré-visualização de mosaicos torna-se extremamente importante e útil ao utilizador porque vai permitir reduzir significativamente os riscos que uma solução defeituosa pode trazer, tornando todo o processo de correção das soluções apresentadas mais célere e conveniente. O utilizador será capaz de modificar minuciosamente as soluções conforme o resultado esperado tendo a ajuda ativa do sistema para evitar erros.

3.3 Modelo de Negócio *Canvas*

Com o objetivo de sumarizar o plano de negócios, o modelo de negócios *Canvas* vai ser apresentado na Tabela 1.

Tabela 1 – Modelo de negócio Canvas

<table>
<tr>
<td rowspan="2">Parceiros Chave

INESC TEC</td>
<td>Atividades Chave

Gestão de recurso do chão de fábrica

Apoio no processo de criação e correção de soluções de paletização.

Pré-visualização do produto final do processo da paletização</td>
<td rowspan="2">Proposta de Valor

Interface Simples e útil

Simplificar os processos existentes

Maior deteção de erros</td>
<td>Relação com o Cliente

Tratamento personalizado

Utilização da plataforma em qualquer lugar

Suporte privilegiado 24/7</td>
<td rowspan="2">Segmento de cliente

Indústria de manufatura</td>
</tr>
<tr>
<td>Recursos Chave

Apoio ao cliente;

Desenvolvedores qualificados em desenvolvimento web e de bases de dados.</td>
<td>Canais de Distribuição

Contacto direto

Plataforma Web</td>
</tr>
<tr>
<td colspan="3">Estruturas de Custo

Colaboradores: Engenheiros de Software

Hardware: Para suportar o desenvolvimento da solução</td>
<td colspan="2">Fluxo de Receitas

Venda do produto

Venda de novas funcionalidades</td>
</tr>
</table>

3.4 Sacrifícios e Benefícios

Na Tabela 2 são demonstrados os benefícios e sacrifícios para o cliente relativamente ao produto, serviço e relacionamento

Tabela 2 – Tabela dos Sacrifícios e Benefícios

Domínio / Âmbito	Produto	Serviço	Relacionamento
Benefício	Qualidade do produto.	Qualidade do serviço Aumento da eficiência dos processos da empresa; Competências técnicas; Usabilidade; Fiabilidade.	Acesso à plataforma em qualquer lugar; Rapidez para criar, manipular e corrigir soluções de paletização; Confiança; Solidariedade.
Sacrifício		Preço para adquirir licença; Tempo para aprender a utilizar as ferramentas.	

3.5 Análise SWOT

Na Tabela 3 são apresentados os fatores internos e externos que influenciam o projeto:

Tabela 3 – Análise SWOT

FATORES INTERNOS	
FORÇAS (+)	**FRAQUEZAS (-)**
Interface Simples e útil Controlos intuitivos	Pouca flexibilidade Mercado nicho
FATORES EXTERNOS	
OPORTUNIDADES (+)	**AMEAÇAS (-)**
Empresas que implementem processos de paletização	Produtos que apresentam funcionalidades semelhantes

4 Análise e Design

Neste quarto capítulo, vão ser abordados os aspetos mais técnicos da solução a desenvolver, desde o levantamento de requisitos funcionais e não funcionais às decisões de design tomadas, tanto da arquitetura como da interface.

4.1 Análise de requisitos

Para garantir um desenvolvimento eficiente e eficaz, é sempre necessário saber pormenorizadamente que características finais o produto tem de ter, e quanto mais informação se tiver no início do desenvolvimento, mais rápido o processo será. A esta fase dá-se o nome de levantamento de requisitos. Esta fase é de suma importância, porque afeta todo o desenvolvimento. É necessário conversar com o cliente para saber exatamente o que este espera da aplicação.

Falhas ao longo do processo são inevitáveis, porque tudo isto depende da capacidade de comunicação tanto por parte do cliente, que pode explicar algo de forma incompleta ou errada, como por parte de quem vai desenvolver a aplicação, que pode entender de uma forma errada o que o cliente deseja. A maior barreira é a linguagem.

A melhor forma de minimizar os erros neste processo é a comunicação iterativa. Ao mostrarmos o que foi desenvolvido ao fim de uma sprint vamos conseguir corrigir eventuais erros, sem que estes envolvam reestruturações arquiteturais, o que resultaria em atrasos desnecessários. Ao longo deste processo foram delineados requisitos funcionais e requisitos não funcionais.

4.1.1 Conceitos

Para uma melhor compreensão do problema, será feita de seguida uma apresentação dos conceitos mais importantes:

- Produto e Palete: Representam as entidades basilares do Problema da Paletização, que consiste na otimização da disposição de produtos, em caixas, em cima de paletes. O produto é abstraído a uma caixa tridimensional com um certo peso: assume-se que o peso está distribuído homogeneamente pelo volume. Da mesma forma, em relação à palete, as únicas características relevantes são as dimensões e o peso;

- Grupo: Representa um conjunto de caixas que pertencem a uma camada e existe em dois contextos diferentes. Os grupos da solução servem de referência para as transformações aplicadas aquando da criação de uma receita;

- Camada: Representa um conjunto de grupos à mesma altura;

- Solução: Consiste numa disposição, definida pelo utilizador ou automaticamente, de produtos em cima da palete. Contém apenas um conjunto de grupos que vão conter as coordenadas dos produtos;

- Mosaico: Esta entidade representa uma combinação de um produto e de uma palete tendo também como característica importante a altura máxima permitida. Um mosaico deverá ter pelo menos uma solução;

- Receita: Contém um mosaico, uma referência a uma solução do mosaico selecionado e um conjunto de camadas. Estas consistem em cópias da solução, à qual foi aplicado uma combinação de transformações de rotação ou espelho. Tem também um conjunto de atributos relativos à velocidade a que pode ser transportado, mas que serão irrelevantes para a presente tese.

Com base nestes conceitos, foi desenhado o diagrama do Modelo de Domínio presente na Figura 3. Este diagrama representa uma vista de alto nível das entidades referidas anteriormente. O Sistema de visualização e manipulação de soluções de paletização será integrado numa plataforma mais complexa, mas para efeitos de simplificação, foram abstraídos conceitos por não serem estritamente necessários.

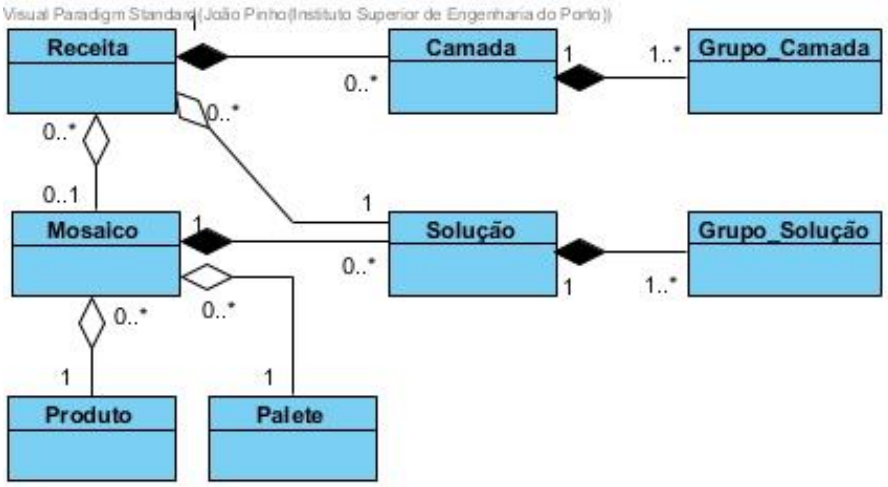

Figura 3 – Modelo de domínio

Apesar de este diagrama não ser de classes, incluíram-se as ligações de composição e agregação para detalhar mais as relações entre os conceitos. Estas relações indicam qual o grau da relação entre duas entidades:

- Composição: representada pelo losango preto, significa que há uma grande dependência entre os ciclos de vida das duas classes. Por exemplo, no caso do Mosaico que contém uma ou várias Soluções, se o primeiro for apagado não fará sentido existirem soluções, e por isso também serão eliminadas;

- Agregação: representada pelo losango branco, significa que não há dependência entre os ciclos de vida das duas classes. Por exemplo, entre o Mosaico e o Produto, a criação de um mosaico depende da existência de produtos, no entanto se o primeiro for apagado, os produtos não são afetados.

De seguida serão apresentados os requisitos funcionais, e não funcionais, levantados nesta fase segundo o modelo FURPS+ (do inglês *Funcionality, Usability, Reliability, Performance, Supportability*). O resultado mais detalhado pode ser visto no Anexo B – QEF onde todos os requisitos serão discriminados em grande detalhe, a fim de serem o padrão de uma das avaliações do produto a desenvolver.

4.1.2 Requisitos Funcionais

Diagrama de Fluxo da Aplicação

O processo de criação de uma receita tem que, necessariamente, passar pela criação das outras entidades. Esta ordem será apresentada de seguida na Figura 4:

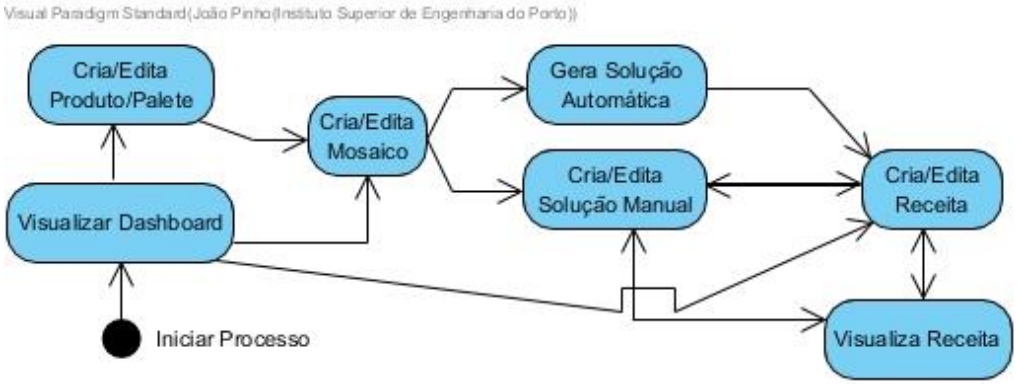

Figura 4 – Diagrama de fluxo da aplicação

O processo de criação de uma receita tem de seguir esta ordem: se o utilizador quiser criar uma receita de raiz, primeiro terá de criar um produto e uma palete seguido de um mosaico e, pelo menos, uma solução; o utilizador poderá também editar um mosaico, e as suas soluções, ou uma receita já existente. Após cada alteração, tanto a uma solução como a uma receita, espera-se que o utilizador valide os resultados através do visualizador de receitas.

A presente tese está focada na criação e edição de soluções manuais e na posterior visualização da receita já construída.

Características de Apresentação

Em ambos os componentes, o utilizador deverá ter total controlo da vista, conseguindo movê-la (*pan*) e aproximá-la (*zoom*).

No manipulador de soluções, o utilizador deverá ter acesso às coordenadas e dimensões de cada caixa.

Em relação ao componente de visualização em três dimensões, este deverá permitir a seleção de cada camada e deverá mostrar as informações relativas às transformações, no caso da camada, e do grupo, no caso da caixa.

Características da Manipulação de Caixas

Durante a criação e edição de soluções, o utilizador vai precisar de um conjunto de operações que facilitem o posicionamento e ajuste das caixas. O objetivo principal destas operações será o de reduzir ao máximo a necessidade de manipular as caixas individualmente. No entanto, a criação e manipulação das soluções também deverá ser auxiliada por guias aquando do arrasto das caixas, sendo que, se for detetada alguma colisão esta deverá ser sinalizada. Estas operações serão divididas em três conjuntos maiores, nomeadamente o do alinhamento, o da distribuição e o da rotação:

- Alinhamento às margens da palete;

- Alinhamento a um eixo (horizontal ou vertical) da palete;
- Alinhamento ao extremo da seleção;
- Distribuição equidistante;
- Rotação de caixas em torno dos seus centros;
- Rotação de caixas em torno do centro da palete.

4.1.3 Requisitos Não Funcionais

De seguida, na Tabela 4, serão apresentados os requisitos não funcionais. Estes vão ser representados continuando a seguir o modelo de FURPS+:

Tabela 4 – Requisitos Não Funcionais

Requisito	Especificação
Usabilidade	A estética da plataforma terá de ser consistente, concisa e clara. As operações de manipulação de caixas terão de ter ícones minimalistas e representativos da ação executada. As operações terão de ter etiquetas do tipo "*tooltip*" para facilitar a memorização das mesmas. A disposição das operações de manipulação terá de ser intuitiva para facilitar a memorização das mesmas O alinhamento manual de caixas terá de ser assistido por linhas guias.
Confiabilidade	O sistema deverá ter mensagens adequadas para assistir o utilizador em caso de falha. Erros do sistema deverão ser lidados de forma graciosa.

Desempenho	Tanto a componente de manipulação 2D, como a componente de visualização em 3D, deverão proporcionar uma experiência sem atritos ao utilizador. As operações de manipulação deverão ser executadas sem qualquer impacto visível na performance do sistema. A fim de evitar potenciais problemas, o sistema deve estar preparado para que o tempo de resposta seja sensivelmente o mesmo, independentemente da carga existente.
Suportabilidade	A plataforma deverá suportar todos os *browsers* mais comuns (*Internet Explorer, Edge, Firefox e Chrome*). A plataforma deverá suportar pelo menos a língua portuguesa e inglesa.
+	
Limitações de Interface	O sistema deverá estar preparado para receber informações do módulo "Solver"
Limitações de Design	Devem ser adotadas boas práticas de design (e.g. SOLID)

4.2 Design

Nesta secção, será descrito como a aplicação deverá ser moldada aos requisitos propostos pelo cliente. Nesta fase, o foco principal deverá ser o de traduzir os requisitos e conceitos recolhidos na fase da Análise (cf. 4.1) numa solução técnica, tendo por base boas práticas de engenharia informática.

Inicialmente, serão apresentadas as decisões mais relevantes quanto ao design da arquitetura da aplicação como um todo, uma descrição do componente de visualização 3D e, de seguida, uma descrição do componente de manipulação 2D.

4.2.1 Diagrama de Implantação

Nesta secção será descrita a implementação da plataforma. De seguida, será apresentado o diagrama de implantação, na Figura 5, que será acompanhado de uma breve descrição de cada nó:

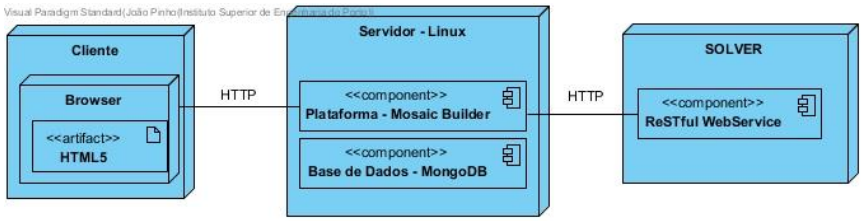

Figura 5 – Diagrama de Implantação

- Plataforma – Mosaic Builder: esta é a plataforma principal onde o utilizador fará a gestão das receitas, mosaicos, produtos e paletes;

- Solver – Este componente, desenvolvido noutra instituição, contém um serviço que devolve soluções de paletização de uma forma automática.

As componentes da Base de Dados e da Solver serão abstraídas ao máximo, pelo que, já tendo sido desenvolvidas de antemão, não fazem parte do âmbito desta dissertação.

4.2.2 Arquitetura Geral da aplicação

A aplicação será desenvolvida em AngularJS, uma *framework* baseada em JavaScript. O seu funcionamento pode ser visto de uma forma mais abstrata, como está representada na Figura 6.

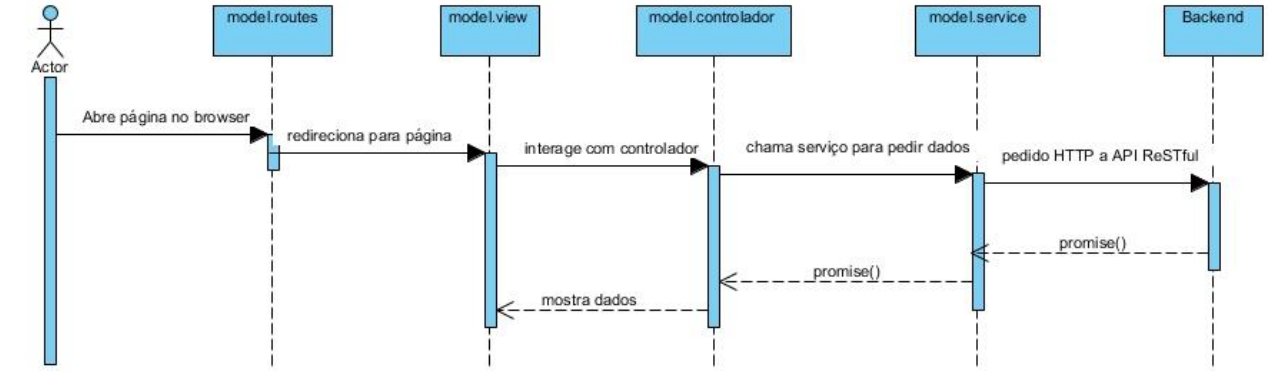

Figura 6 – Diagrama de sequência AngularJS

Nesta figura, podemos ver como uma página é apresentada ao utilizador em duas fases, ainda que estas possam ser quase impercetíveis. O utilizador poderá abrir a página principal, ou outra qualquer, sendo logo mostrada a página base. Será só nesse momento que o *browser* vai fazer o pedido à base de dados, para popular a página apresentada em tabelas, por exemplo.

Este comportamento estará presente na aplicação em todas as interações entre o utilizador e o browser, seja para abrir uma página nova, ou para carregar informação presente nesta em tempo real. Para atingir esta experiência mais suave todos estes pedidos serão, portanto, assíncronos.

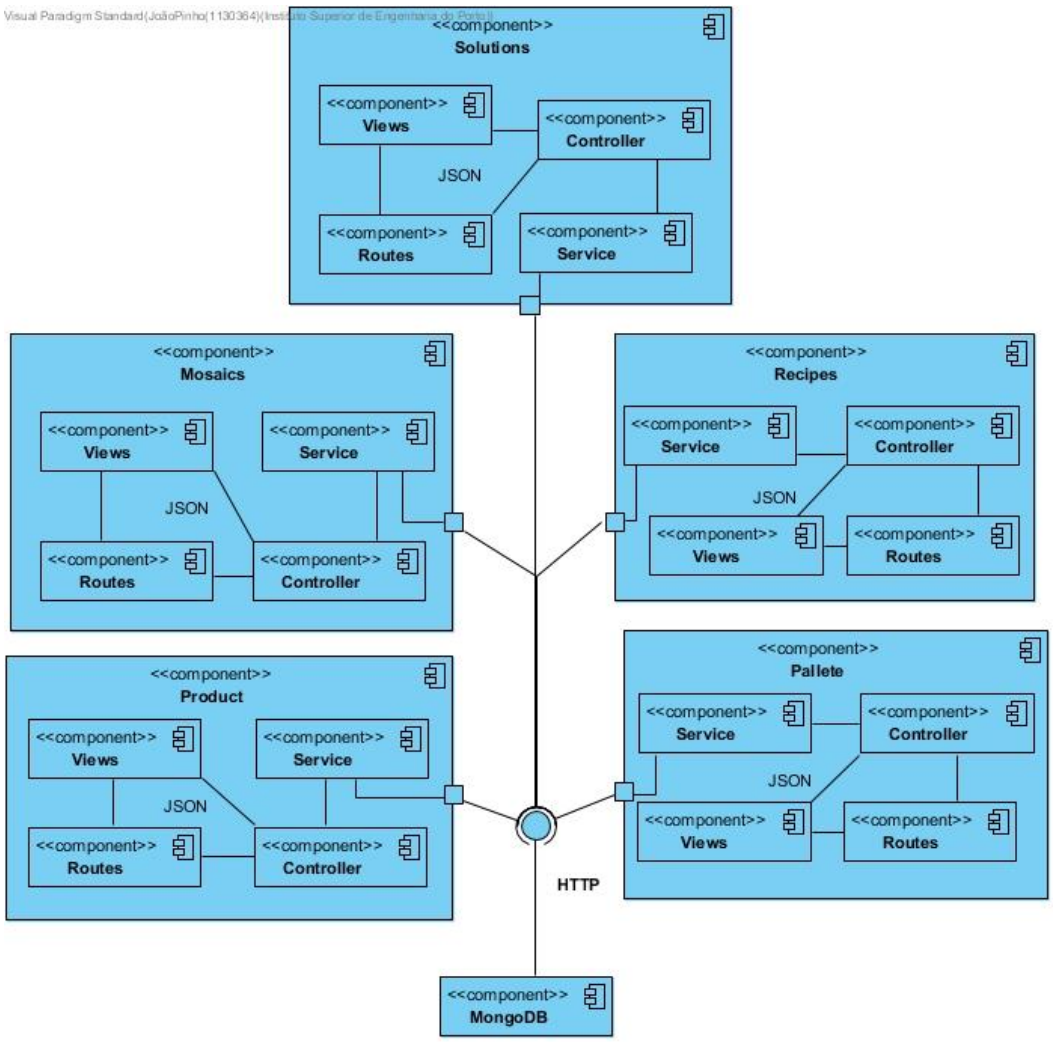

Figura 7 – Diagrama de Componentes

Como se poderá ver na Figura 7, todos os módulos do projeto serão compostos por estes componentes, que comunicam entre si usando JSON:

- *Routes* (rotas): Este componente é responsável por ligar cada página ao seu controlador responsável, tendo em conta o URL inserido no *browser*;

- *Views* (vistas): Este representa uma página que recebe informação de forma assíncrona do controlador;

- *Controller* (controlador): O controlador é responsável por comunicar com o serviço e transformar os dados de modo a que a vista os possa apresentar;

- *Service* (serviço): Este componente é responsável por comunicar com a base de dados através de pedidos HTTP.

De seguida, será apresentado em maior detalhe o design da arquitetura dos dois componentes: primeiro, o visualizador 3D, e depois o manipulador 2D.

4.2.3 Arquitetura do Componente de Visualização 3D

Este componente deverá ser desenvolvido de forma a que a aplicação principal não tenha de lidar com alterações, sempre que alguma seja feita nele, de maneira a cumprir os princípios e boas práticas da alta coesão e baixo acoplamento. O ideal será garantir que a interface entre a aplicação e o componente seja o mais simples possível, de preferência apenas um método para inicializar e outro para terminar, mantendo todas as regras de negócio dentro do próprio componente.

Na Figura 8 pode ver-se como deverá ser o fluxo de funcionamento do componente. A aplicação deverá ter um componente que servirá de *wrapper*, que encapsulará toda a interação com o Mosaic3DViewer. Aqui será feita toda a comunicação com este componente: os dados serão passados para ele juntamente com a referência do elemento HTML. Desta forma, a responsabilidade de gerar a página HTML estará totalmente do lado da aplicação, e a de gerar a cena estará totalmente do lado do componente (1.1).

O componente, ao ser inicializado, deverá gerar todas as entidades necessárias ao seu funcionamento, que tendo em conta a documentação do Three.JS deverá incluir a câmara, controlos, iluminação, texturas e cenário (1.1.1.1 a 1.1.1.7).

Só após a inicialização destes objetos, mais básicos e fundamentais, é que se poderá avançar para a criação dos objetos finais, através da manipulação dos dados recebidos (1.1.1.7 a 3).

Com base na documentação da biblioteca (mrdoob, et al., 2019) vemos também que será necessário implementar um ciclo recursivo para desenhar a cena (4 a 4.2), que deverá ser interrompido aquando do término do processo.

Diagrama de Sequência – Mosaic3DViewer

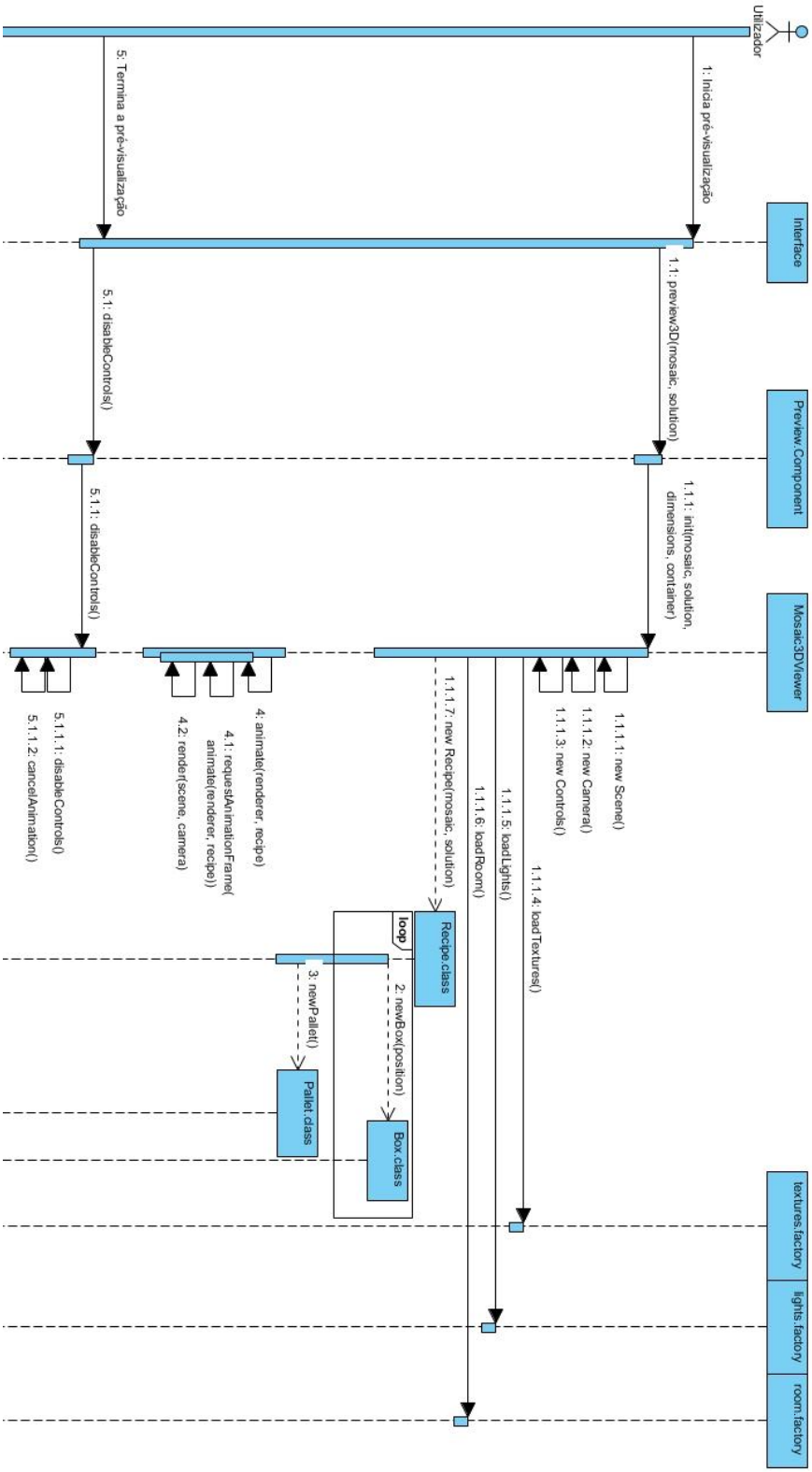

Figura 8 – Diagrama de sequência do visualizador 3D

Diagrama de Classes – Mosaic3DViewer

De seguida será apresentado o diagrama de classes deste módulo, onde cada classe será explicada em maior detalhe, na Figura 9.

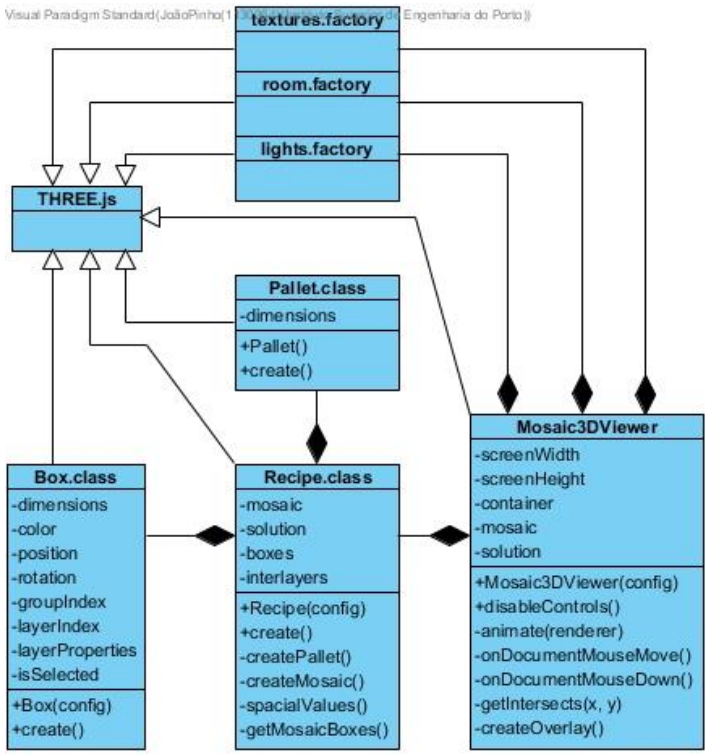

Figura 9 – Diagrama de classes – Mosaic3DViewer

A classe principal, Mosaic3DViewer, será responsável por instanciar todo o ambiente do Three.JS. Esta é a única classe que interage com a aplicação principal, e irá receber por parâmetros apenas o estritamente necessário: as dimensões do painel (“*screenWidth*” e “*screenHeight*”); o elemento HTML onde o painel vai estar agregado e os dados necessários (container); o mosaico com as dimensões da palete e do produto (*mosaic*) e a solução com as coordenadas das caixas (*solution*). Esta interação com a aplicação será feita através de dois métodos: um para iniciar o processo e outro para o terminar (“*disableControls*”). Esta classe também será responsável por gerar, manipular e gerir o painel de informações através da deteção de alterações no ponteiro do rato: tanto através do seu movimento, como do clique.

Ligadas a esta classe estão também as três responsáveis por carregar as texturas, luzes e cenário, “*texture.factory*”, “*lightning.factory*”, “*room.factory*” respetivamente.

De forma a organizar este processo, a classe “*Recipe*” irá ser responsável por gerar e organizar as outras classes basilares recebendo, portanto, por parâmetro o mosaico e a solução. Os únicos

métodos públicos desta classe são o construtor e o "*create*" que retornará o objeto final que irá ser representado. Os restantes métodos desta classe serão apenas auxiliares.

As classes "*Box*" e "*Pallet*", que representam os produtos e a palete respetivamente, são caracterizadas pelas suas dimensões. No entanto, o produto tem ainda as informações respetivas da camada em que está inserido para se poder atualizar o painel de informações. Ambas têm também um método "*create*" que, à semelhança da classe "*Recipe*", retorna um objeto da biblioteca Three.JS que irá ser representado.

As ligações neste diagrama são de composição, porque o ciclo de vida de todas estas classes depende exclusivamente da classe principal, "Mosaic3DViewer". Esta sendo destruída, no fim do processo, tem de resultar obrigatoriamente na destruição de todas as outras. Neste diagrama vê-se também as relações de herança, porque todas estas classes irão depender de objetos e métodos disponibilizados pela biblioteca Three.JS.

4.2.4 Design do Componente de Manipulação 2D

Dada a natureza visual do componente de manipulação em duas dimensões, as decisões arquiteturais fluíram do processo de Design da interface. Tendo em conta as características modulares da tecnologia usada para o desenvolvimento deste componente, resolveu-se começar por primeiro decidir quais subcomponentes seriam necessários. Antes de se explorar as decisões arquiteturais vai começar-se por discutir qual o comportamento e disposição da interface do componente.

Maquete da Interface Gráfica

Nesta fase de preparação para o desenvolvimento foram apresentadas duas maquetes da interface para este módulo, como se poderá ver na Figura 10 e na Figura 11.

A maior diferença entre os dois exemplos é a posição dos painéis de navegação e as ferramentas que na primeira Figura se encontram à direita, e na segunda estão à esquerda, e também o menu da lateral esquerda da Figura 10 que que se encontra em baixo na Figura 11.

O objetivo que se tentou atingir com os dois exemplos foi agrupar os controlos, de forma a reduzir o trajeto que o utilizador tem de fazer com o cursor do rato, sem comprometer o tamanho da área de desenho. Foram tidas também em atenção as regras definidas por Ben Shneiderman (cf. 2.3) para reduzir o esforço da utilização desta interface, mantendo o utilizador no controlo.

Na primeira Figura, os três botões que serão usados com menos frequência, ficam encostados à esquerda. Estes representam as funções de gerar uma solução automaticamente, esconder o painel de ferramentas e de navegação, respetivamente.

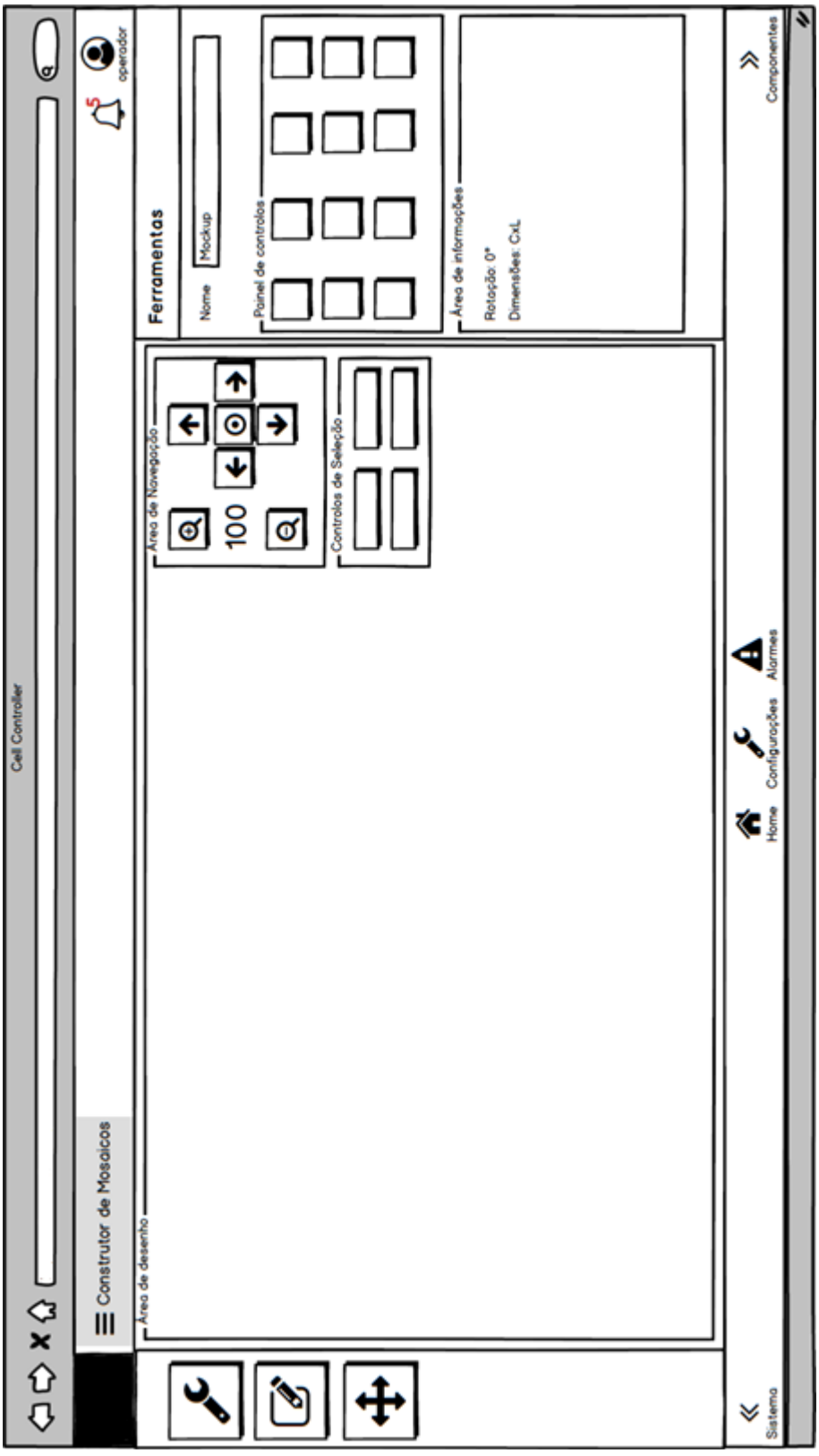

Figura 10 – Maquete com painel de navegação à direita

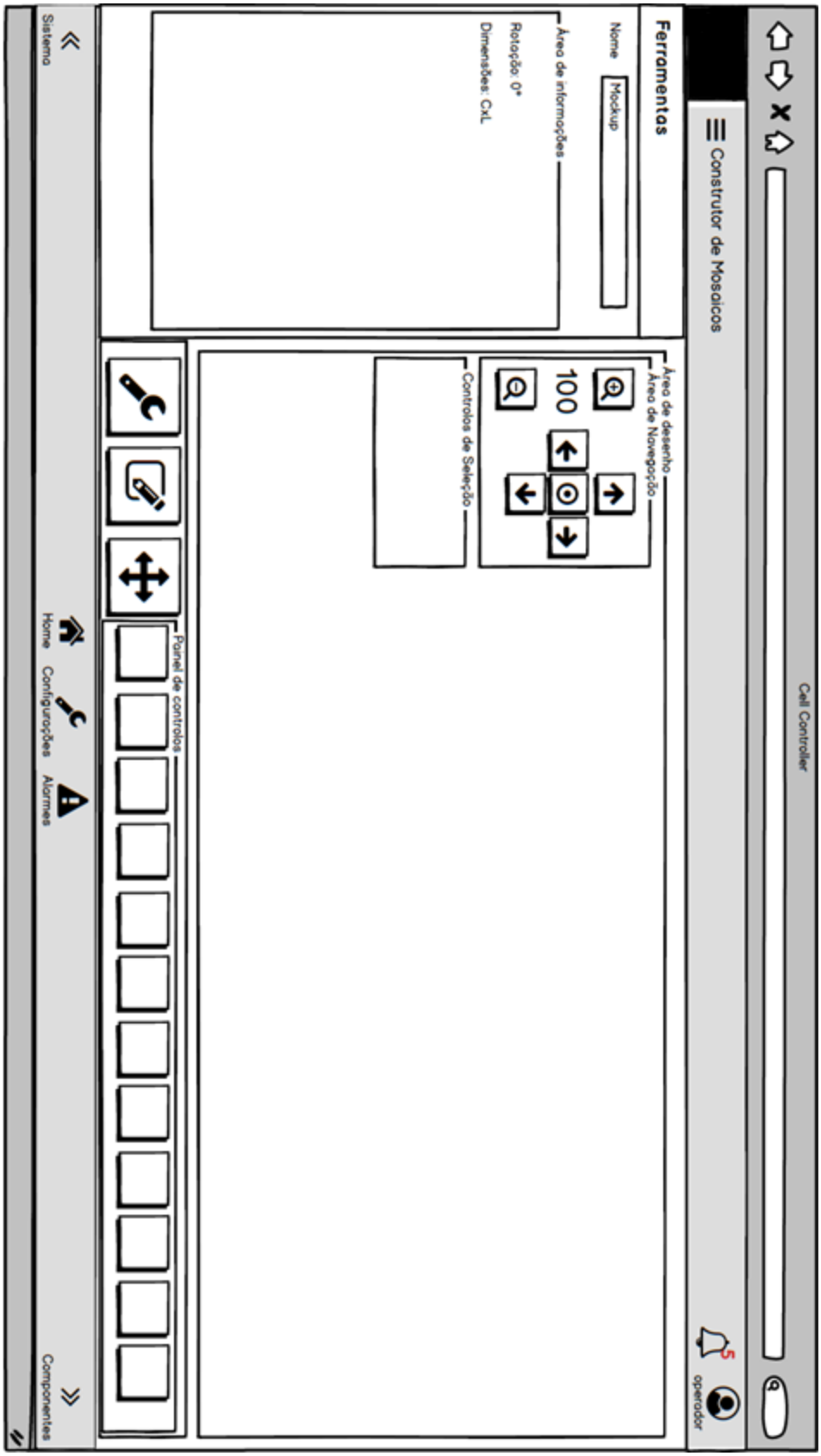

Figura 11 – Maquete com painel de navegação à esquerda

Na segunda Figura, estes três botões encontram-se em baixo, juntos aos controlos de manipulação, permitindo um maior aproveitamento do painel de informações.

Comparando os dois exemplos, as maiores diferenças são as dimensões na área de desenho e a disposição relativa dos painéis. Tanto a nível interno, como em conversas com o cliente, houve uma preferência pela altura do painel de desenho, em detrimento da largura, o que forçou o menu inferior do segundo exemplo a ficar orientado verticalmente e que os controlos passassem para o painel de informações.

Para além disso, a disposição relativa dos painéis foi uma questão onde a estética e a experiência foi mais importante do que a eficiência: para o cliente foi mais importante que a interface seguisse uma apresentação mais habitual, como a de outros softwares proprietários, onde os controlos estão encostados à direita. A experiência e a expectativa do utilizador superaram a eficiência que agrupar os controlos traria, sendo que na forma preferida do utilizador, este vai ter de movimentar mais o rato entre a cena, que estará maioritariamente do lado esquerdo do ecrã, e os controlos, que estarão do lado direito.

Nesta fase a comunicação com o cliente foi crucial para garantir que esta interface, quando desenvolvida, irá satisfazer as necessidades e as expectativas dos utilizadores.

Ícones das Transformações

Na Figura 12 pode ver-se desenhados os ícones que representam as funções pedidas pelo cliente. Estas estão agrupadas em três conjuntos lógicos, para facilitar a memorização das suas funções:

- A: As quatro funções permitirão alinhar as caixas selecionadas às margens da palete;
- B: Estas, permitirão ao utilizador alinhar as caixas usando os limites da própria seleção como referência;
- C: Estas são dois pares de transformações relativas aos eixos, as primeiras para alinhar e as segundas para distribuir, de forma equidistante, ao longo da palete.

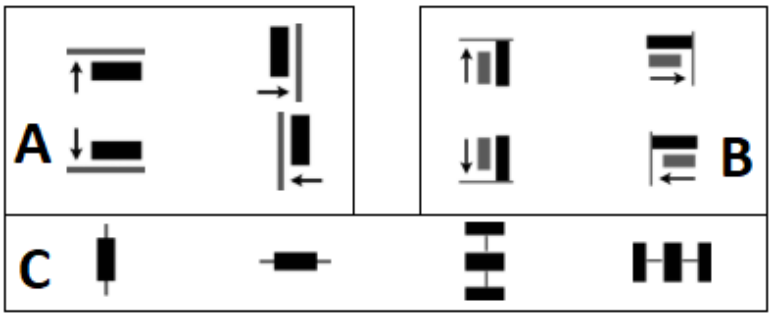

Figura 12 – Ícones das transformações

No Grupo A e no Grupo B foram postas setas para facilitar a interpretação dos ícones, tornando a direção e o sentido da translação claras. Para além disso, a ordem dos ícones foi relevante: do lado esquerdo dos Grupos A e B estão as transformações relativas ao eixo do Y, e do lado direito, as relativas ao eixo do X. Os pares de controlos do Grupo C também seguem esta ordem.

Diagrama de Classes

A interface gráfica, estando delineada, encontra-se pronta para passar ao estágio do Design Arquitetural do componente.

Cada uma das áreas definidas anteriormente na interface deverá ter o seu componente modular, que comunica com um componente central. Isto poderá ser visualizado na Figura 13:

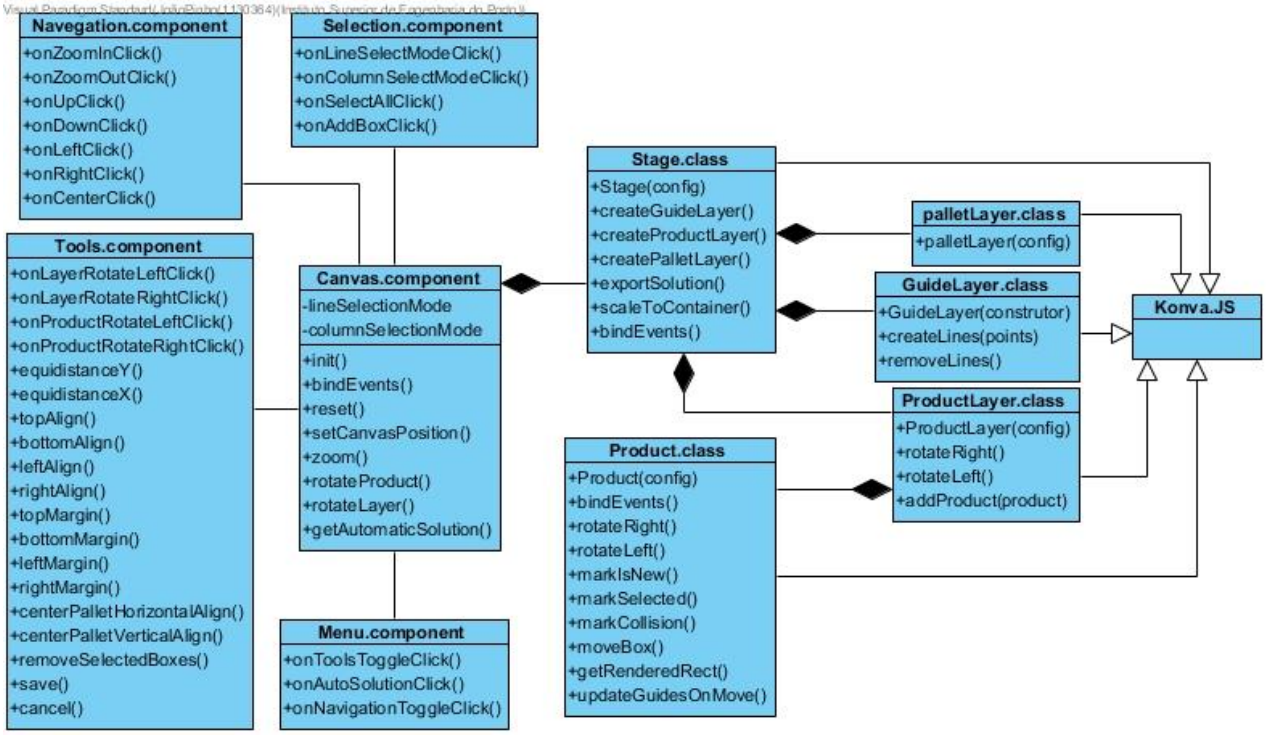

Figura 13 – Diagrama de classes do manipulador 2D

As entidades mais relevantes ao processo estão detalhadas neste diagrama. Cada área vista nas maquetes tem o seu componente respetivo, onde cada botão é mapeado ao seu próprio método. Uma vez que o "*Canvas.component*" é uma *"superclass"*, todos os outros componentes herdam os seus métodos desta. Para simplificar o diagrama, os métodos herdados foram agrupados na "*superclass*".

O "*Canvas.component*" servirá, portanto, de ligação entre os componentes e os objetos necessários para a representação gráfica. Estes consistem nos objetos homónimos que a biblioteca Konva.JS fornece, e por isso herdam os comportamentos adicionando outros pertinentes à presente aplicação. Aquando da inicialização deste componente com o mosaico

e a solução, se alguma tiver sido escolhida para edição, a classe "*stage*" é a primeira a ser instanciada. Esta é a estrutura principal a partir da qual todos os objetos que a compõem são desenhados na cena. É neste processo que as diferentes camadas são inicializadas: a da palete com os dados do mosaico, e a dos produtos com os da solução. A camada das guias também é inicializada, mas estará vazia até que o utilizador arraste uma caixa e esta seja detetada em proximidade com outra.

As relações de composição são necessárias para demonstrar que todas as instâncias das classes apresentadas terão de ser destruídas aquando do término do processo: os ciclos de vida de todas estas classes estão intimamente relacionados.

5 Implementação

Neste capítulo proceder-se-á à descrição da implementação dos dois módulos, expondo e detalhando a sua estrutura, componentes e métodos.

5.1 Visualizador de Soluções em 3D

De seguida, será descrito a implementação do componente de visualização das soluções em três dimensões, começando por algo que o diferencia do módulo de manipulação em duas dimensões: este componente é uma biblioteca externa ao projeto onde será usado. De seguida será apresentada a estrutura do componente, como também os aspetos mais relevantes como a própria modelação 3D e o painel de informações.

5.1.1 Metodologia e Instalação

Como foi dito na apresentação desta secção, este componente foi desenvolvido à parte para que o processo de inclusão num outro projeto seja mais simples.

Para encapsular o componente de visualização criou-se um projeto separado do principal para que este esteja disponível para instalação através do Bower. Esta é uma ferramenta cujo objetivo é o de agilizar a gestão de bibliotecas externas e as suas dependências[2]. Assim, a inclusão do módulo de visualização é realmente simples, podendo-se abstrair do seu

[2] Disponível em: hhtps://bower.io. Acesso em 2019

funcionamento interno, sendo apenas necessário desenvolver um *wrapper* que lide com o envio dos dados necessários através da API fornecida.

```
"mosaic3Dviewer": git@git.inesctec.pt:mosaic-3d-viewer.git#1.6.7
```

Código 1 – URL de Referência para o Componente de Visualização 3D

No Excerto de Código 1 pode-se ver o URL para o repositório de onde o componente é disponibilizado. No lado direito da imagem, o número "1.6.7" indica a versão que se pretende importar.

```
<mosaic3d-viewer mosaic="$ctrl.mosaic" solution="$ctrl.solution"
on-init="$ctrl.onInit(api)"> </mosaic3d-viewer>
```

Código 2 – Elemento que inclui o componente na página HTML

No Excerto de Código 2 pode-se observar o elemento que é responsável por passar os dados para o componente. Este elemento despoleta o método de inicialização representado no Excerto de Código 3.

```
function init() {
  var container = angular.element('div', $element);
  vm.width = container.width();
  vm.height = container.height();

  threeDViewerInstance = new ThreeDViewer({
    mosaic: vm.mosaic,
    solution: vm.solution,
    SCREEN_WIDTH: vm.width,
    SCREEN_HEIGHT: vm.height,
    container: container[0],
  });
}
```

Código 3 – Método de inicialização do componente de Visualização 3D

Este método simples cria uma instância da componente de visualização passando todos os dados pertinentes:

- "*mosaic*": Estrutura que contém as dimensões da palete e dos produtos;
- "*solution*": Estrutura que contém as coordenadas dos produtos, juntamente com o conjunto de propriedades que podem transformar cada camada (angulo de rotação, espelho e divisória entre camadas);
- "*SCREEN_WIDTH*" e "*SCREEN_HEIGHT*": Dimensões herdadas do contexto em que o elemento que contém o visualizador está contido;

- "*container*": Elemento base que vai conter o visualizador e painel de informações.

O projeto principal abstrai por completo o funcionamento interno do componente, tendo apenas de fornecer os dados iniciais e dimensões do elemento. Assim, seguindo o princípio de Alta Coesão e Baixo Acoplamento, as responsabilidades estão bem delineadas: o projeto principal tem apenas a responsabilidade de prover os dados necessários e o componente a de modelar a informação fornecida; na eventualidade do componente sofrer alguma eventual correção ou acréscimo de funcionalidades, o projeto principal está protegido.

5.1.2 Estrutura

De seguida, é apresentada a estrutura base dos ficheiros do componente sendo feita uma breve descrição de cada diretório:

- "dist": Este é o diretório que é importado para o projeto principal. Aqui estão o ficheiro "minificado" (ficheiro resultado da concatenação e ofuscação de todos os outros ficheiros necessários ao funcionamento do módulo);

- "classes": Aqui estão os construtores responsáveis por gerar as entidades principais da cena: palete e caixas;

- "utils": Neste diretório encontram-se as classes estáticas responsáveis por gerar a cena e as luzes e também de carregar as texturas;

- "resources": Aqui estão as texturas da palete, das caixas, do chão e paredes.

5.1.3 Modelação

De seguida, serão descritas as classes principais e o fluxo operacional do visualizador.

A *class* principal, normalmente denominada por "*main",* está contida no ficheiro 3DViewer.js. Aqui é gerada a entidade da receita que é responsável por gerar e retornar um objeto que será usado para representar a cena nas três dimensões.

Para gerar a cena em três dimensões será usada a biblioteca Three.JS (cf. 2.5.2 - Three.js). Esta ferramenta fornece vários objetos e métodos que simplificam este processo.

Produtos

Os produtos são representados pelos seguintes objetos fornecidos pela biblioteca Three.JS:

- BoxGeometry: Este objeto representa a geometria do produto e recebe por parâmetros as suas dimensões;

- MeshPhonMaterial: Aqui são atribuídas as várias componentes da textura do produto. Este processo é descrito em maior detalhe posteriormente nesta secção;

- Mesh: Este objeto agrega os dois anteriormente referidos, podendo receber atributos externos para caracterizar o produto em maior detalhe, facilitando o processo da criação da receita.

A informação passada pela aplicação, através da API fornecida pelo componente, é composta por coordenadas, tanto das caixas como da palete, e pelas propriedades de cada camada (que indicam a rotação aplicada e transformações de espelho).

Para visualizar as caixas de uma forma mais realista optou-se por usar texturas. Estas são compostas por 3 componentes distintas, que segundo o Modelo de Phong são a ambiente, a difusa e a especular. Estas três componentes estão representadas na Figura 14 (A, B e C, respetivamente).

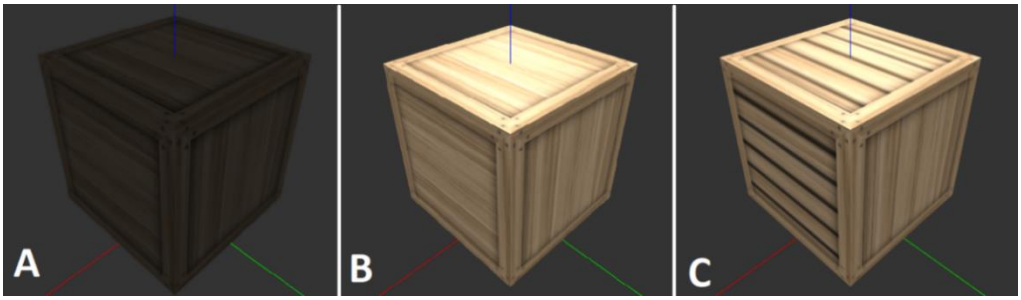

Figura 14 – Exemplificação de iluminações e texturas diferentes

Na secção A, a cena está apenas iluminada por uma fonte de luz ambiente: esta não tem direção nem produz sombra. Na secção B a cena já tem 4 focos, produzindo a sombra que se verifica nas faces laterais enquanto que na face superior há um gradiente de luminosidade que reduz gradualmente quando se aproxima das arestas. Na secção C a textura da caixa já tem a componente de relevo; isto produz sombra ao longo das ranhuras da caixa, resultando numa caixa mais verosímil.

Como foi referido anteriormente, a textura da caixa é composta por mais do que uma componente. Quando estas interagem com as fontes de iluminação da cena resultam numa caixa mais realista.

```
material = new THREE.MeshPhongMaterial({
    color: 0xffffff,
    map: ThreeDViewer.Textures.boxTexture,
    bumpMap: ThreeDViewer.Textures.boxBumpMap,
    normalMap: ThreeDViewer.Textures.boxNormalMap
});
```

Código 4 – Composição da textura da caixa

No Excerto de Código 4 pode-se ver o nome das três componentes, e na Figura 15 uma exemplificação:

- *Map*: é uma simples imagem, apenas indica a cor de cada pixel, a preto na Figura 15;
- *Bump Map*: indica a profundidade de cada pixel através de uma escala de cinzentos, a verde na Figura 15;
- *Normal Map*: este é um tipo de *Bump Map* que ajuda a criar detalhes como ranhuras indicando qual o ângulo da normal a considerar (representada a laranja na Figura 15).

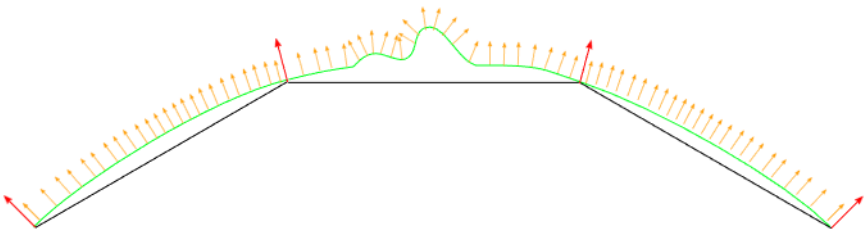

Figura 15 – Exemplo visual de um *Normal Map*[3]

Passando agora para o conjunto dos produtos com a palete, a classe Recipe será discutida de seguida. Esta é responsável por criar e organizar a estrutura que será desenhada no final do processo.

Este processo começa com uma preparação dos dados de entrada, reformulando a estrutura de dados recebida de forma a ser processada de uma forma mais simples e eficiente.

Um ponto importante que se teve de ter em consideração foi a questão das transformações: o cálculo delas são da responsabilidade deste componente. A situação é a seguinte: o utilizador pode manipular cada camada da solução e tem de conseguir vê-la antes de gravar e como o manipulador 2D não traduz as transformações aplicadas em coordenadas reais, é consideravelmente mais simples gravar as transformações à parte e enviá-las junto das coordenadas bases para o componente. Assim, o cálculo das coordenadas reais não terá de ser feito sempre que o utilizador interagir com a interface, garantindo responsividade da mesma.

Assim sendo, o processo terá de ter em conta não só as coordenadas, mas também possíveis transformações que deverão ser aplicadas.

Este processo foi implementado da seguinte forma:

- Primeiro extraiu-se as coordenadas de cada caixa, agrupando-as por camadas, ignorando toda a informação desnecessária que vem junto com o objeto da receita;

[3] Disponível em: https://docs.unity3d.com/uploads/Main/BumpMapBumpShadingDiagram.svg. Acesso em 2019

- Guardou-se as propriedades de cada camada;
- De seguida calculou-se os extremos da solução final;
- As dimensões de cada caixa são reduzidas para 90% do original, para que haja uma margem entre cada uma, facilitando assim a visualização de cada camada;
- Finalmente é calculado o centro da solução final, para que este possa ser alinhado com os eixos da cena.

Também teve de se lidar com uma diferença nas abordagens de caracterizar a posição de cada caixa: a do Departamento responsável por comunicar com os autómatos e a Three.JS. O primeiro usa dois pontos extremos da caixa não precisando das dimensões da mesma (pois estas estão implícitas nas coordenadas) enquanto que o Three.JS necessita de saber o centro do objeto para o desenhar.

Interlayers

Na Figura 16 está representado a verde um separador de camadas, ou *interlayer*. Como se poderá ver também nesta figura, estes *interlayers* estão no espaço vazio entre as caixas. Este foi um processo bastante simples devido à decisão de reduzir o tamanho das caixas: não foi necessário criar espaço extra para cada camada com uma *interlayer*.

Estes elementos opcionais são caracterizados de uma forma semelhante à dos produtos: apenas diferem na textura, porque em vez de terem uma, têm uma cor verde. As suas dimensões dependem das da palete e são ligeiramente maiores que esta para serem vistas mais facilmente.

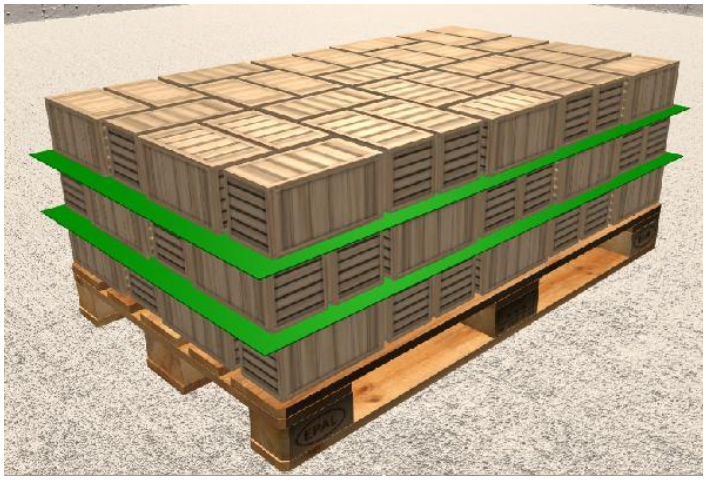

Figura 16 – Solução com *Interlayers*

Cena

A cena é composta por dois objetos que vão representar o chão e as paredes:

- Buffer Plane Geometry: este objeto é uma alternativa mais conservadora quanto à memória que ocupa e menos flexível em relação à alternativa “Plane Geometry”.

- Buffer Box Geometry: à semelhança do objeto descrito anteriormente, este também é uma alternativa mais leve à “Box Geometry” usada para os produtos.

Para a cena foram usados estes objetos mais leves para poupar memória onde ela não seria necessária, nem daria nenhum retorno palpável.

Para os elementos da cena também foram usadas texturas com as três componentes atrás referidas. Como se poderá verificar na Figura 17, a textura como demonstrada no painel B parece muito mais real do que a do painel A, porque a esta falta aplicar a componentes responsável pela diferença de altura e por consequência não tem a ilusão de profundidade.

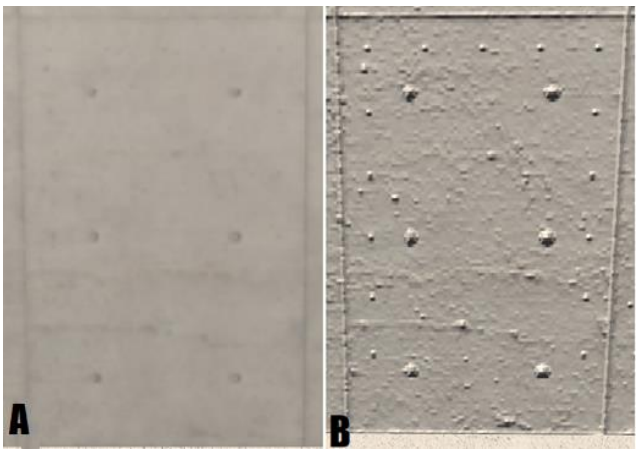

Figura 17 – Comparação da textura da parede com diferentes composições

Controlos

Com o objetivo de prover uma experiência mais completa ao utilizador, optou-se também por integrar controlos através do uso do rato.

O Three.JS fornece um tipo de controlos Orbitais muito completo. Como o próprio nome indica, este tipo de controlo permite ao utilizador mover a câmara em torno de um ponto, o que neste caso significa que ele será capaz de ver a solução final de qualquer direção.

Para garantir uma experiência mais imersiva, os controlos são restringidos de forma a que o utilizador não consiga ver dentro das caixas nem debaixo da cena.

```
controls = new THREE.OrbitControls(camera);
controls.maxDistance = 2;
controls.minDistance = 1.5;
controls.maxPolarAngle = (Math.PI / 2) * .95;
controls.enablePan = false;
```

Código 5 – Excerto de código para a criação dos controlos

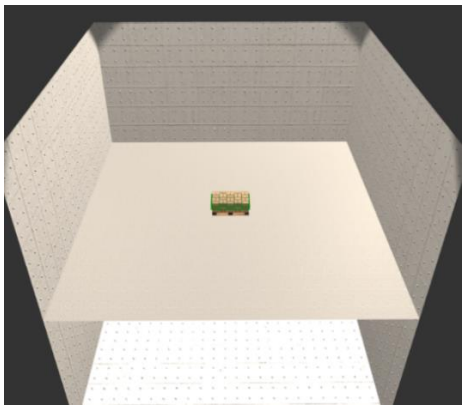

Figura 18 – Visão total da cena

Como se pode verificar no excerto de Código 5, as distâncias máxima e mínima (segunda e terceira linha) estão definidas para impedir que o utilizador veja a cena como na Figura 18, nem que aproxime a câmara a ponto de intercetar as caixas, como se pode ver na Figura 19, garantindo assim uma experiência mais controlada. Na quarta linha está a limitação relativa ao ângulo que a câmara faz com o plano XOY. Neste caso está limitado entre ligeiramente acima dos 0 graus e os 90.

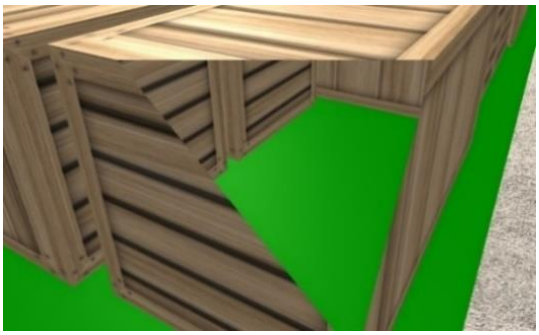

Figura 19 – Câmara a intercetar uma caixa

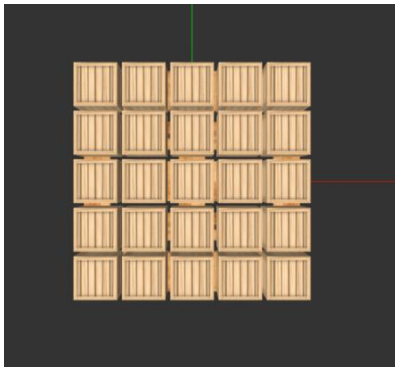

Figura 20 – Vista de Topo da Solução

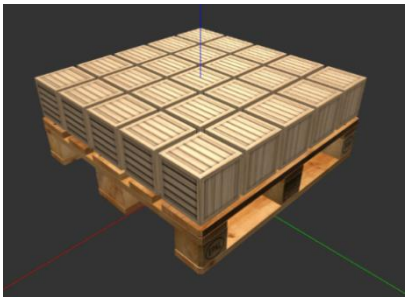

Figura 21 – Vista da solução

Na Figura 20 e Figura 21 vê-se uma camada de uma solução de cima centrada na origem. Desta forma os controlos de órbita são mais facilmente aplicados, evitando uma configuração desnecessariamente complicada. Os traços representam os eixos do X, Y e Z (a vermelho, a verde e a azul, respetivamente). Foi também necessário elevar todos os objetos acima da origem, para facilitar a criação da cena, pondo o chão a coincidir com o plano XOY. A cena está, portanto, dividida em duas metades como se poderá ver na Figura 18.

Processo de Desenho

Para o processo de desenho, o ThreeJS necessita de um método recursivo. Este pode ser visualizado na Código 6.

```
function animate(renderer) {
    renderId = requestAnimationFrame(function () {
        return animate(renderer);
    });
    controls.update();
    renderer.render(scene, camera);
}
```

Código 6 – Método de Desenho

Este método é responsável por redesenhar a cena utilizando a função “requestAnimationFrame” presente na segunda linha. Esta função de baixo nível faz um pedido diretamente ao *browser* que desenhe a próxima *frame* da animação. Tem também a vantagem de trabalhar à velocidade

de atualização do sistema (*refresh rate*) e garante que o desenho está sincronizado com o ciclo de desenho do browser (Stuart, 2017). Este ciclo por si só não desenha nada; para isto é necessário fazer a chamada ao motor de desenho presente na penúltima linha. Aqui sim, a cena é desenhada no elemento de HTML que a aplicação passa por parâmetro para o componente.

Esta função retorna um número inteiro que representa o número da *frame* atual (*renderID*), que é usado para cancelar o pedido da *frame* seguinte, sendo assim o método de paragem deste ciclo recursivo.

Luzes e Sombras

Para as luzes foram utilizadas duas fontes diferentes:

- Luzes de foco (*spotlight*): Esta fonte de luz tem um ponto de origem e uma direção. Cria basicamente um cone de luz do qual também se pode controlar a abertura. Pode também criar sombras;

- Luz de ambiente: ao contrário das luzes de foco, a de ambiente não tem um ponto de origem e ilumina a cena toda uniformemente de todas as direções, nunca gerando sombras.

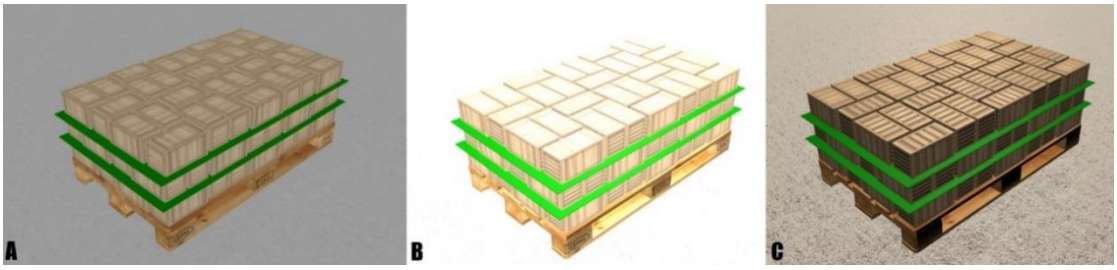

Figura 22 – Solução de exemplo com iluminações diferentes

Na Figura 22 pode-se ver uma solução de exemplo com três formas de iluminação diferentes, que foram rejeitadas: no painel A apenas se usou uma luz de ambiente, com um valor mediano, que deixou a solução muito escura e com os detalhes, como os do relevo, irreconhecíveis; no painel B subiu-se o valor da luz de ambiente, o que mais uma vez não foi aceitável pelo que a cena ficou demasiado clara; por fim, no painel C usaram-se quatro luzes de foco, o que já deu resultados mais aceitáveis mas, ainda assim, o contraste entre as faces laterais e as de cima é grande.

Por fim decidiu-se juntar a luz de ambiente do Painel A e as luzes de foco no Painel C: consegue-se ter o gradiente das luzes de foco e claridade nas faces menos expostas em simultâneo. O resultado do conjunto de luz escolhido pode ser se visto em todas as outras imagens neste capítulo.

Outra decisão importante foi a de não desenhar as sombras na cena, como se poderá ver um exemplo na Figura 23.

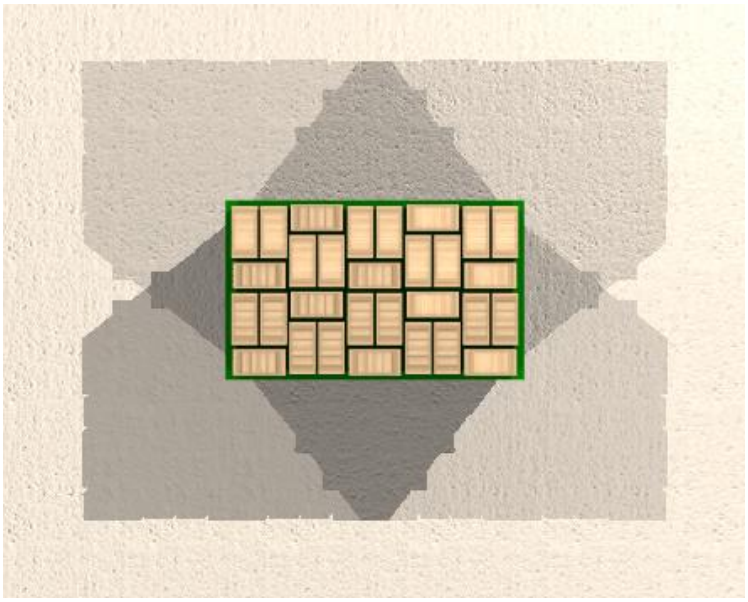

Figura 23 – Cena com Sombras

O cálculo das sombras, para tantos objetos como os desta figura, torna a animação muito pouco fluida reduzindo o número de *frames* por segundo. Entre uma experiência mais realista e uma mais fluida optou-se pela última, sendo esta a mais segura porque não será possível controlar as capacidades gráficas dos sistemas que integrarão este componente.

5.1.4 Painel de Informação e Seleção de Camadas

Para esta parte do visualizador foi necessário recorrer a uma técnica de *Raytracing* denominada de *Raycasting*. Esta técnica consiste no "disparo" de um raio a partir de um ponto que passa na origem e pelo ponto onde se encontra o cursor do rato, que neste caso é a vista do utilizador e detetar que objeto é intercetado. No contexto de *Raytracing* esta técnica é utilizada para computar imagens foto-realistas. Neste caso esta técnica foi utilizada para detetar para qual camada o utilizador está a apontar usando o rato.

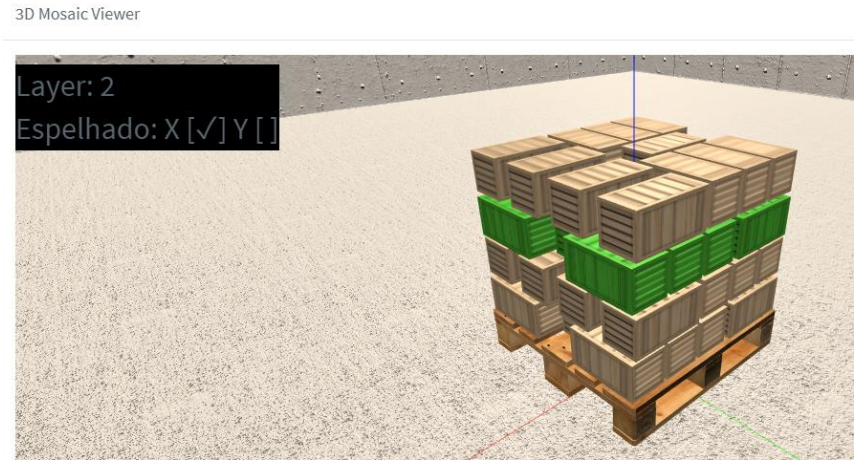

Figura 24 – Representação do painel de informação

Usando esta técnica é possível então detetar qual a caixa mais próxima da câmara, e por consequência saber qual a camada a selecionar. Na Figura 24 pode-se ver a terceira camada com um tom verde, para indicar que foi selecionada.

```
window.addEventListener("mousedown", onDocumentMouseDown, false);
window.addEventListener("mousemove", onDocumentMouseMove, false);
```

Código 7 – Excerto de Código para a criação de *eventListeners*

Para conseguir a interação desta funcionalidade com o HTML foi necessário agregar dois *eventListeners*, como visto no Excerto de Código 7: um para quando o utilizador carrega na camada e outro para o movimento do rato. Um *Event Listener* é um método que executa uma função sempre que uma condição é verificada.

Enquanto o componente estiver a ser utilizado, estes dois eventos, o de movimento e o de carregar no rato, serão usados para detetar e atualizar a informação da camada e caixa selecionada.

O método de *raycasting* permite ver as propriedades do objeto selecionado, o que por sua vez faz com que a atualização do painel um processo simples. Cada elemento do painel tem o seu id, ou seja, isto permite que se for detetado uma alteração no objeto selecionado basta alterar o texto do elemento correspondente.

Ao carregar, o utilizador poderá ver apenas a camada selecionada, como ilustra a Figura 25.

Figura 25 – Exemplo de uma camada selecionada

Neste modo de visualização o utilizador poderá ver mais informações sobre a camada, incluindo as cores respetivas de cada grupo de produtos, e o identificador do grupo da caixa que o utilizador selecionar ao mover o rato.

5.2 Manipulador de soluções em 2D

Por fim será descrito como o componente de manipulação das soluções em duas dimensões foi implementada. Será feita uma descrição semelhante ao componente anterior começando pela estrutura do componente e de seguida as características mais importantes do componente serão exploradas.

5.2.1 Estrutura

Segundo o que foi decidido na secção do Design (cf. 4.2.4) do componente, as várias entidades deste foram organizadas em diretórios distintos segundo as funções que cumprem, de forma a delimitar bem as suas responsabilidades. À exceção das classes, que descrevem todas os elementos necessários à representação e manipulação dos produtos, os vários diretórios são compostos por um componente, ou controlador, responsável pelas ações e comportamentos da área respetiva, um ficheiro de estilos e um ficheiro onde se encontra o HTML para a vista parcial. Será agora, então, sumarizada a estrutura do componente:

- */canvas*: Aqui está presente o componente que comunica diretamente com a vista, e que fornece todos os métodos que lidam diretamente com a tela de desenho (p. ex. controlos de navegação, seleção e manipulação);

- */classes*: Neste diretório estão declaradas todas as classes necessárias, onde os dados recebidos da base de dados são transformados em objetos do Konva.JS;

- /menu: Aqui estão agrupadas todas as ações que irão estar visíveis no menu lateral do editor;

- /navigation: Diretório que serve de intermediário entre as ações de navegação visíveis na tela e os comportamentos respetivos no componente do *canvas;*

- /tools: Diretório que contém o painel informativo da palete ou produtos selecionados e onde estão presentes os controlos de manipulação;

- /utils: Diretório que contém funções auxiliares de cálculo.

5.2.2 Representação

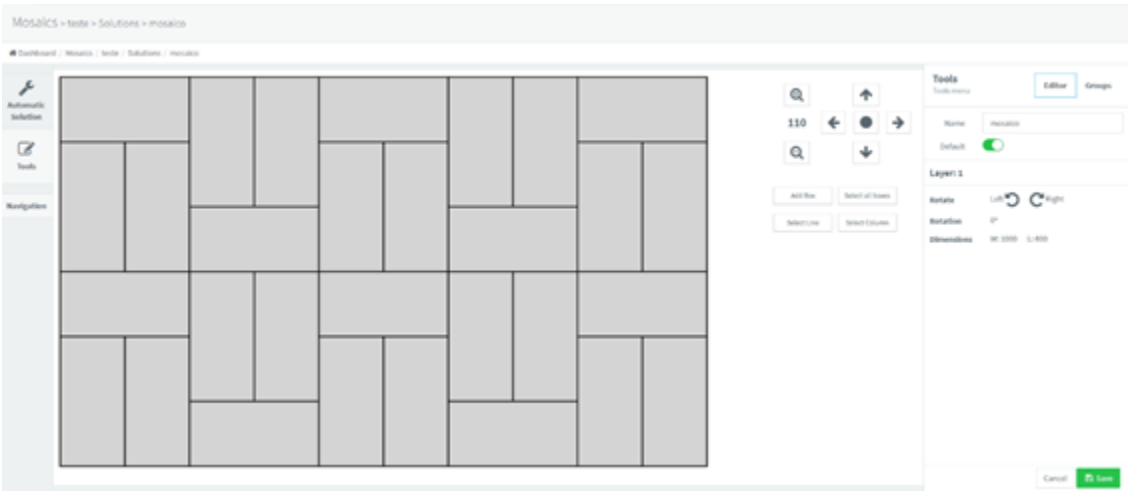

Figura 26 – Exemplo de mosaico no manipulador

Como foi referido anteriormente, cada uma das partes deste manipulador de mosaicos, como visto na Figura 26, está encapsulada em componentes diferentes. Na prática, o que isto permite fazer é construir uma página HTML que contém cada componente na forma de um elemento HTML.

```
<mosaic-editor mosaic="$ctrl.mosaic" solution="$ctrl.solution" on-
save="$ctrl.save(solution)" on-cancel="$ctrl.cancel()"></mosaic-editor>
```

Código 8 – Exemplo de elemento HTML que representa um componente

No excerto Código 8 o componente do manipulador de mosaicos é inicializado com dois parâmetros de dados e dois de "*callbacks*" (referências para funções): "*mosaic*" para as dimensões da palete e dos produtos, "*solution*" para as coordenadas, "*on-save*" que permite ao componente gravar a solução e o "*on-cancel*" que termina a edição sem gravar. Estas duas últimas funções estão implementadas pelo componente das Soluções que contém este, sendo apenas necessário passar as referências. Desta forma o padrão de Alta Coesão e Baixo

Acoplamento não é quebrado: o componente das Soluções que tem a responsabilidade de comunicar com o serviço e obter e enviar os resultados para a base de dados fornece ao subcomponente os métodos de gravar e de voltar à listagem de soluções.

De seguida, os vários componentes necessários ao funcionamento do manipulador serão descritos

Menu

Figura 27 – Secção do menu

Situada no lado esquerdo do painel, aqui encontram-se três botões:

- "*Automatic Solution*": faz o pedido ao serviço Solver que devolve um mosaico com base das dimensões atuais;
- "*Tools*" e "*Navigation*": mostram ou escondem, neste caso, o painel das ferramentas, e os botões de navegação.

Painéis de Navegação e Seleção

Figura 28 – Secção dos painéis de navegação e seleção

Em cima, na Figura 28, pode-se ver os controlos para alterar o zoom, com o valor atual entre os dois, as quatro setas direcionais, que movem a instância da classe "*stage*" na direção respetiva, e um botão central para repor à posição inicial.

A classe "*stage"* herda a classe do mesmo nome fornecida pela biblioteca Konva.JS, e contém três camadas:

- Camada da palete: Contém um retângulo castanho, como visto na Figura 29, que representa a palete e está em primeiro lugar para ser sobreposta pelas camadas superiores;

Figura 29 – Exemplo de uma palete no visualizador

- Camada das guias: Nesta camada surgem as guias, como vistas na Figura 30, que auxiliam o utilizador a encostar as caixas umas às outras. Este processo será falado em maior detalhe mais à frente neste documento;

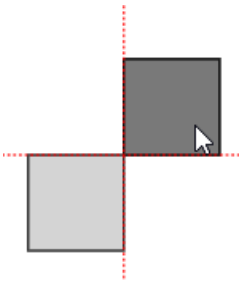

Figura 30 – Exemplo de guias de auxílio

- Camada dos produtos: Por fim, nesta camada estão os produtos visíveis ao utilizador.

Aplicando translações ou escalas, através do *zoom*, todas as entidades contidas nela sofrem as transformações proporcionalmente.

Avançando para as operações de seleção, que estão imediatamente abaixo das de navegação, podem ser vistas e apresentam um grupo de quatro botões:

- "Adicionar Caixa": este não representa uma ação de seleção. Apenas está aqui para ficar perto das outras operações de manipulação e de seleção;

- “Selecionar Tudo”: como o nome indica, este botão seleciona todas as caixas instanciadas;

- “Selecionar Linha” e “Selecionar Coluna”: estes dois botões ligam o modo de, como os nomes indicam, seleção de linhas (A) e colunas (B) respetivamente, como se pode ver na Figura 31.

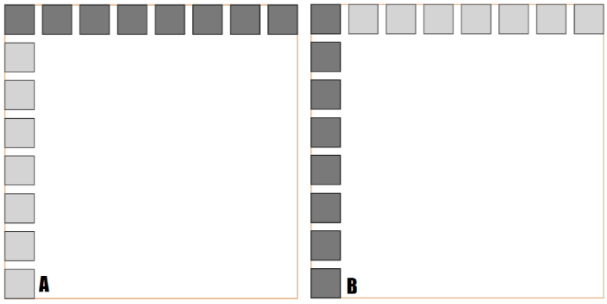

Figura 31 – Modos de seleção em linha e coluna

Estes dois últimos modos de seleção funcionam filtrando todas as caixas pelo valor da coordenada em X, no caso da linha, ou Y, no caso da coluna, selecionando apenas as que forem iguais à caixa em que o utilizador carregar.

Painel de informações e controlo

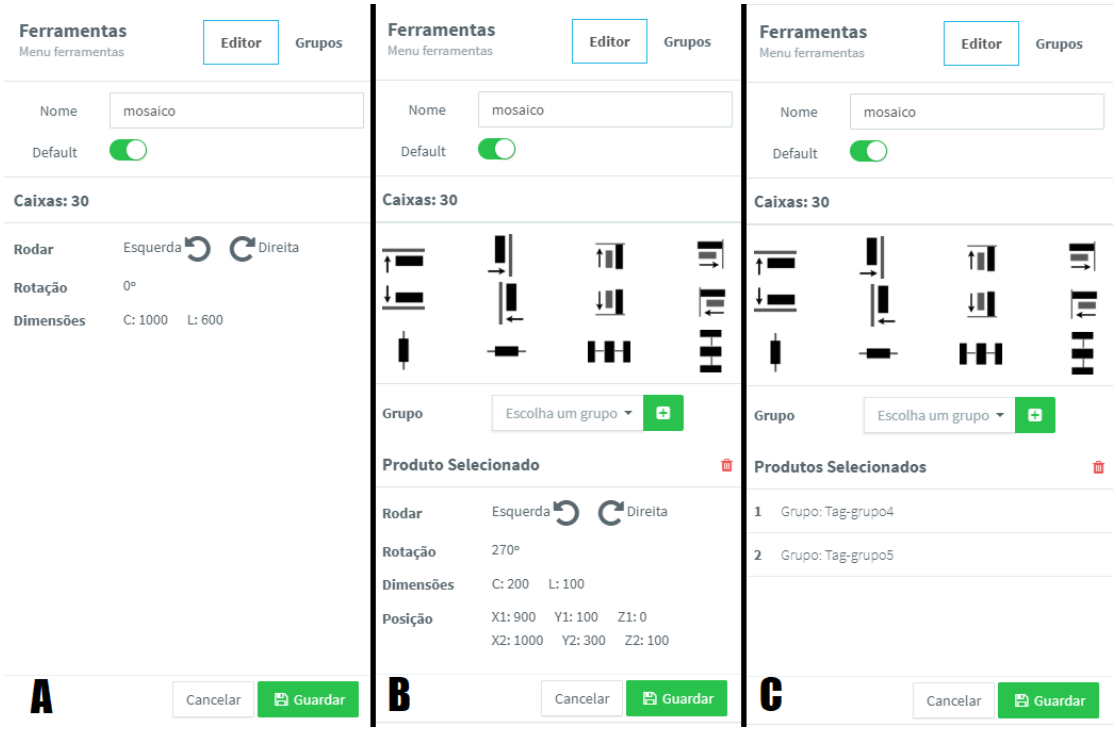

Figura 32 – Painel de informações e controlo

No painel apresentado na Figura 32, o utilizador tem acesso às informações tanto do mosaico, como da seleção atual. Se não estiver nada selecionado (A), a informação disposta é a do ângulo de rotação da camada e das dimensões da palete. Tem também dois botões para rodar todos as caixas em cima da palete: no caso da palete quadrada, esta rotação é feita em incrementos de 90º e no caso de ser retangular esta é feita em incrementos de 180º. Se o utilizador tiver selecionado uma única caixa (B), consegue ver a rotação, as dimensões e as coordenadas dos dois extremos. Tem também, dois botões para rodar a caixa. E por fim, se o utilizador tiver feito uma seleção de duas ou mais caixas, é mostrado no painel quais os grupos selecionados.

Tanto no painel B como no C, é dada a opção ao utilizador de alterar o grupo como também para aplicar transformações de translação às caixas selecionadas. Estas transformações, e respetivas ações, estão distribuídas conforme foi definido na subsecção 4.2.4.

As implementações destas transformações são todas semelhantes, à exceção da distribuição equidistante, pelo que explicando uma de cada, as outras serão esclarecidas. Como exemplo será demonstrada como o alinhamento das caixas ao extremo esquerdo da seleção. Para este efeito, todo o processo de seleção e manipulação será detalhado.

Assumindo que o utilizador já criou um certo número de caixas, e que as dispôs na tela, ele terá de selecionar quais deseja alinhar. Sempre que o utilizador carrega numa caixa, um evento, que contém as caixas afetadas, é lançado. Este evento é consumido pelo componente principal que atualiza o estado de cada uma destas caixas. Assim consegue-se saber quais as caixas selecionadas para poderem ser manipuladas.

Havendo sido feita a seleção, o utilizador pode agora escolher qual a transformação que deseja aplicar. Como foi referido anteriormente assume-se que esta é a do alinhamento ao extremo esquerdo da seleção. O excerto que corresponde a esta manipulação pode ser visto de seguida no Código 9.

```
vm.leftAlign = function () {
  var anchorX = 9999;
  angular.forEach(vm.selectedProducts, function (box) {
    if (box.getCoordinates().X1 < anchorX) {
      anchorX = box.getCoordinates().X1;
    }
  });

  angular.forEach(vm.selectedProducts, function (box) {
    box.moveBox({X: anchorX, Y: box.getCoordinates().Y1});
  });
};
```

Código 9 – Alinhamento ao extremo esquerdo da seleção

Como se pode verificar no excerto apresentado, primeiro é necessário encontrar qual o ponto de referência. As caixas são caracterizadas por dois pontos, como se pode verificar na Figura 33.

Neste caso, o objetivo da operação é alinhar todas as caixas segundo o valor mais baixo de "X1" da seleção feita. Este valor é encontrado no primeiro ciclo "*forEach*" em que o vetor é percorrido em busca do valor de "x1" mais baixo.

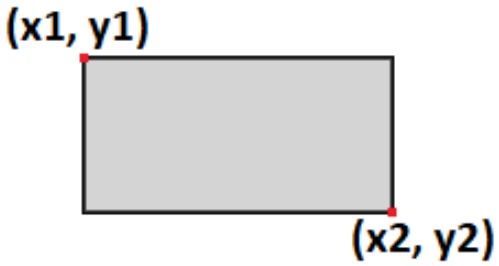

Figura 33 – Modelo de uma caixa e dos pontos que a definem

Este valor, denominado de âncora, tendo sido descoberto agora só resta aplicá-lo a todas as caixas selecionadas. Para isto será necessário percorrer mais uma vez o vetor das caixas selecionadas.

Todas as transformações seguem este fluxo geral, em que primeiro é necessário descobrir qual a âncora, seja uma margem ou um eixo da palete e de seguida esse valor é aplicado a todas as caixas selecionadas.

Olhando agora para o exemplo na Figura 34 temos a situação em que o utilizador tem cinco caixas alinhadas horizontalmente (A). Ao carregar no controlo de distribuição equidistante pelo eixo horizontal o algoritmo faz as seguintes operações: primeiro assume-se que há duas caixas encostadas às duas margens da palete (B); de seguida é calculado o espaço que as separa, representado pela seta no painel "B"; sabendo este valor pode-se então dividir o espaço em partes iguais (C) e assim mover cada uma das caixas restantes para os espaços respetivos (D).

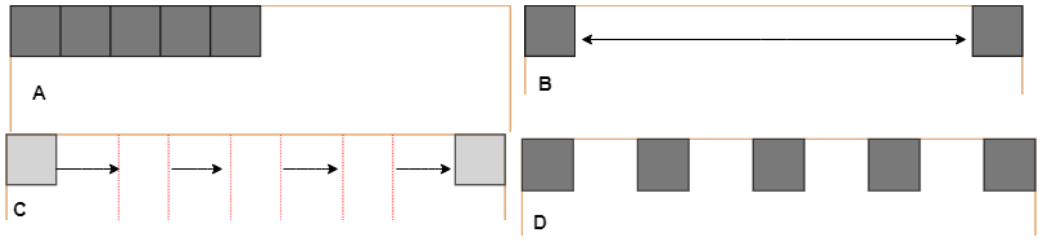

Figura 34 – Explicação do algoritmo de distribuição das caixas

Painel de desenho

Passando agora à secção principal, será detalhado o processo de criação da cena quando, por exemplo, o utilizador carrega um mosaico existente para o editar.

O primeiro estágio do processo é a inicialização do componente, que recebendo as dimensões do mosaico, as coordenadas dos produtos e as dimensões da tela de apresentação instancia o "*stage*" com estas informações criando também as camadas auxiliares das guias e da palete. De seguida é iniciado o processo de criação da solução em que as caixas são instanciadas e adicionadas à sua camada a fim de serem representadas. Antes deste do resultado deste processo ficar visível ele é ajustado às dimensões da janela e encostado à esquerda da janela, como se pode verificar na Figura 35 e Figura 36:

Figura 35 – Exemplo do ajuste da representação às dimensões da janela

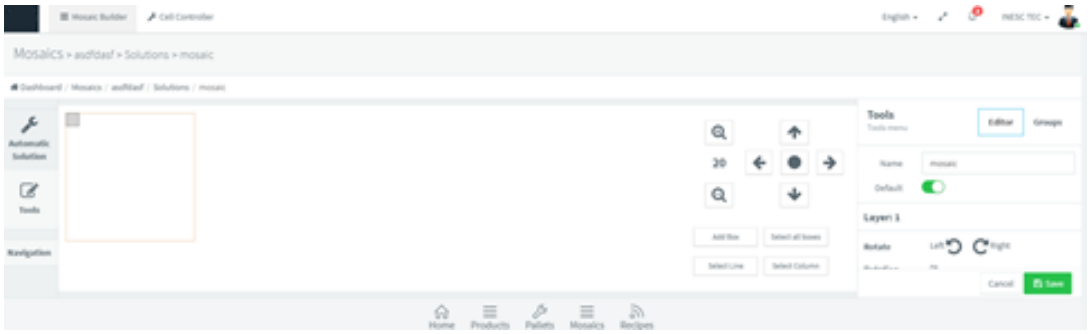

Figura 36 – Exemplo do ajuste da representação às dimensões da janela

Nos dois exemplos pode-se ver como a representação se ajusta ao ecrã do utilizador, permitindo sempre que este consiga ver a totalidade da palete.

Sistema de ajuste automático das caixas

Como foi referido de passagem, na enumeração das diferentes camadas contidas no objeto "*stage*". Este sistema tem o objetivo de auxiliar o utilizador a ajustar as caixas quando as quer encostar ou alinhar por um dos lados das caixas já existentes como se pode verificar na Figura 37.

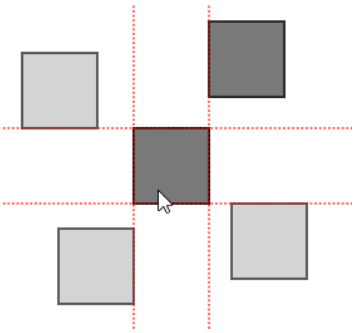

Figura 37 – Exemplo das linhas guias

As linhas guias são geradas sempre que o utilizador acaba de arrastar uma caixa e é gerada uma para cada valor distinto de X e de Y tal que todas as arestas estão contidas numa linha guia. Enquanto uma caixa é arrastada pelo utilizador um método é executado a cada instancia de redesenho da tela. Este verifica se pelo menos um dos vértices da caixa a ser movida se aproxima de pelo menos uma destas guias. Se um dado canto ultrapassa um limite arbitrário de pixéis, a caixa a ser movida é imediatamente ajustada à guia mais próxima. Se o utilizador mover a caixa tal que se ela não estivesse presa ela ficaria mais longe do que a margem estabelecida, a caixa deixa de estar presa e volta a estar no controlo do utilizador. Esta margem de ajuste é de 10 pixéis porque foi achada como a mais confortável e balançada. Segue-se uma breve explicação deste valor com o suporte da Figura 38, em que no painel A as caixas estão a 10 pixéis de distância, no B a 5 pixéis e no C a 20 pixéis:

- Se a margem fosse mais curta perderia o uso porque seria mais difícil para o utilizador ajustar a caixa em circunstâncias mais delicadas. Quando a margem tende para 0, como no painel B, mais inútil é porque não ajuda o utilizador a "prender" a caixa onde pretende;

- No caso oposto, se fosse mais comprida, em soluções com padrões de caixas mais densos, o espaço útil da palete seria reduzido tal que um movimento curto com a caixa resultaria num ajuste desproporcional.

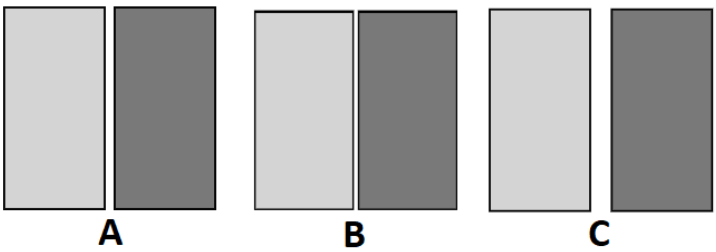

Figura 38 – Representação de diferentes margens de ajuste

O valor desta margem deverá ser ajustado consoante a experiência do utilizador, porque de certa forma está a limitá-lo. A diferença entre essa limitação causar ou não frustração é ténue pelo que será avaliada mais à frente.

6 Experiências e Avaliação

Neste capítulo será discutido o processo de avaliação do produto final, todos os métodos envolvidos, métricas utilizadas, os testes e respetivos resultados.

6.1 Metodologia de Avaliação

Segundo Marie (Baehr, 2004), a Metodologia de Avaliação é uma ferramenta cujo propósito é ajudar a entender os passos necessários para fazer uma avaliação de qualidade. Ao seguir este processo, é possível aprender o que é preciso para determinar o nível de qualidade de um produto.

Esta metodologia é composta pelos seguintes passos principais:

Definir os Parâmetros de Avaliação

Neste passo são traçados os parâmetros da avaliação de um ponto de vista geral, isto é, qual é a necessidade da avaliação; como os resultados serão utilizados; o que terá de ser relatado e como será apresentado às partes interessadas; e quais serão as diretrizes para implementar a avaliação. No presente caso de estudo, o mais importante a avaliar será se os componentes desenvolvidos apresentam uma interface intuitiva, se a usabilidade da interação com o utilizador é adequada e se corresponde aos requisitos definidos na fase da Análise de Requisitos (cf. 4.1). Para o primeiro, serão selecionadas pessoas para testarem ambos os componentes e para responderem a um questionário; para o segundo, será desenhado um Cenário de Qualidade, segundo o *Quality Evaluation Framework*. Ambos os métodos de avaliação serão descritos em maior detalhe mais à frente no documento.

Desenhar os Métodos Utilizados para a Avaliação

Com base nas diretrizes delineadas no passo anterior, deverão escolher-se critérios para a avaliação. Para cada critério deverá ser determinado que evidências serão necessárias e, se apropriado, que amostras serão utilizadas. Por fim será necessário determinar que rumo tomar de modo a coletar estas evidências.

Definir Padrões e Reunir Evidências

Antes de se iniciar a coleção de evidências, será necessário informar as partes interessadas das escalas que serão usadas para determinar a qualidade. Isto é feito para que esta desenvolva um processo de geração de decisões e defina os padrões que serão usados nesse processo, tudo baseado na qualidade do desempenho avaliado. Após estas tarefas terem sido completadas pode então passar-se à recolha de evidências e documentação das mesmas.

Relatar e Tomar Decisões

Neste último passo, a informação do primeiro terá de ser relatado às partes interessadas, para que estas comparem com os padrões definidos no terceiro passo. De seguida, poderão ser tomadas decisões segundo os resultados obtidos.

Como foi dito anteriormente, a avaliação do produto final passará por quatro fases distintas:

- Cenário de Teste: nesta fase será feita uma apresentação presencial dos componentes desenvolvidos, onde as funcionalidades principais serão testadas por um período de tempo. Após este, o participante irá realizar uma tarefa cujos dados da sua utilização serão gravados para uma análise posterior;

- Questionário: realizado após o teste, onde questões mais subjetivas da experiência da utilização da ferramenta serão analisadas;

- Cenário de Qualidade QEF: este servirá para avaliar o produto em si e se este cumpre os requisitos funcionais, e não funcionais, definidos no capítulo “Análise e Design”.

6.2 Cenário de Teste

Como foi falado na descrição da Metodologia da Avaliação, será necessário selecionar os participantes desta e da próxima fase da avaliação. Na escolha dos participantes foram tidas em consideração pessoas que tivessem experiência com interfaces gráficas num ambiente informático, ou que fossem potenciais utilizadores do tipo de aplicação apresentada neste documento. Estes foram os únicos requisitos impostos para garantir que, a própria utilização do computador, não fosse uma barreira que impactasse os resultados obtidos, e cujo *feedback* fosse relevante ao aprimoramento da aplicação desenvolvida.

Esta fase contou com a colaboração de 25 participantes, nas instalações do INESC TEC, e este grupo foi constituído maioritariamente por bolseiros e trabalhadores do Centro de Engenharia de Sistemas Empresariais, sendo quatro destes externos. Sobre este grupo verifica-se o seguinte: a idade média dos participantes ronda os 27 anos; destes, apenas quatro apresentavam algum conhecimento prévio relativo ao funcionamento dos componentes apresentados, e dois são operários fabris cujo *feedback* e perceção do sistema está mais próximo da sua aplicação final; em contraste a estes dois participantes, outros dois de uma faixa etária mais nova foram cruciais para uma análise mais superficial da interface de um ponto de vista da sua intuição e usabilidade. Todos estes têm experiência em interfaces equivalentes. As apresentações foram feitas num posto de trabalho, equipado com teclado e rato, no qual a aplicação estava em execução.

Segundo o estudo realizado por Virzi (Virzi, 1992), há uma relação entre o número de participantes numa sessão de avaliação e a percentagem de problemas de usabilidade detetados.

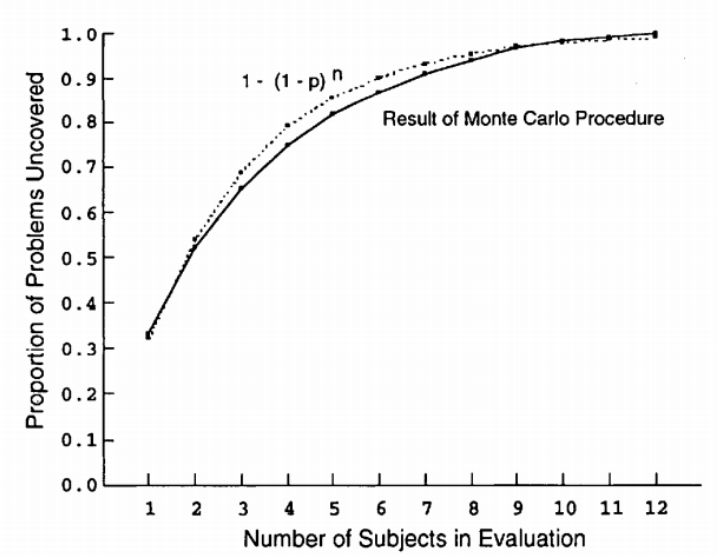

Figura 39 – Relação entre número de participantes e número de problemas de usabilidade detetados

Segundo a Figura 39, Virzi descobriu que com apenas três participantes a percentagem de problemas detetados seria de cerca de 65%; com nove, 95% dos problemas são detetados e com doze, esta percentagem já se encontra muito próxima dos 100%. Naturalmente, quantos mais participantes testarem a aplicação, a probabilidade de um novo erro ser detetado é sucessivamente mais baixa.

Neste cenário, cujo guião se encontra no Anexo A, decidiu-se então começar por contextualizar os componentes desenvolvidos, explicar as funcionalidades mais importantes, começando pelo componente de manipulação em duas dimensões, que exige uma maior atenção e detalhe, visto que é mais complexo, passando por fim para o componente de visualização em três dimensões.

Todas as sessões de avaliação foram feitas presencialmente no Centro, referido anteriormente, no INESC TEC, onde, como já foi descrito, se começou por descrever o âmbito do projeto e se explicou em detalhe as funcionalidades e controlos fornecidos pelo componente de manipulação em duas dimensões. Depois desta apresentação foram dados 10 minutos para os participantes testarem livremente a aplicação e, no fim, foi-lhes pedido que replicassem um exemplo de uma solução, o qual pode ser visto na Figura 40.

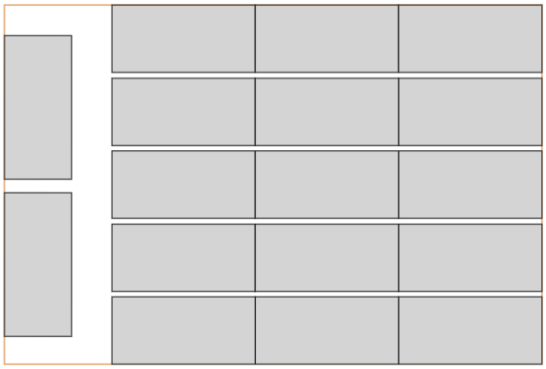

Figura 40 – Exemplo de uma solução a replicar pelos participantes

Este exemplo foi escolhido porque, para além de ser uma solução real, apresenta formas diferentes de o criar. O utilizador tanto pode arrastar cada caixa individualmente para o seu sítio respetivo, como pode usar os controlos desenvolvidos realizando a tarefa de uma forma muito mais rápida e eficiente.

A solução ideal, cujos passos intermédios podem ser vistos na Figura 41, poderia ser atingida se o utilizador seguisse os passos descritos de seguida:

A. Criar cinco caixas e selecioná-las;

B. Alinhá-las à margem direita da palete e distribuí-las verticalmente, usando os controlos respetivos;

C. Voltar a criar cinco caixas, selecioná-las, arrastando uma de modo a que fique encostada às caixas criadas anteriormente e alinhar as restantes à margem direita da seleção, mais uma vez usando os controlos respetivos;

D. Voltar a repetir o passo C para a terceira coluna e, por fim, criar as últimas duas caixas, rodando cada uma delas e arrastando para a margem esquerda.

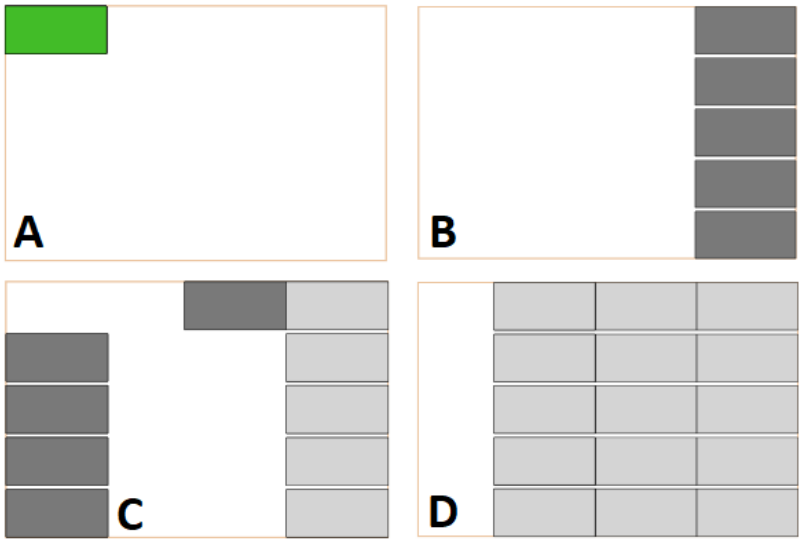

Figura 41 – Passos da solução ideal do cenário de teste

Para cada teste de eficiência foram registados os tempos de execução, o número de cliques nos controlos e o número de vezes que as caixas foram arrastadas usando o ponteiro do rato.

O teste do componente de visualização foi consideravelmente mais simples, tendo em conta o número reduzido de funcionalidades que este apresenta. Nesta fase foi dada ao utilizador a oportunidade de gerar algumas receitas para ver o comportamento do componente. Este processo de gestão da receita não está abrangido pelo presente documento. No entanto, decidiu-se acrescentar ao processo de teste para que o participante tivesse uma experiência mais próxima da real, e para que o teste pudesse ser mais completo.

Para este efeito, então, o teste do componente de visualização foi composto por uma introdução ao procedimento de criação de uma receita, e de um período de utilização livre que durou entre os cinco e a dez minutos. Dada a natureza menos complexa deste componente quando comparado ao anterior, não se preparou nenhuma tarefa para os participantes completarem.

6.2.1 *Feedback* dos Participantes

De um modo geral todos os participantes compreenderam bem o propósito da ferramenta apresentada e conseguiram alcançar a solução pretendida. Como seria de esperar, os testes sucessivos dos participantes resultaram em várias sugestões que nos foram comunicadas, tanto verbalmente como a partir da observação das suas participações.

Todos os participantes têm experiência nos mais variados ambientes informáticos e essa experiência prévia criou uma expectativa que, devido a algumas limitações da implementação, não foram atingidas. De seguida, serão apresentados os vários problemas de usabilidade descobertos nesta fase de testes.

Começando pelas caraterísticas que os utilizadores estavam na expectativa de encontrar no componente de manipulação:

- A seleção de múltiplas caixas: a grande maioria dos participantes esperava que seria necessário carregar na tecla "*ctrl*" para selecionar mais que uma caixa, ou que seria possível criar uma área de seleção usando o ponteiro do rato, à semelhança de muitos outros ambientes. Estas funcionalidades são, no entanto, cruciais e a sua implementação é imprescindível no produto final (cf. 7.2);

- A opção de retroceder: todos os participantes sentiram a falta desta característica tão comum. A falta desta funcionalidade gerou alguma frustração por parte de alguns participantes, porque quebrou o raciocínio durante a realização da tarefa;

- A opção de copiar e replicar uma seleção: dada a natureza repetitiva da tarefa proposta, uma grande percentagem dos participantes sentiu a falta da sequência "*copy/paste*" tão usada no ambiente informático. O facto de os participantes terem de repetir a criação de uma estrutura também foi fonte de alguma frustração;

- A capacidade de arrastar um conjunto de caixas: esta característica, muito relacionada com as anteriormente enumeradas, também foi sugerida por muitos participantes. No entanto, muitos foram capazes de usar os controlos disponíveis para contornar esta falta.

Para além destas características que foram contra as expectativas dos participantes, estes também encontraram um conjunto de erros de usabilidade:

- A organização dos controlos: apesar de estes estarem divididos em grupos lógicos, esta divisão não é clara e deveria ser explícita na forma de uma tabela;

- Os botões de rotação estão separados dos restantes controlos de manipulação e só funcionam quando apenas uma caixa está selecionada. Estes deviam conseguir rodar todas as caixas selecionadas;

- Devia ser possível alinhar uma seleção de caixas mantendo as posições relativas das mesmas, evitando que elas fiquem sobrepostas;

- Ao deslocar uma seleção, recorrer a um mecanismo de deteção de colisões que valide a operação, de modo a evitar sobreposições de caixas, e a um mecanismo de *snapping* que proceda ao correto alinhamento das mesmas;

- Retificar a sinalização das caixas sobrepostas: esta só atualiza quando uma caixa é arrastada manualmente. Por vezes, quando caixas estão sobrepostas e uma ação distribui as caixas, estas permanecem assinaladas a vermelho;

- Garantir a simetria da solução devia ser mais simples: na tarefa proposta, em que é necessário dispor duas caixas sem estarem alinhadas às margens opostas, garantir que estas estão à mesma distância das margens devia ser simplificado.

O *feedback* relativo ao componente de visualização foi muito positivo, onde foram dadas algumas sugestões valiosas de apresentação:

- Reduzir parcialmente a opacidade das camadas não selecionadas, em vez de as esconder completamente;

- Dada a quantidade reduzida de informação apresentada no painel de informações, transformar este num "*tooltip*" translúcido que se prende ao ponteiro do rato aquando da seleção de uma camada, tornando a interface mais limpa e concisa.

6.2.2 Discussão sobre os resultados

Dos 25 testes foram recolhidos os tempos de execução da tarefa, as vezes que os controlos foram usados e quantas vezes as caixas foram arrastadas, para estudar a forma como a interface foi utilizada.

Na Tabela 5, pode ver-se os resultados obtidos:

Tabela 5 – Resultados obtidos das 25 experiências

	Ideal	Média	Mínima	Máxima
Tempo	-	2m15s	1m16s	4m12s
Caixas Arrastada	4	23.3	5	50
Controlos Acionados	8	11.3	3	41

Os valores na Tabela 5 representam o tempo que cada participante demorou a replicar a solução apresentada na Figura 40, o número de vezes que arrastou cada caixa e o número de vezes que acionou cada controlo. Está também quantificada a combinação ideal de caixas arrastadas e ações acionadas. Assume-se que o ideal é arrastar o mínimo número de vezes as caixas, usando os controlos sempre que possível.

A intenção inicial dos controlos foi a de reduzir as vezes que os utilizadores teriam de arrastar manualmente as caixas e, por isso, segundo a solução ideal exposta na Figura 40, os participantes só teriam que arrastar as caixas um total de quatro vezes usando os controlos oito vezes. Alguns participantes conseguiram interiorizar rapidamente as limitações da interface e obtiveram resultados muito próximos do ideal, embora os tempos difiram muito. Os três

participantes, que mais se aproximaram da solução ideal, serão apresentados de seguida, na Tabela 6.

Tabela 6 – Participantes mais eficientes

Tempo	Caixas Arrastadas	Controlos Acionados
01m16s	6	9
01m37s	5	7
02m26s	7	9

O que distingue estes três participantes não foi tanto a forma como usaram os controlos, mas sim a velocidade com que moviam o rato e o tempo de que precisaram para pensar em cada ação. Isto significa que, com mais experiência de uso, o tempo será cada vez menos um fator significativo porque com a prática e a memorização dos controlos os utilizadores habitar-se-ão à interface. Em contraste a estes participantes, o participante menos eficiente apenas usou os controlos três vezes (uma vez para distribuir verticalmente as caixas e duas para rodar as caixas) recorrendo ao arrasto manual de todas as caixas.

Ao longo dos testes notou-se que o que distinguiu mais os participantes foi a forma como estes percecionaram os padrões da solução a replicar. Enquanto que uns recriaram a solução uma coluna de cada vez, outros fizeram-no uma linha de cada vez. Isto foi fulcral na maneira como interagiram com a interface porque, devido às falhas de usabilidade enumeradas anteriormente, não é possível alinhar caixas lado a lado sem se recorrer ao arrasto manual destas.

Uma observação que todos os participantes comunicaram foi a necessidade de recorrer ao mecanismo de ajuste automático, ou "*snapping*", de uma caixa, aquando do seu arrasto manual. Este mecanismo tornou-se fulcral devido à sua presença noutros ambientes informáticos. Esta realidade verificou-se porque, sem exceção, nenhum participante mostrou qualquer tipo de resistência ou dificuldade ao usar este mecanismo.

6.3 Inquérito de Satisfação

Como foi estipulado anteriormente, foram feitos inquéritos para avaliar a experiência subjetiva dos utilizadores da plataforma, tanto para o componente de manipulação em duas dimensões, como para a visualização de receitas em três dimensões.

Perante a definição formal de Usabilidade (cf. 2.1), irá ser apresentado um inquérito que avalie as interfaces construídas segundo os três indicadores apresentados: eficiência, eficácia e satisfação.

Este inquérito usará a escala de Likert. Esta escala tem várias categorias, das quais os elementos selecionados escolhem para indicar as suas opiniões, atitudes em relação a um dado tópico (Nemoto & Beglar, 2014).

As opções disponibilizadas serão enumeradas de seguida, na Tabela 7:

Tabela 7 – Opções do questionário

Avaliação	Opinião
1	Discordo completamente
2	Discordo parcialmente
3	Não concordo nem discordo
4	Concordo parcialmente
5	Concordo completamente

O questionário foi dividido em duas partes, uma para cada componente desenvolvido contendo perguntas relativas à experiência do utilizador. De seguida serão enumeradas as questões relativas ao componente de manipulação de soluções na Tabela 8:

Tabela 8 – Questões sobre o manipulador de soluções

Manipulador de Soluções	
1	Os ícones dos controlos são representativos da ação realizada.
2	Os controlos estão dispostos de uma forma confortável.
3	Os controlos são intuitivos.
4	As etiquetas dos controlos são úteis.
5	O sistema de ajuste automático é intuitivo.
6	Consegui realizar as tarefas sem interrupções inesperadas.
7	Os controlos disponíveis foram suficientes para atingir os objetivos.
8	Sou capaz de usar o sistema de forma autónoma.
9	Achei a interface esteticamente agradável.
10	Eu senti-me realizado ao utilizar esta ferramenta.

De seguida serão apresentados os resultados dos questionários onde serão discutidos:

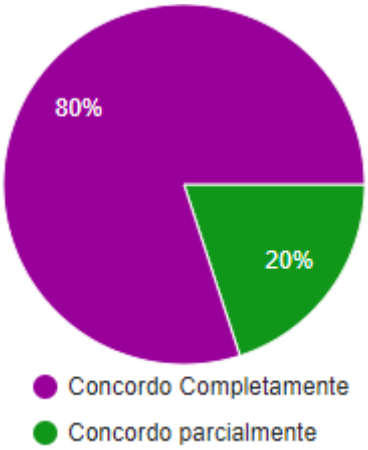

Figura 42 – Os ícones dos controlos são representativos da ação realizada.

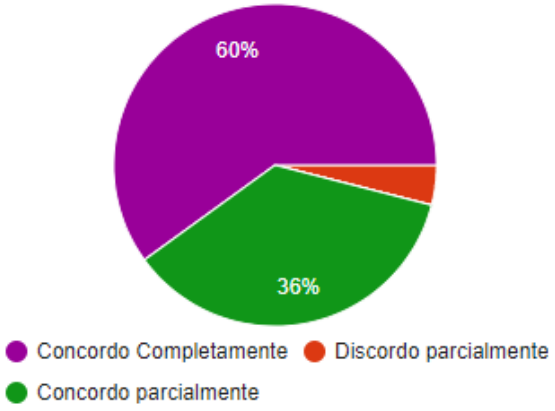

Figura 43 – Os controlos estão dispostos de uma forma confortável.

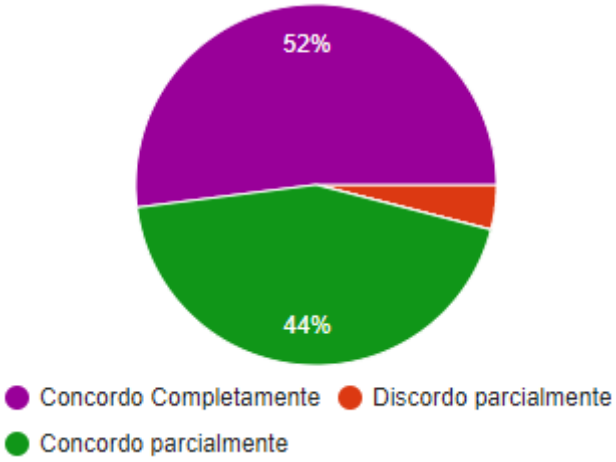

Figura 44 – Os controlos são intuitivos.

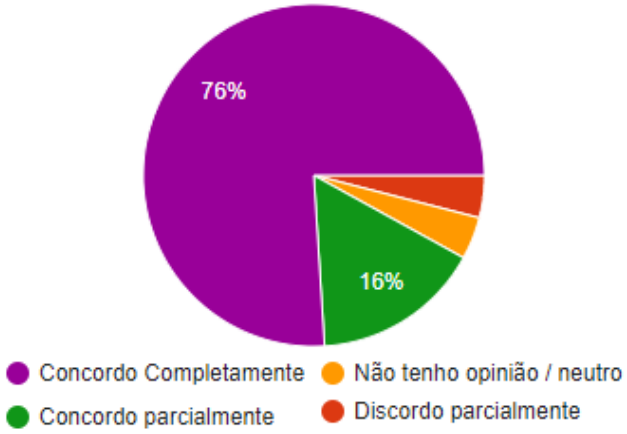

Figura 45 – As etiquetas dos controlos são úteis.

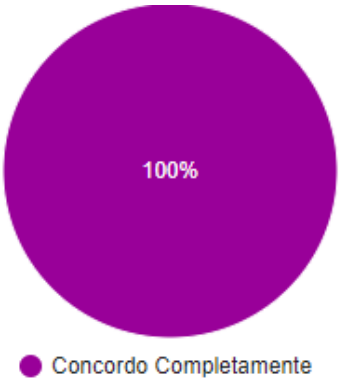

Figura 46 – O sistema de ajuste automático é intuitivo

O foco destas quatro questões foi o conforto e a experiência, de um ponto de vista mais prático, aquando da interação com os controlos durante o teste livre e a execução da tarefa descrita anteriormente. Destas quatro figuras pode ver-se que as respostas foram menos positivas nas questões referentes aos controlos, na Figura 43 e na Figura 44, devido principalmente às falhas de usabilidade já referidas. Um fator também referido pelos participantes foi a distância da palete, que está no lado esquerdo do ecrã, e os controlos, que estão no lado direito. Uma solução possível para este último aspeto será a implementação do arrasto de múltiplas caixas e da cópia de estruturas selecionadas. Isto terá dois efeitos: fará com que os controlos não sejam tão cruciais à criação e manipulação de soluções e reduzirá o trajeto do ponteiro do rato entre a área de desenho e o painel de controlos. Para além disso, a criação de atalhos fará com que utilizadores com mais experiência consigam usar a totalidade das funcionalidades implementadas de uma forma mais cómoda e eficiente, mais uma vez reduzindo os movimentos mais longos com o rato e reduzindo a necessidade da memória visual dos ícones dos controlos. Em relação a estes e às etiquetas respetivas, também podemos ver na Figura 42 e na Figura 45 que a opinião dos participantes foi bastante positiva. Isto pode ser facilmente explicado pela pouca experiência dos participantes face a esta nova interface: quanto maior for a experiência dos utilizadores, menos importantes e menos usados vão ser estes aspetos auxiliares.

Em relação à última questão, cujos resultados estão representados na Figura 46, pode ver-se como as observações sobre o mecanismo relatadas anteriormente se traduziram num resultado muito positivo: todos os participantes tinham experiência prévia, adquirida noutros ambientes informáticos, e por isso não sentiram qualquer atrito com este mecanismo.

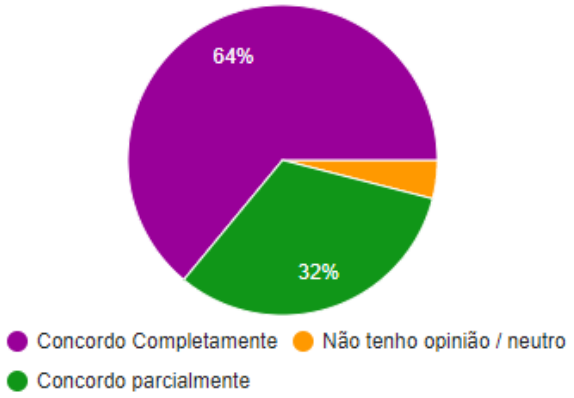

Figura 47 – Consegui realizar as tarefas sem interrupções inesperadas.

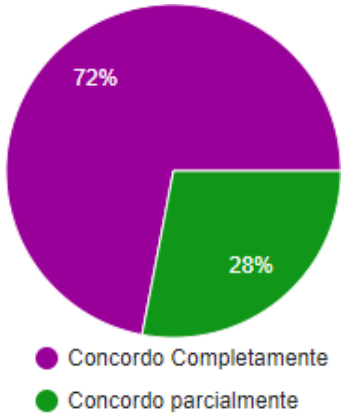

Figura 48 – Os controlos disponíveis foram suficientes para atingir os objetivos.

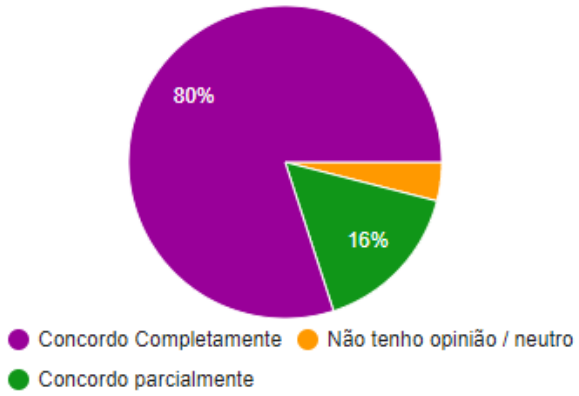

Figura 49 – Sou capaz de usar o sistema de forma autónoma.

Com estas três questões, o foco foi mais geral, olhando para a experiência de teste de uma forma mais holística: se as pessoas foram capazes de realizar a tarefa com as ferramentas fornecidas. Como foi referido anteriormente, a falta de algumas características basilares presentes noutros ambientes, como o "*copy/paste*" e, principalmente, como o retroceder, tornou os participantes mais propensos a cometer erros que acabaram por ser mais disruptores do que o expectável. Apesar dos transtornos causados pelos erros de usabilidade, os participantes conseguiram adaptar-se à interface e interiorizar as suas limitações.

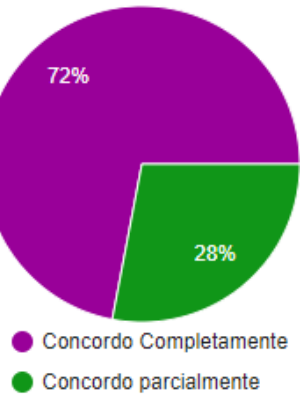

Figura 50 – Achei a interface esteticamente agradável.

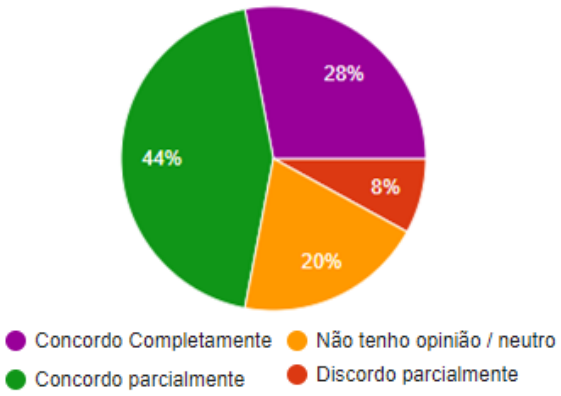

Figura 51 – Eu senti-me realizado ao utilizar esta ferramenta.

Finalmente, com estas duas questões foi possível avaliar uma faceta mais subjetiva e emocional da interação com a interface. A experiência dos participantes foi afetada negativamente devido aos erros de usabilidade enumerados anteriormente, como se pode verificar na Figura 51. O facto de os participantes não se sentirem realizados com a utilização da ferramenta, também se deverá ao facto de ela ser alheia às áreas de estudo de um grande número dos participantes.

Passando agora para o componente de visualização das receitas, em semelhança ao último componente, as perguntas do questionário serão apresentadas na Tabela 9:

Tabela 9 – Questões sobre o visualizador de Receitas

Visualizador de Receitas	
1	O controlo da rotação é intuitivo
2	É fácil aceder a cada camada individualmente
3	O Painel de informações mostra informações relevantes
4	A iluminação da cena é adequada
5	As texturas são adequadas ao contexto
6	Achei a interface esteticamente agradável

Dada a natureza informativa e simples deste componente, o questionário foi consideravelmente mais curto que o anterior. Os resultados serão apresentados de seguida, segundo uma estrutura semelhante à do componente anterior:

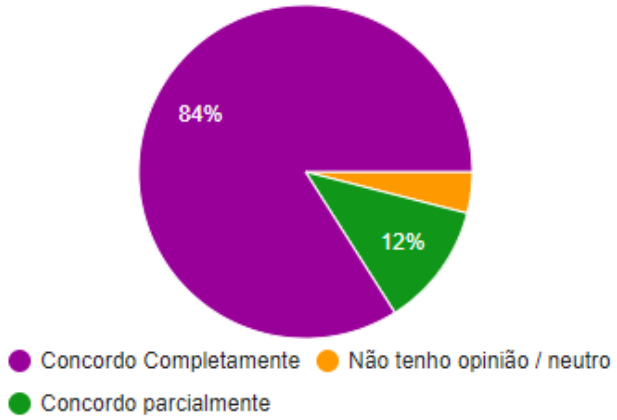

Figura 52 – O controlo da rotação é intuitivo

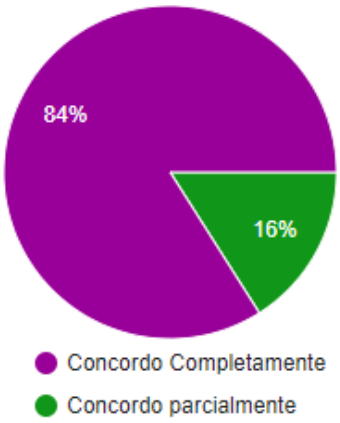

Figura 53 – É fácil aceder a cada camada individualmente

Com estas duas questões tratou-se de analisar a interação entre os participantes e o componente. A maioria dos participantes não comunicaram qualquer tipo de atrito, no entanto, alguns deles comunicaram alguma dificuldade a controlar a vista, devido à sensibilidade do visualizador.

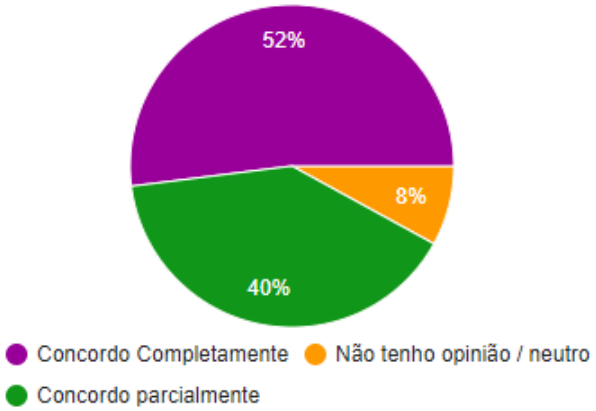

Figura 54 – O Painel de informações mostra informações relevantes

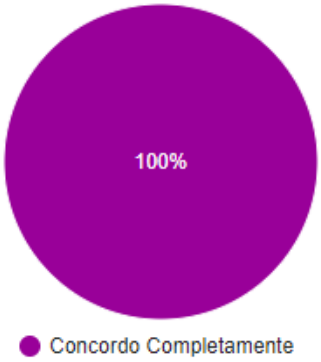

Figura 55 – A iluminação da cena é adequada

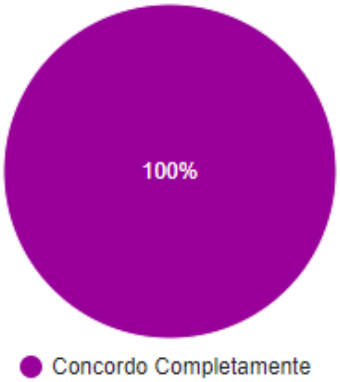

Figura 56 – As texturas são adequadas ao contexto

Com estas questões, a apresentação da informação, na Figura 54, e o realismo da cena foram analisadas, na Figura 55 e na Figura 56. Quanto à primeira, alguns participantes apresentaram

algumas sugestões quanto à apresentação, como foi referido anteriormente. No entanto, foram unânimes quanto à apresentação da cena: tanto as texturas como a iluminação eram adequadas.

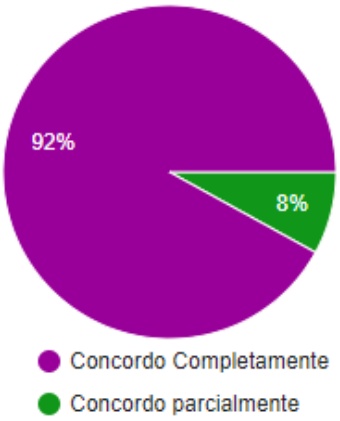

Figura 57 – Achei a interface esteticamente agradável

Finalmente, a parte estética do componente foi analisada com esta questão, cujos resultados podem ser vistos na Figura 57. A maioria dos participantes reagiu positivamente à interface, no entanto, alguns sugeriram a alteração da cor das "*interlayers*".

6.4 Cenário de Qualidade – QEF

Este sistema de avaliação permite quantificar três áreas, ou dimensões: funcionalidade, adaptabilidade e eficiência. Cada uma destas dimensões é caracterizada por fatores, e cada fator por um conjunto de requisitos. O objetivo do QEF é determinar o grau de desempenho do sistema em qualquer fase do desenvolvimento.

O estado do desenvolvimento do projeto é avaliado num sistema espacial de três dimensões. Estas três dimensões foram identificadas anteriormente, e cada uma delas representa um dos eixos deste espaço tridimensional.

Neste espaço, o estado do desenvolvimento é representado por um vetor cujas coordenadas são funcionalidade, adaptabilidade e eficiência. Dado que cada uma destas dimensões varia entre 0 e 1, o cenário ótimo é o vetor coincidir com o vetor (1,1,1). Estas três dimensões representam o seguinte:

- Funcionalidade: esta dimensão refere-se aos vários casos de uso, características e aspetos da interação de ambos os componentes. Esta dimensão é essencial pois vai ser responsável por medir até que ponto os requisitos funcionais e não funcionais foram implementados. A maior parte destes requisitos serão avaliados de uma forma direta,

conforme a implementação de cada requisito funcional, no entanto, alguns dependerão dos resultados dos questionários;

- Adaptabilidade: esta dimensão refere-se às questões de versatilidade e manutenção dos componentes: se os componentes se adaptam ao utilizador;

- Eficiência: esta dimensão refere-se às questões de navegação e qualidade do conteúdo dos componentes.

Cada dimensão será composta, então, por um conjunto de fatores e estes por requisitos, o que se poderá verificar no Anexo B - QEF.

Para cada requisito foi definido uma métrica de avaliação. Esta métrica é representada por uma percentagem de implementação entre 0% e 100%, em incrementos de 25%. No caso dos requisitos, cuja avaliação depende de uma questão do questionário, é feita uma soma dos pesos e o resultado será dividido pela pontuação máxima, segundo a seguinte fórmula.

$$\frac{\sum \text{Nº de Seleções da Resposta*Peso da Resposta}}{\text{Nº Total de Respostas*4}} * 100 \qquad (1)$$

Os pesos para esta soma são os seguintes:

- Discordo completamente: 0;

- Discordo parcialmente: 1;

- Sem opinião/ Neutro: 2;

- Concordo parcialmente: 3;

- Concordo completamente: 4.

Usando este padrão de análise de qualidade foi possível quantificar a qualidade dos componentes desenvolvidos em 98%. Por inerência, os requisitos em falha foram identificados e, por isso, será feita uma reflexão no próximo capítulo sobre o trabalho futuro.

7 Conclusões e Trabalho Futuro

Neste capítulo, serão apresentadas as conclusões sobre o trabalho que foi realizado, junto de uma discussão acerca dos objetivos atingidos, todas as limitações e trabalhos futuros que advêm destas.

Por fim, é feita uma apreciação final, tanto sobre o projeto como um todo, como sobre aprendizagem obtida a nível pessoal.

7.1 Conclusões

O objetivo principal do presente projeto foi apresentar uma solução para uma parte do problema da paletização, onde caixas de dimensões conhecidas são dispostas em múltiplas camadas, em cima de paletes, para fins de transporte. Este problema é complexo, tendo sido tratado de forma parcelar nesta tese. No entanto, propuseram-se soluções para duas componentes do referido problema: a manipulação de soluções de paletização, permitindo a reconfiguração de uma determinada camada de caixas, e a visualização do resultado obtido, ambas de uma forma natural para o utilizador.

O presente documento retrata todo o processo de desenvolvimento das interfaces para os dois componentes, referentes aos processos supracitados. Este desenvolvimento teve sempre em vista os objetivos definidos inicialmente, baseando-se no estudo do Estado da Arte relativo ao conceito de usabilidade, no desenvolvimento centrado no utilizador e sobre a experiência deste quando interage com sistemas computacionais, tanto a nível prático como a nível emocional.

O desenvolvimento deste projeto foi também, necessariamente, fundamentado nas fases de Análise de Requisitos e Design tanto da Arquitetura como da Interface. Foi nestas fases, especialmente na do Design da Interface do componente de manipulação em duas dimensões,

onde a interação e o *feedback* do cliente estiveram mais presentes, para garantir que o produto final estivesse o mais próximo de satisfazer as necessidades do mesmo. Esta fase também foi muito importante para garantir que o produto se destacava de sistemas de criação e visualização de soluções de paletização já existentes: é de suma importância tanto para o cliente, como para a organização responsável pelo desenvolvimento, garantir que o trabalho realizado tem realmente valor e que estará a suprir uma necessidade do mercado.

O desenvolvimento de ambos os componentes também foi detalhado para que os vários elementos que os compõem sejam bem compreendidos, tanto para justificar o valor destes, como para explicar o seu funcionamento. Para validar os componentes desenvolvidos foram feitas três avaliações diferentes:

- Em primeiro lugar, no cenário de teste foi pedido a vários participantes, após aprenderem a usar as ferramentas desenvolvidas, que replicassem um exemplo de uma solução de paletização, a fim de se estudar a sua interação com as interfaces;

- De seguida, no questionário, cada participante teve a oportunidade de avaliar e descrever a sua experiência após ter usado cada um dos componentes. Após isto, as suas opiniões e *feedback* foram recolhidos para estudar a usabilidade das interfaces desenvolvidas;

- Finalmente, foi feito um cenário de qualidade, segundo o padrão QEF e cujo objetivo foi o de comparar os resultados obtidos com os requisitos propostos inicialmente.

Nesta fase da avaliação recolheu-se o *feedback* dos participantes, o que promoveu a reflexão e discussão do trabalho futuro, como se poderá ler mais à frente.

Concluindo, verifica-se que os objetivos inicialmente propostos foram atingidos, embora haja ainda espaço para melhorias.

7.2 Trabalho futuro

Com base nos resultados obtidos nas três fases de avaliação consegue ter-se uma visão muito mais apurada do trabalho futuro. Este deverá passar pelo desenvolvimento de um conjunto de características de usabilidade. Inicialmente será discutido o trabalho futuro relativo ao componente de manipulação de soluções de paletização.

A primeira característica que carece de melhoramento é a da seleção de múltiplas caixas. Esta característica já se encontra presente noutros sistemas muito populares, como por exemplo nos sistemas operativos do MacOS da Apple, e do Microsoft Windows, pelo que será expectável para os utilizadores que a interface apresente comportamentos similares, nomeadamente:

- O uso do "*ctrl*" para a seleção múltipla de caixas: neste momento a interface encontra-se num modo de seleção múltipla, o que os participantes do cenário de teste acharam contraintuitivo. O ideal será o utilizador, por omissão, ser apenas capaz de selecionar uma caixa de cada vez, tendo que carregar na tecla "*ctrl*" para fazer uma seleção múltipla;

- A seleção em área: outro método de seleção muito popular cujos participantes acharam em falta. Este método de seleção torna toda a experiência da criação e manipulação de soluções de paletização mais fluida, fruto da redução drástica de cliques em soluções mais complexas. Através deste método um utilizador é capaz de, ao clicar na área de desenho e arrastar, criar uma área retangular em que todas as caixas nela contidas são selecionadas.

Outra característica de usabilidade muito importante é a capacidade de retroceder, também existente em inúmeros ambientes. Toda e qualquer interface gráfica deverá estar preparada para lidar com erros por parte do utilizador, e no caso desta interface, o mais simples será permitir ao utilizador voltar atrás no seu erro. Esta característica é fundamental pois tem a capacidade de reduzir os níveis de frustração do utilizador numa situação em que este comete um erro. Por isso o seu valor não está apenas no facto de ser prática, mas também pelo facto de impactar toda a experiência emocional do utilizador.

Também, a presença de atalhos para a criação rápida de caixas, nomeadamente do par "*copy/paste*": esta sequência tão usada em ambientes informáticos tem um grande impacto na usabilidade de um sistema, e isto verificou-se particularmente nesta interface porque a grande maioria dos participantes do cenário de teste estava na espectativa de conseguir usar estes atalhos.

De valor semelhante às características atrás referidas, a capacidade de arrastar livremente múltiplas caixas é também fundamental à usabilidade do componente. Para que tal seja implementado integralmente, o sistema de *snapping* terá de ser adaptado para ter em consideração todas as caixas selecionadas.

O *feedback* dos participantes não se resumiu a funcionalidades em falta, estes também descobriram falhas em algumas funcionalidades existentes:

- Aquando da manipulação, com recurso aos controlos, de um conjunto de caixas, estas deverão manter as suas posições relativas, evitando assim a sua sobreposição;

- Retificar a sinalização da sobreposição de caixas: por vezes esta não atualiza corretamente quanto caixas deixam de estar sobrepostas, fruto do uso de um controlo de manipulação;

- Facilitar o posicionamento simétrico de caixas: neste momento o sistema não fornece nenhum controlo para distribuição simétrica de caixas sem que pelo menos duas fiquem encostadas à margem. Pode ver-se um exemplo disso na Figura 58.

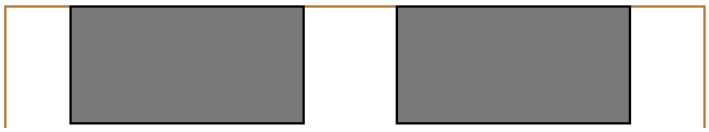

Figura 58 – Exemplo de duas caixas que poderão não estar distribuídas simetricamente

Para além das questões funcionais, também houve *feedback* sobre a organização e apresentação dos controlos: estes deverão estar separados por linhas para facilitar a sua localização; os controlos da rotação não deverão estar separados dos outros controlos de manipulação.

Em relação ao componente de visualização em três dimensões, o *feedback* foi sobre a apresentação das camadas e sobre o painel de informações:

- Aquando da seleção de uma camada as outras deverão estar translúcidas, e não completamente invisíveis;

- O painel de informações deverá ser adaptado à forma de um "*tooltip*" translúcido que aparece aquando da seleção de uma camada ou caixa.

- Fruto do método iterativo e incremental de desenvolvimento destes dois componentes, o processo de design, implementação e avaliação serão repetidos, com o objetivo de aprimorar o produto em termos de usabilidade tanto a nível funcional como ao nível de experiência.

7.3 Apreciação Final e Pessoal

Considero que ao longo do desenvolvimento do projeto como um todo, e não só apenas dos componentes apresentados, o meu conhecimento sobre desenvolvimento web, mais especificamente em desenvolvimento *frontend*, aumentou consideravelmente, particularmente no que toca ao conceito de usabilidade. Como por exemplo, a noção de que não basta que uma interface seja funcional e que a vertente estética e, sobretudo, ergonómica da criação de uma interface gráfica não seja tão superficial; que a estética de uma interface não se reserve ao que é belo ou atraente, mas que seja algo que pode ser usado para manipular a experiência do utilizador. De facto, a minha visão sobre a criação de interfaces gráficas evoluiu: no início do desenvolvimento era mais pragmática, a única prioridade era o lado funcional, pondo de parte quaisquer preocupações com experiências subjetivas porque estas não podem ser quantificadas. No entanto, aprendi rapidamente que a interface tem que se adaptar ao

utilizador, que esta pode preencher todos os requisitos funcionais e ainda assim ser inútil para o utilizador.

O trabalho realizado irá permitir que os operadores logísticos consigam executar o seu trabalho de forma mais rápida, simples e menos propensa a erros.

Em suma, considero que este projeto foi um contributo bastante positivo e edificador para mim e sinto que a minha capacidade de trabalho, tanto individual como em equipa, e também a de pesquisa melhorou bastante.

Referências

Abdou, G. & Yang, M., 1994. A systematic approach for the three-dimensional palletization problem. *International Journal of Production Research,* 10, 32(10), pp. 2381-2394.

Abras, C., Maloney-Krichmar, D. & Preece, J., 2004. *User-Centered Design,* s.l.: Sage Publications.

Alben, L. & Lauralee, 1996. Quality of experience: defining the criteria for effective interaction design. *interactions,* 1 5, 3(3), pp. 11-15.

Anton, 2018. *Overview | Konva - JavaScript 2d canvas library.* [Online] Available at: https://konvajs.github.io/docs/overview.html [Acedido em 16 Fevereiro 2019].

Baehr, M., 2004. *Faculty Development Series Evaluation Methodology,* s.l.: s.n.

Bjerknes, G., Ehn, P., Kyng, M. & Nygaard, K., 1987. *Computers and democracy: A Scandinavian challenge.* s.l.:Gower Pub Co.

Blanton, M. et al., 2009. Human-Computer Interaction. Em: *Encyclopedia of Database Systems.* Boston, MA: Springer US, pp. 1327-1331.

Cockton, G., 2002. From doing to being: bringing emotion into interaction. *Interacting with Computers,* 2, 14(2), pp. 89-92.

Craighead, J. D. et al., s.d. *Using the Unity Game Engine to Develop SARGE: A Case Study Wandering Pattern Detection for Person with Dementia View project Using the Unity Game Engine to Develop SARGE: A Case Study,* s.l.: s.n.

Crockford, D., 2008. *JavaScript: The Good Parts: The Good Parts.* s.l.:O'Reilly Media, Inc..

Danchilla, B., 2012. Three.js Framework. Em: *Beginning WebGL for HTML5.* Berkeley, CA: Apress, pp. 173-203.

Desmet, P., Overbeeke, K. & Tax, S., 2001. Designing Products with Added Emotional Value: Development and Appllcation of an Approach for Research through Design. *The Design Journal,* 28 3, 4(1), pp. 32-47.

Dewulf, K., 2013. Sustainable Product Innovation: The Importance of the Front- End Stage in the Innovation Process. Em: *Advances in Industrial Design Engineering.* s.l.:InTech.

Englebart, D., 1962. *Augmenting human intellect: a conceptual framework,* Stanford: s.n.

Forlizzi, J. & Battarbee, K., 2004. *Understanding experience in interactive systems.* New York, New York, USA, ACM Press, p. 261.

Gaver, B. & Martin, H., 2000. *Alternatives: exploring information appliances through conceptual design proposals.* New York, New York, USA, ACM Press, pp. 209-216.

Grudin, J., 1992. Utility and usability: research issues and development contexts. *Interacting with Computers,* 8, 4(2), pp. 209-217.

Hassenzahl, M., 2003. The Thing and I: Understanding the Relationship Between User and Product. Em: s.l.:s.n., pp. 31-42.

Hassenzahl, M., 2006. *Hedonic, Emotional, and Experiential Perspectives on Product Quality QUALITY OF INTERACTIVE PRODUCTS,* s.l.: s.n.

Hassenzahl, M. & Tractinsky, N., 2006. User experience - a research agenda. *Behaviour & Information Technology,* 3, 25(2), pp. 91-97.

ISO/TC 159/SC 4, 1998. *ISO 9241-11,* s.l.: s.n.

Katz, N., Cook, T. & Smart, R., 2011. Extending Web Browsers with a Unity 3D-Based Virtual Worlds Viewer. *IEEE Internet Computing,* 9, 15(5), pp. 15-21.

Kim, J. & Moon, J. Y., 1998. Designing towards emotional usability in customer interfaces—trustworthiness of cyber-banking system interfaces. *Interacting with Computers,* 3, 10(1), pp. 1-29.

Koen, P. A. et al., s.d. *FuzzyFrontEnd: Effective Methods, Tools, and Techniques LITERATURE REVIEW AND RATIONALE FOR DEVELOPING THE NCD MODEL,* s.l.: s.n.

Lahti, A., 2016. *Datan visualisointi HTML5-sovelluksessa,* s.l.: Kajaanin ammattikorkeakoulu.

Loewenstein & Lerner, J. S., 2003. *Handbook of Affective Science.* Oxford: Oxford University Press.

Logan, R. J., Augaitis, S. & Renk, T., 1994. Design of Simplified Television Remote Controls: A Case for Behavioral and Emotional Usability. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* 5 10, 38(5), pp. 365-369.

Maslow, A., 1954. *Motivation and personality.* First edition. ed. s.l.:s.n.

McDonagh, D., Hekkert, P., van Erp, J. & Gyi, D., 2004. *Design and emotion : the experience of everyday things.* s.l.:Taylor & Francis.

Merlo, A. et al., s.d. *3D MODEL VISUALIZATION ENHANCEMENTS IN REAL-TIME GAME ENGINES,* s.l.: s.n.

Miles, L. D., 2015. *Techniques of value analysis and engineering.* s.l.:Miles Value Foundation.

mrdoob, et al., 2019. *three.js example webgl_multiple_renderers.html.* [Online] Available at: https://github.com/mrdoob/three.js/blob/master/examples/webgl_multiple_renderers.html

Nemoto, T. & Beglar, D., 2014. *Developing Likert-Scale Questionnaires,* s.l.: s.n.

Nielson, J., 2012. *Usability 101: Introduction to Usability.* [Online] Available at: https://www.nngroup.com/articles/usability-101-introduction-to-usability [Acedido em 31 Dezembro 2018].

Norman, D., 2013. *The DESIGN of EVERYDAY THINGS.* s.l.:s.n.

Norman, D. A. & Draper, S. W., 1986. *User centered system design : new perspectives on human-computer interaction.* s.l.:L. Erlbaum Associates.

Parisi, T., 2012. *WebGL: up and running.* s.l.:O'Reilly Media, Inc..

Petridis, P., Dunwell, I., de Freitas, S. & Panzoli, D., 2010. *An Engine Selection Methodology for High Fidelity Serious Games.* s.l., IEEE, pp. 27-34.

Picard, R. W., 1997. *Affective Computing,* s.l.: s.n.

Picard, R. W. & Klein, J., 2002. *Computers that Recognise and Respond to User Emotion: Theoretical and Practical Implications,* s.l.: s.n.

Postrel, V. I., 2003. *The substance of style : how the rise of aesthetic value is remaking commerce, culture, and consciousness.* s.l.:HarperCollins.

Rüßmann, M. et al., 2015. *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries,* s.l.: s.n.

Shackel, B., 1959. Ergonomics for a Computer. *Design,* Volume 120, pp. 36-39.

Shneiderman, B., 2004. Designing for fun. *interactions,* 1 9, 11(5), p. 48.

Shneiderman, B. & Plaisant, C., 2004. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* 4th Edition ed. s.l.:Addison Wesley.

Silva, E., Oliveira, J. F. & Wäscher, G., 2016. INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH The pallet loading problem: a review of solution methods and computational experiments. *Intl. Trans. in Op. Res,* Volume 23, pp. 147-172.

Slant, 2019. *What are the best JavaScript drawing libraries?.* [Online] Available at: https://www.slant.co/topics/28/~best-javascript-drawing-libraries [Acedido em 15 Fevereiro 2019].

Stuart, G., 2017. *Introducing JavaScript Game Development.* Berkeley, CA: Apress.

Suh, E., Diener, E. & Fujita, F., 1996. *Events and Subjective Weil-Being: Only Recent Events Matter,* s.l.: s.n.

Tractinsky, N. & Zmiri, D., 2006. Exploring attributes of skins as potential antecedents of emotion in HCI. pp. 405-422.

Van Boven, L. & Gilovich, T., 2003. To Do or to Have? That Is the Question.

van Dam, A., 1997. Post-WIMP user interfaces. *Communications of the ACM,* 1 2, 40(2), pp. 63-67.

van der Hoog, W., Stappers, P. J. & Keller, I., 2004. Connecting mothers and sons. *interactions,* 1 9, 11(5), p. 68.

Virzi, R. A., 1992. Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough?. *Human Factors: The Journal of the Human Factors and Ergonomics Society,* 23 8, 34(4), pp. 457-468.

Voss, L., 2018. *The State of JavaScript Frameworks.* [Online] Available at: https://www.npmjs.com/npm/state-of-javascript-frameworks-2017-part-1 [Acedido em 15 Fevereiro 2019].

Wang, S. et al., 2010. *A new method of virtual reality based on Unity3D.* s.l., IEEE, pp. 1-5.

Anexo A – Guião do Teste

Desde já obrigado pela disponibilidade para testar a nossa interface.

O período de formação e posterior teste que se seguem servem exclusivamente para avaliar o desempenho da interface, e a forma como o utilizador interage com ela. Não se pretende avaliar as capacidades cognitivas dos participantes.

O tempo estimado para este teste será de cerca de 25 minutos:

- +/- 5 minutos para a apresentação das ferramentas;
- +/- 10 minutos para exploração e esclarecimento de dúvidas do componente de Manipulação;
- +/- 5 minutos para o teste cronometrado.
- +/- 5 minutos para a exploração e esclarecimento de dúvidas do componente de Visualização

Contexto

Este projeto está relacionado com um problema de Logística chamado de "Problema da Paletização". Este problema consiste basicamente no posicionamento e distribuição de caixas em cima de paletes. Esta distribuição tem que ser eficiente para maximizar o número de caixas por palete para reduzir os custos de transporte.

Componente de Manipulação

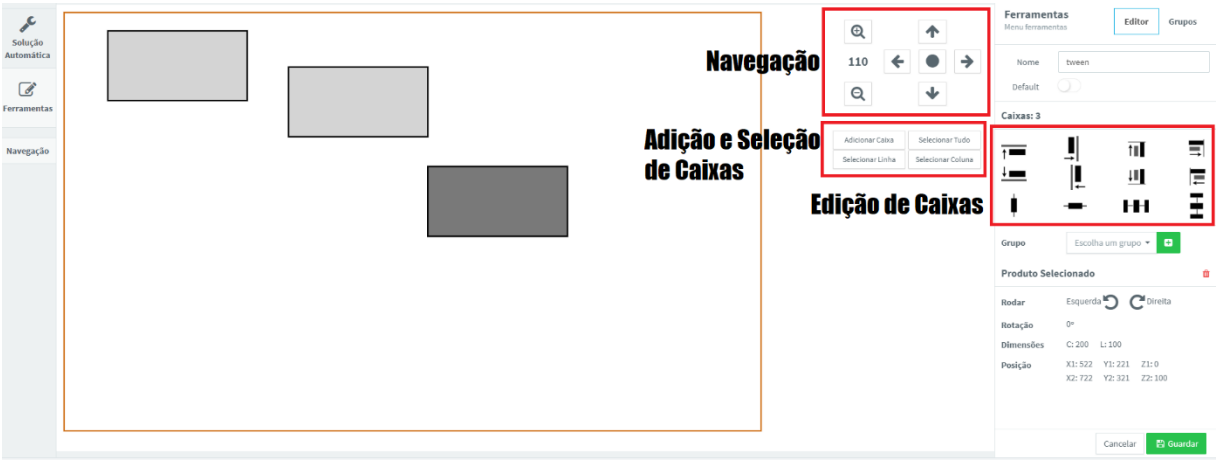

Figura 59 – Componente de Manipulação

Nesta Figura pode-se ver a área de desenho composta por:

A palete representada a castanho;

As caixas representadas a cinzento (mais escuro quando estão selecionadas);

O painel de Navegação onde a vista poderá ser controlada nas quatro direções, centrada e onde o zoom poderá ser controlado também.

O painel de Adição e Seleção de Caixas onde poderá acrescentar caixas novas, e onde se podem encontrar três formas de selecionar as caixas criadas:

Selecionar todas;

Selecionar uma linha ou uma coluna: ao carregar num destes botões, o utilizador poderá depois carregar numa caixa o que selecionará todas as caixas que estiverem alinhadas, à aresta superior ou à da esquerda respetivamente;

O painel de Edição das caixas

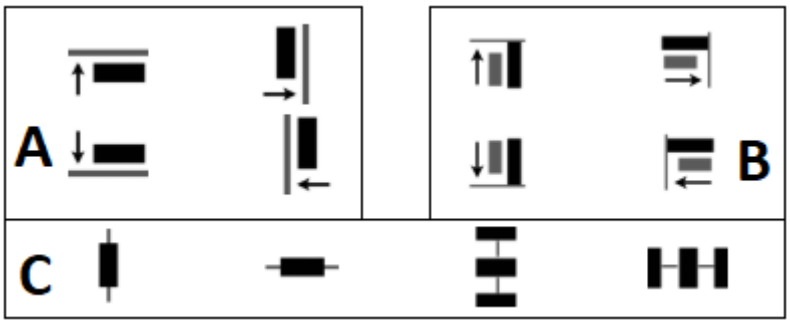

Figura A.60 – Ícones das transformações

Este teste terá que ser realizado sem ajuda para que os dados recolhidos sejam o mais verosímeis possível. Mais uma vez, apenas a interface está a ser avaliada, e não o utilizador.

Na terceira fase da avaliação, será pedido que o participante replique a seguinte solução.

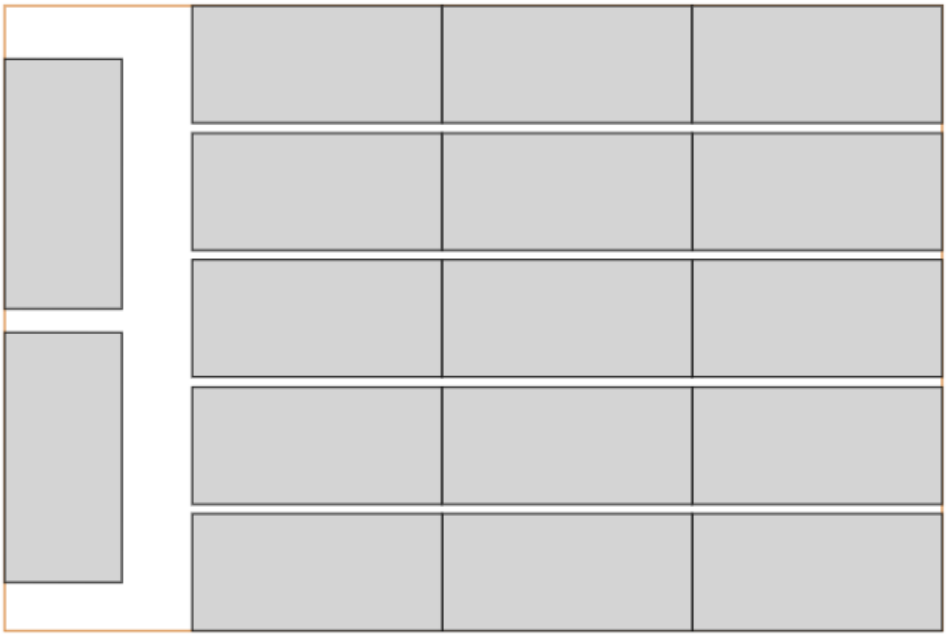

Figura A.61 – Exemplo a replicar

Anexo B - QEF

q	D	Qi	Dimensão	Qj	Wij (Factor Weight j in Dim i) [0,1]	Factor	rwjk (requirement weight k in Factor j) {2, 4, 6, 8, 10}	Requisitos	wfk % requirement fulfillment k) [0,100]
98%	0.09	91.18	Funcionalidade	80.00	0.294	Casos de uso - 2D	10	FCU-2D-1: O utilizador pode gerar soluções automaticamente.	100
							10	FCU-2D-2: O utilizador pode criar e modificar soluções existentes.	100
							10	FCU-2D-3: O utilizador pode adicionar caixas.	50
							10	FCU-2D-4. O utilizador pode apagar caixas.	100
							10	FCU-2D-5: O utilizador pode rodar caixas.	50
							10	FCU-2D-6: O utilizador pode rodar a solução.	100
							10	FCU-2D-7: O utilizador pode arrastar caixas.	50
							10	FCU-2D-8: O utilizador pode aplicar transformações a caixas.	50
							10	FCU-2D-9: O utilizador pode mover a vista.	100
							10	FCU-2D-10: O utilizador pode fazer zoom.	100
				100	0.118	Casos de uso-3D	10	FCU-3D-1: O utilizador pode visualizar a receita em três dimensões.	100
							10	FCU-3D-2: O utilizador pode mover a vista.	100
							10	FCU-3D-3: O utilizador pode fazer zoom.	100
							10	FCU-3D-4: O utilizador pode selecionar uma camada.	100
				87.5	0.235	Características - 2D	10	FC-2D-1: Apresenta Linhas guias	100
							10	FC-2D-2: Sistema de ajuste automático de caixas	100
							10	FC-2D-3: permite a seleção de múltiplas caixas em simultâneo	50

					10	FC-2D-4: Apresenta as informações de uma caixa quando selecionada	100
					10	FC-2D-5: Apresenta as informações sobre a palete	100
					10	FC-2D-6: Permite centrar a vista	100
					10	FC-2D-7: Apresenta um contador das caixas criadas	100
					10	FC-2D-8: Apresenta um indicador de colisões entre caixas	50
		100	0.088	Características - 3D	10	FC-3D-1: Apresenta as informações sobre a receita	100
					10	FC-3D-2: Apresenta as informações sobre a camada selecionada	100
					10	FC-3D-3: Apresenta as informações sobre os grupos que compõem	100
		100	0.176	Interação - 2D	10	FI-2D-1: Funções semelhantes apresentam controlos semelhantes	100
					10	FI-2D-2: Os ícones dos controlos representam a ação respetiva	100
					10	FI-2D-3: A interface é intuitiva	100
					10	FI-2D-4: O sistema de ajuste automático é intuitivo	100
					10	FI-2D-5: Os ícones têm etiquetas	100
					10	FI-2D-6: O funcionamento do componente é estável	100
		100	0.088	Interação - 3D	10	FI-3D-1: Os controlos são intuitivos	100
						FI-3D-2: É fácil aceder a cada camada individualmente	100
					10	FI-3D-3: A receita é apresentada de forma percetível	100

					10	FI-3D-4: O funcionamento do componente é estável	100
100	Adaptabilidade	100	0.80	Versatilidade	10	AV-1 - Os componentes estão preparados para o ambiente Web	100
					10	AV-2 - A aplicação contém outros idiomas	100
					10	AV-3 - Os componentes adaptam-se à resolução do ecrã	100
					10	AV-4 - A curva de aprendizagem está adaptada a novos utilizadores	100
		100	0.20	Manutenção	10	AM-1 - Os componentes estão preparados para adicionar mais funcionalidades	100
100	Eficiência	100	0.33	Navegação	10	EN-1: Os controlos estão dispostos de forma confortável	100
					10	EN-2: Os controlos foram suficientes para o funcionamento da aplicação	100
		100	0.67	Qualidade do Conteúdo	10	EQC-1: As etiquetas dos controlos são úteis	100
					10	EQC-2: Todas as mensagens são de fácil compreensão	100
					10	EQC-3- Ações significantes apresentam notificações	100
					10	EQC-4- As informações apresentadas são relevantes	100

Dimension	Funcionalidade
Factor	Casos de uso - 2D

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
FCU-2D-1: O utilizador pode gerar soluções automaticamente.	O componente deve conseguir receber soluções geradas automaticamente pelo Solver	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-2D-2: O utilizador pode criar e modificar soluções existentes.	O componente deve permitir ao utilizador criar soluções novas e editar soluções existentes	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-2D-3: O utilizador pode adicionar caixas.	O componente deverá permitir criar uma ou várias caixas simultaneamente	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-2D-4. O utilizador pode apagar caixas.	O componente deverá permitir apagar uma ou várias caixas simultaneamente	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-2D-5: O utilizador pode rodar caixas.	O componente deverá permitir rodar uma ou várias caixas simultaneamente	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-2D-6: O utilizador pode rodar a solução.	O componente deverá permitir rodar a solução	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-2D-7: O utilizador pode arrastar caixas.	O componente deverá permitir o arrasto de uma ou várias caixas	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado

FCU-2D-8: O utilizador pode aplicar transformações a caixas.	O componente deverá permitir aplicar transformações a várias caixas simultaneamente	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-2D-9: O utilizador pode mover a vista.	O componente deverá permitir mover a vista da solução	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-2D-10: O utilizador pode fazer zoom.	O componente deverá permitir mover a vista da solução	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado

Factor	Casos de uso- 3D

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
FCU-3D-1: O utilizador pode visualizar a receita em três dimensões.	O componente deverá permitir ao utilizador visualizar a receita em três dimensões	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-3D-2: O utilizador pode mover a vista.	O componente deverá permitir ao utilizador movimentar a câmara tal que ele consiga ver a receita em todas as direções	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FCU-3D-3: O utilizador pode fazer zoom.	O componente deverá permitir ao utilizador aproximar a câmara da receita	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado

FCU-3D-4: O utilizador pode selecionar uma camada.	O componente deverá permitir ao utilizador selecionar uma camada, escondendo as outras	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado

Factor	Características - 2D

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
FC-2D-1: Apresenta Linhas guias	O componente deverá apresentar linhas guias ao arrastar uma caixa em proximidade de uma outra	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FC-2D-2: Sistema de ajuste automático de caixas	O componente deverá ajustar uma caixa automaticamente quando esta é arrastada para perto de outra, ou da margem da palete	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FC-2D-3: permite a seleção de múltiplas caixas em simultâneo	O componente deverá ter pelo menos um mecanismo de seleção de várias caixas em simultâneo	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado

FC-2D-4: Apresenta as informações de uma caixa quando selecionada	O componente deverá apresentar toda a informação de uma caixa quando selecionada (dimensões e coordenadas)	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FC-2D-5: Apresenta as informações sobre a palete	O componente deverá apresentar as dimensões da palete quando nenhuma caixa estiver selecionada	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FC-2D-6: Permite centrar a vista	O componente deverá permitir centrar a vista à palete depois da vista ser movida, ou de zoom ser aplicado.	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FC-2D-7: Apresenta um contador das caixas criadas	O componente deverá apresentar um contador das caixas existentes na solução a ser criada ou editada.	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FC-2D-8: Apresenta um indicador de colisões entre caixas	O componente deverá indicar visualmente se duas ou mais caixas estiverem a intersecionar	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado

Factor	Características - 3D

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**

FC-3D-1: Apresenta as informações sobre a receita	O componente deverá apresentar as informações sobre a receita.	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FC-3D-2: Apresenta as informações sobre a camada selecionada	O componente deverá apresentar as informações sobre a camada quando uma for selecionada	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FC-3D-3: Apresenta as informações sobre os grupos que compõem	O componente deverá apresentar as informações sobre os grupos de cada caixa	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado

Factor	Interação - 2D

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
FI-2D-1: Funções semelhantes apresentam controlos semelhantes	Operações com resultados semelhantes deverão ser apresentados de formas semelhantes.	Não				Sim
FI-2D-2: Os ícones dos controlos representam a ação respetiva	Todos os controlos deverão ter um ícone que representa a ação que eles executam. A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários

FI-2D-3: A interface é intuitiva	A interface deverá ser intuitiva na forma como o utilizador interage com o sistema, segundo as regras apresentadas no estado da arte.	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários
FI-2D-4: O sistema de ajuste automático é intuitivo	O sistema de ajuste automático deverá ser intuitivo	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários
FI-2D-5: Os ícones têm etiquetas	Todos os ícones deverão ter uma etiqueta que explica sucintamente o que o controlo faz.	Não Implementado	-	Parcialmente Implementado	-	Totalmente Implementado
FI-2D-6: O funcionamento do componente é estável	O funcionamento do componente não é afetado por erros de execução. Quando estes acontecem são lidados de forma adequada	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários

Factor	Interação - 3D

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
FI-3D-1: Os controlos são intuitivos	A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários

FI-3D-2: É fácil aceder a cada camada individualmente	A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários
FI-3D-3: A receita é apresentada de forma percetível	A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários
FI-3D-4: O funcionamento do componente é estável	O funcionamento do componente não é afetado por erros de execução. Quando estes acontecem são lidados de forma adequada	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários
Dimension	Adaptabilidade					
Factor	Versatilidade					

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
AV-1 - Os componentes estão preparados para o ambiente Web	A aplicação é capaz de ser utilizada em ambiente web	Não	-	-	-	Sim
AV-2 - A aplicação contém outros idiomas	A aplicação é capaz de apresentar a interface pelo menos um idioma além do português	Não	-	-	-	Sim

AV-3 - Os componentes adaptam-se à resolução do ecrã	A interface dos componentes não depende da resolução do ecrã e conseguem-se adaptar.	Não	-	-	-	Sim
AV-4 - A curva de aprendizagem está adaptada a novos utilizadores	Os utilizadores são capazes de utilizar a interface autonomamente.	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários

Factor	Manutenção

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
AM-1 - Os componentes estão preparados para adicionar mais funcionalidades	-	Não	-	-	-	Sim

Dimension	Efficiência
Factor	Navegação

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**

EN-1: Os controlos estão dispostos de forma confortável	A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários
EN-2: Os controlos foram suficientes para o funcionamento da aplicação	A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários

Factor	Qualidade do Conteúdo

		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
EQC-1: As etiquetas dos controlos são úteis	A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários
EQC-2: Todas as mensagens são de fácil compreensão	A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários
EQC-3- Ações significantes apresentam noticações		Não	-	-	-	Sim

EQC-4- As informações apresentadas são relevantes	A ser determinado pelo resultado dos questionários	0-20% pontuação nos questionários	20-40% pontuação nos questionários	40-60% pontuação nos questionários	60-80% pontuação nos questionários	80-100% pontuação nos questionários