PM

Sistema de Faturas Eletrónicas

PROJETO DE MESTRADO

José André Ferreira da Silva
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

fevereiro | 2018

Sistema de Faturas Eletrónicas

PROJETO DE MESTRADO

José André Ferreira da Silva
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTADOR
José Luís Silva

CO-ORIENTADORA
Karolina Baras

UNIVERSIDADE da MADEIRA

Sistema de Faturas Eletrónicas

José André Ferreira da Silva

Constituição do júri de provas públicas:

Eduardo Miguel Dias Marques, (Professor Auxiliar da Universidade da Madeira), Presidente

Leonel Domingos Telo Nóbrega, (Professor Auxiliar da Universidade da Madeira), Vogal

Karolina Baras, (Professora Auxiliar da Universidade da Madeira), Vogal

Fevereiro de 2018

Funchal – Portugal

Resumo

As empresas que vendem produtos ou serviços podem emitir por ano milhares ou até mesmo milhões de faturas. Para cada fatura emitida, esta deve ser devidamente entregue ao destinatário e é necessário seguir o seguinte fluxo: primeiramente, a fatura é impressa e colocada num envelope que de seguida é enviado para o correio. Depois, o correio fica responsável por entregar a fatura ao cliente. Todo esse processo de entrega da fatura ao destinatário é dispendioso e demorado para as empresas.

Nesse sentido, o objetivo deste projeto de mestrado consiste em desenvolver um sistema de faturas eletrónicas capaz de ultrapassar todo o processo de entregas de faturas ao destinatário referido anteriormente. Ou seja, o novo sistema de faturas eletrónicas deve garantir a entrega da fatura ao destinatário sem qualquer alteração da mesma, com baixo custo e o mais rápido possível.

O sistema desenvolvido disponibiliza várias funcionalidades tais como, integração com vários sistemas de faturação, receção de faturas eletrónicas dos sistemas de faturação, envio de fatura emitidas para o destinatário, receção de faturas dos seus fornecedores, criação de ligações entre entidades, entre outras funcionalidades.

Está previsto que a solução desenvolvida de Faturas Eletrónicas seja lançada para o mercado brevemente.

Palavras-Chave

Angular

API *RESTful*

Aplicações Móveis

Faturas Eletrónicas

Laravel

Testes HTTP

Abstract

Companies that sell products or services can issue thousands or even millions of invoices a year. Each invoice issued must be properly delivered to the recipient and must follow the following flow: first, the invoice is printed and placed in an envelope that is sent to the mail. Then the mail is responsible for delivering the invoice to the customer. This whole process of delivery of the invoice to the recipient is expensive and time-consuming for the companies.

In this sense, the goal of this master's project is to develop a system of electronic invoices capable of surpassing the entire process of delivering invoices to the recipient. That is, the new electronic invoice system must guarantee the delivery of the invoice to the recipient without any change thereof, with low cost and as fast as possible.

The developed system offers several functionalities such as, integration with multiple invoicing systems, reception of electronic invoices from invoice systems, send invoice issued to the recipient, reception of invoices from its suppliers, creating links between entities, among other functionalities.

It is expected that the developed Electronic Invoices solution will be released to the market soon.

Keywords

Angular

RESTful API

Mobile Applications

Electronic Invoices

Laravel

HTTP Tests

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu orientador, Professor José Luís Silva, à minha co-orientadora, Professora Karolina Baras, por todo o apoio, orientações prestadas, pela disponibilidade incondicional e pelos conhecimentos que me transmitiram ao longo do desenvolvimento deste projeto de mestrado.

Agradeço ao Coordenador de Projetos, Engª Ricardo Garcês da ACIN iCloud Solutions, pela confiança nas minhas capacidades para desenvolver este projeto com sucesso e pela flexibilidade de horários que sempre me disponibilizou.

Também quero agradecer à minha namorada, que vivenciou de perto todo o desenvolvimento deste projeto e sempre me ajudou nos momentos mais difíceis.

Por último, mas não menos importante, quero agradecer à minha mãe, irmãos, tias, tios, primos, cunhados pelo apoio e confiança que depositaram em mim e nas minhas capacidades ao longo do meu percurso académico.

A todos, um muito obrigado!

Índice

Índice de Tabelas

Índice de Figuras

Lista de Acrónimos

AJAX – *Asynchronous Javascript and* XML

API – *Application Programming Interface*

CSS – *Cascading Style Sheets*

DRY – *Dont Repeat Yourself*

EDI – Intercâmbio Eletrónico de Dados

ERP – *Enterprise Resource Planning*

GPS – *Global Positioning System*

HTML – *HyperText Markup Language*

HTTP – *Hypertext Transfer Protocol*

IDE – *Integrated Development Environment*

IVA – Imposto sobre o Valor Acrescentado

JSON – *JavaScript Object Notation*

JWT – JSON *Web Token*

KISS – *Keep It Simple Stupid*

MVC – *Model-View-Controller*

ORM – *Object Relational Mapping*

PDF – *Portable Document Format*

PHP – *PHP: Hypertext Preprocessor*

PHQL – *Phalcon Query Language*

SCSS – *Sassy* CSS

UML – *Unified Modeling Language*

URL – *Uniform Resource Locator*

XML – *eXtensible Markup Language*

1 Introdução

1.1 Contextualização

Faturas eletrónicas são faturas assinadas eletronicamente. Esta assinatura digital é usada como identificação da autoria de documentos eletrónicos e tem a mesma autenticidade que uma assinatura em papel. Para estas assinaturas serem válidas é necessário conterem as menções obrigatórias e satisfazer as condições exigidas na lei em vigor para garantir a veracidade da sua origem e a integridade do seu conteúdo. Para tal é necessário obter um certificado disponibilizado pelas entidades certificadoras.

Estas entidades são responsáveis pela emissão de "Certificados Digitais para Assinatura Eletrónica Qualificada". Um Certificado de Assinatura Eletrónica Qualificada é um mecanismo de autenticação digital baseado em infraestruturas de chaves assimétricas, produzido através de técnicas criptográficas, apresentando o mais elevado grau de segurança entre trocas de dados em redes abertas. Os documentos de faturação emitidos cumprem os requisitos legais e fiscais definidos pela legislação Portuguesa, no que respeito à forma de emissão.

A utilização da faturação eletrónica não é obrigatória, sendo que em alguns sectores, dominados por empresas com grande poder de compra, estas exigem faturação eletrónica aos seus fornecedores, como forma de otimizarem o processo logístico e administrativo de processamento dos documentos.

Em Portugal os conceitos de faturas emitidas em *software* certificado e comunicadas na Autoridade Tributária e de faturas eletrónicas são muitas vezes confundidos. A emissão das faturas através de um *software* certificado e a sua comunicação à Autoridade Tributária de forma eletrónica, não faz com que as faturas sejam eletrónicas. Para serem consideradas faturas eletrónicas devem cumprir os requisitos legais e fiscais definidos pela legislação Portuguesa, no que respeita à forma de emissão. [31] [32]

A ACIN iCloud Solutions, é uma empresa tecnologia que está presente no mercado há quase 20 anos. Tem como objetivo a criação, manutenção e evolução de plataformas aplicacionais na *cloud*. A ACIN disponibiliza plataformas para gestão documental (iDOK), compras públicas (acinGov), prescrição eletrónicas (iMED), faturação eletrónica (iGEST), entre outras. [70]

1.2 Motivação

O iGEST é uma plataforma de gestão *online* que permite gerir a faturação das empresas [4]. Qualquer empresa que utiliza o iGEST pode emitir centenas ou até milhares de faturas por mês, que depois devem ser devidamente entregues aos seus clientes. Esse processo de entrega das faturas aos seus clientes por vezes é demorado e tem custos elevados, visto que é necessário imprimir e enviar por correio. Para imprimir as faturas ainda existem gastos com o papel e tinta de impressora.

É importante fornecer às empresas que utilizam o iGEST uma ferramenta que permita diminuir o processo e o custo de envio das faturas aos seus clientes. Através dessa ferramenta a produtividade dos colaboradores da empresa aumenta e as faturas são mantidas *online* para uma consulta rápida e simples sempre que possível. Além disso, é importante referir que a diminuição do uso do papel contribui para melhorar o meio ambiente [5].

1.3 Objetivo

O objetivo deste projeto é o desenvolvimento de uma plataforma que permita, de forma fácil e com baixo custo, a gestão, receção e envio de faturas eletrónicas com todos os requisitos legais.

É pretendido para esta solução o desenvolvimento de um *website* "*responsive*" e as aplicações móveis para *iOS* e *Android*. É importante o desenvolvimento de aplicações móveis para diferentes sistemas, visto que hoje em dia as pessoas passam mais tempo utilizando dispositivos e aplicações móveis. As aplicações móveis tornaram-se uma ferramenta de *marketing* importante para as empresas [6] [7].

Umas das características principais e importantes desta nova solução é a integração com a plataforma de faturação iGEST [4], que faz a emissão de faturas. Depois da emissão da fatura, a nova solução ficará responsável por todo o processo de entrega da fatura ao destinatário. A nova solução deve assegurar os princípios de usabilidade para garantir uma melhor utilização, aprendizagem, eficiência e segurança da solução.

1.4 Estrutura do Relatório

Este documento encontra-se estruturado em oito capítulos. No primeiro capítulo "Introdução", apresenta-se a contextualização, motivação e os objetivos para o desenvolvimento deste projeto de mestrado.

No segundo capítulo "Estado da Arte", estuda-se e compara-se as plataformas existentes no âmbito das faturas eletrónicas.

No terceiro capítulo "Desenho do Sistema", define-se os requisitos, diagramas, protótipos e arquitetura do sistema para compreender o seu funcionamento.

No quarto capítulo "*Framework*", estuda-se várias *frameworks* PHP e decide-se qual framework utilizar para o desenvolvimento do *back-end* do sistema.

No quinto capítulo "Aplicações Móveis", estuda-se os tipos de aplicações móveis existentes e as respetivas ferramentas para o seu desenvolvimento. Por fim, ainda se decidiu qual ferramenta utilizar para desenvolver a aplicação móvel.

No sexto capítulo "Implementação da API *RESTful*", apresenta-se o processo de desenvolvimento da API e os respetivos testes e resultados.

No sétimo capítulo "Implementação do *Front-End*", apresenta-se o processo de desenvolvimento do front-end do sistema e explica-se em detalhe o funcionamento de uma funcionalidade importante.

Por fim, no oitavo capítulo "Conclusões", apresenta-se as conclusões e considerações finais deste projeto e ainda algumas sugestões de trabalho futuro.

2 Estado da Arte

2.1 Introdução

No início de um novo sistema é relevante o estudo de trabalhos relacionados existentes no âmbito de faturas eletrónica. O objetivo deste estudo é reunir ideias sobre o que já existe, de forma a verificar o que pode ser feito e melhorado no novo sistema. Por isso, neste capítulo será descrito as principais soluções de faturas eletrónicas existentes no mercado.

Este estudo concentra-se essencialmente nas funcionalidades e nos aspetos de usabilidade de cada solução. Quanto a descrição das funcionalidades da solução, inclui todo o seu funcionamento como também a existência ou não das aplicações móveis para vários dispositivos, visto à importância das aplicações móveis nos dias de hoje [1] [3].

Desta forma para encontrar trabalhos relacionados com o novo sistema que é pretendido implementar, foram realizadas pesquisas com a palavra-chave "fatura eletrónica" e "*e-invoicing system*" com o intuito de encontrar plataformas *web* e aplicações móveis. Depois de acordo com as plataformas/aplicações obtidas nessas pesquisas, foi feito uma análise mais detalhada das mesmas.

2.2 Trabalhos Relacionados

2.2.1 SaphetyDoc

O SaphetyDoc [8] é uma solução disponibilizada pelo Saphety [9] que permite a criação, o envio e receção de faturas eletrónicas e outros documentos como notas de encomenda, guias de remessa, etc. Através do SaphetyDoc é possível fazer a desmaterialização completa do processo de faturação, desde a sua emissão, ao envio e à respetiva aprovação.

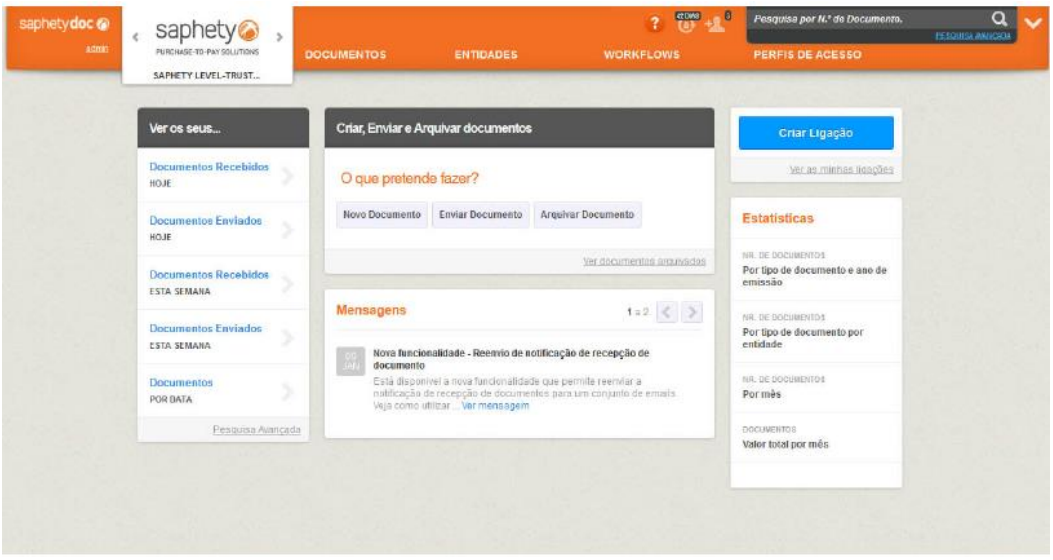


Figura 1 - Página inicial do Portal SaphetyDoc.

Para utilização desta solução é necessário efetuar o registo de uma entidade, que por sua vez poderá associar os seus utilizadores. Inicialmente com o tipo de conta gratuita, uma entidade tem acesso as seguintes funcionalidades:

- Receção de documentos

- Visualização e *download* dos formatos disponíveis
- *Upload* e manutenção de outros documentos (menos faturas)
- Solicitar receção de documentos
- Número limitado de utilizadores

No caso de subscrever o serviço SaphetyDoc de 1 ano ou 6 meses, a entidade terá como funcionalidades:

- Receção e emissão de documentos
- Criar um documento com base num existente
- Integração com ERP (*Software* de Gestão da Empresa)
- Processo de aprovação *online*
- Exportação de dados para formato Excel
- Anexar documentos ou imagens
- Solicitar receção e emissão de documentos
- Definir centro de custo
- Definição de tipo de documentos transacionados
- Número ilimitado de utilizadores e com perfis de acesso diferentes

Depois do registo, as entidades acedem ao SaphetyDoc utilizando os seus dados de registo e terão disponíveis para consulta, impressão e *download* os documentos enviados e recebidos pelas entidades que têm ligações. A consulta dos documentos enviados e recebidos podem ser pesquisados pela data de criação, data do documento, tipo de documento, número de documentos, emissor, recetor entre outros campos de pesquisa. Os documentos podem ser visualizados no próprio *website* ou então pode ser feito *download* em formato PDF. As respostas às encomendas, guias de remessa e faturas podem ser criadas diretamente no sistema através de formulário específicos.

Caso seja realizadas algumas configurações por parte do utilizador, o sistema pode envia automaticamente um *e-mail* para notificar o utilizador da chegada de um novo documento. [10]

Em termos de usabilidade podemos considerar que é razoável pois quanto à sua navegação os utilizadores não têm dificuldade em encontrar o que pretendem, não faz utilização abusiva de muitas cores numa só página, disponibiliza zona de ajuda com manuais de utilização e é compatível com os principais *browsers* (Chrome Versão 54, Microsoft Edge 38 e Firefox 49.0.2). Contudo tem alguns problemas visto que não é possível retornar a uma página anterior nem a página principal sempre que possível, os *links* não são consistentes e não são fáceis de encontra e por fim o *website* não se ajusta à resolução do ecrã.

2.2.2 NOS – Área de Cliente

A NOS [11] é uma empresa de comunicações e entretenimento portuguesa, que oferece aos seus clientes soluções fixas e móveis de última geração, televisão, internet, voz e dados para todo o tipo de mercado (pessoas, residências e empresas).

Para os seus clientes a NOS disponibiliza uma área de cliente, em que o cliente tem acesso aos seus serviços, dados de conta, notificações, permissões, faturas e pagamento, fatura eletrónicas, etc. Essa área de cliente pode ser acedida quer pela plataforma *web* ou aplicações móveis (*Android* [13] e *iOS* [14]).

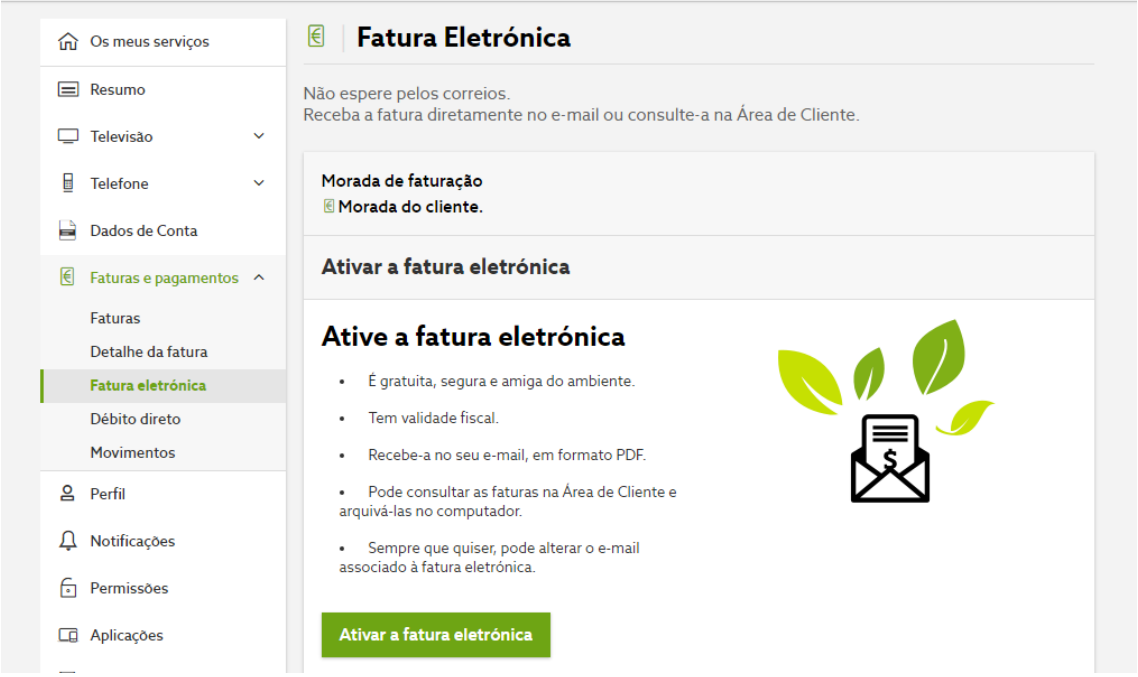


Figura 2 - Modulo de fatura eletrónica na área de cliente da NOS.

O módulo de fatura eletrónica disponibilizada na área de cliente da NOS serve de alternativa a receção das faturas em casa sem qualquer custo adicional. Para aderir a fatura eletrónica é bastante simples pois basta indicar na área de cliente o endereço de *e-mail* no qual será enviado as faturas eletrónicas. As faturas enviadas por *e-mail* também podem ser consultadas ou arquivadas na área de cliente que tem o histórico dos últimos 6 meses, garantindo a máxima segurança e confidencialidade [12].

No que diz respeito a sua usabilidade, tanto pela plataforma *web* ou pelas aplicações móveis, a área de cliente da NOS é muito simples, intuitivo e fácil de utilizar. Podemos considerar ainda que faz utilização adequada da combinação de cores, apenas utiliza informação relevante ao contexto, fornece sempre o histórico de navegação, é sempre possível aceder à ajuda ou terminar sessão, a sua plataforma *web* é compatível com os principais *browsers* (Chrome Versão 54, Microsoft Edge 38 e Firefox 49.0.2) e se ajusta à resolução do ecrã. Quanto aos aspetos negativos, é de fazer referência apenas a má utilização dos ícones no menu principal que podem levar ao erro. Podemos concluir que em termos de usabilidade a área de cliente da NOS é muito satisfatória.

2.2.3 B2BRouter.net

O B2BRouter.net [15] é uma solução destinada as empresas espanholas que permite a criação de faturas eletrónicas no formato correto e o envio das mesmas diretamente aos seus clientes. Esta solução também oferece outras funções relacionadas com o processo de faturação como a gestão de pagamentos das faturas. Para utilização desta solução não é necessária qualquer instalação, pois existe uma plataforma *web* ou aplicações móveis (*Android* [16] e *iOS* [17]) e disponibiliza um plano gratuito que é totalmente funcional.

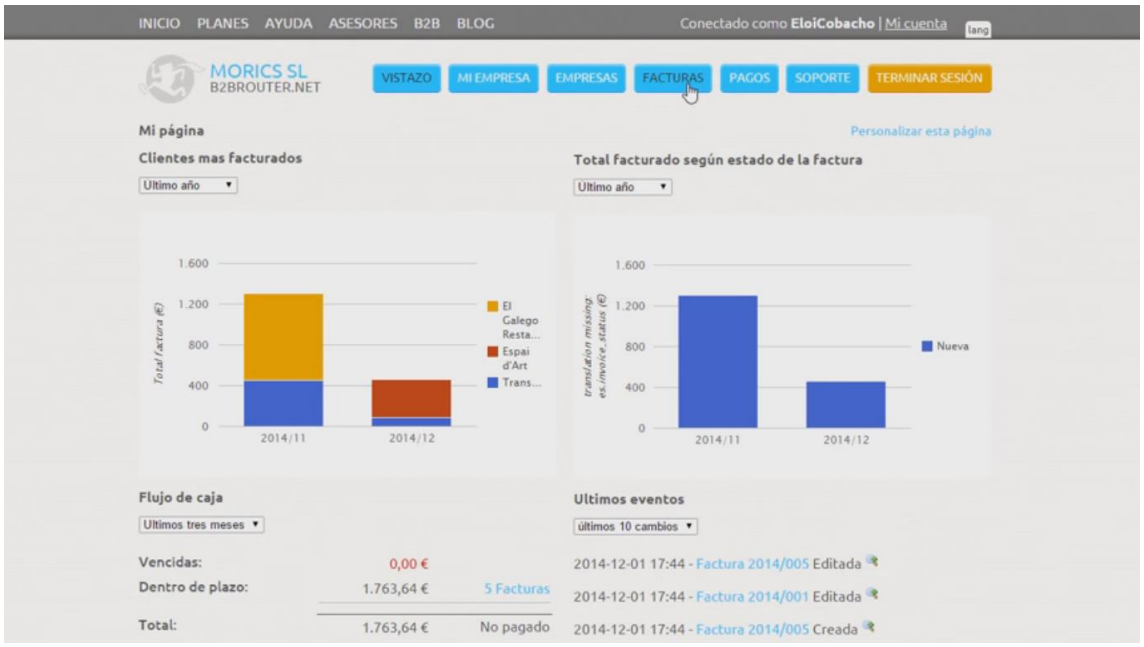


Figura 3 - Página inicial do B2BRouter.net.

O registo no sistema é rápido e simples, mas depois de iniciar sessão só é possível usufruir do sistema se preencher corretamente todos os dados fiscais da empresa. Na página inicial existe dois gráficos (um com o resumo dos clientes com mais faturas emitidas e o outro com o resumo dos estados das faturas emitidas), fluxo de caixa da empresa e um histórico dos últimos acontecimentos relacionados com as faturas e os clientes.

Com o B2BRouter.net apenas é possível criar faturas eletrónicas. Ao criar uma nova fatura é necessário preencher vários dados como os do cliente, data de emissão, número da fatura, moeda, forma de pagamento, desconto, taxas, retenções, cada produto com a respetiva quantidade, unidade, descrição, preço, etc. Depois de todos os dados preenchidos existe a possibilidade de emitir apenas a fatura ou então emitir a fatura e enviar diretamente para o cliente, que de seguida independentemente da opção selecionada é apresentado a fatura emitida. Nessa mesma página ainda é apresentado o histórico de eventos da fatura, comentários associados a fatura e várias opções tais como:

- Criar uma nova fatura
- Modificar ou eliminar a fatura
- Enviar fatura por correio eletrónico
- Mudar o estado da fatura
- Descarregar fatura em XML ou PDF
- Criar um novo modelo
- Duplicar ou retificar a fatura
- Reportar problemas com a fatura

Uma fatura eletrónica pode ter diversos estados nomeadamente o estado nova, enviada, aceite, registada, recusada ou fechada. Quando o estado da fatura é aceite, enviada ou registada é possível registar o pagamento dessa fatura.

No sistema são disponibilizadas listagens de todos os documentos emitidos ou recebidos pela empresa e também os pagamentos registados. No sentido de facilitar a pesquisa de um ou mais documentos ou pagamentos existe vários filtros de pesquisa [18].

Quanto a usabilidade do sistema B2BRouter.net verificamos que tem alguns aspetos positivos como negativos. Em termos de aspetos positivos consideramos que é um sistema fácil de utilizar, disponibiliza ajuda em qualquer parte do sistema, não faz utilização abusiva de texto, apresenta o histórico de navegação e a sua solução para a *web* é compatível com os principais *browsers* (Chrome Versão 54, Microsoft Edge 38 e Firefox 49.0.2). Relativamente aos aspetos negativos foi possível identificar que o sistema *web* não se adapta à resolução do ecrã, não faz a validação correta dos campos dos formulários e não tem opção para voltar à página anterior. Contudo podemos concluir que a usabilidade desta solução é razoável.

2.2.4 ViaCTT

A ViaCTT [19] é uma caixa postal eletrónica que permite a receção do correio em formato digital. Uma ou mais pessoas podem ter uma caixa postal com os seus documentos eletrónicos utilizando a ViaCTT, ao longo da sua vida ou existência jurídica. A utilização da ViaCTT não tem qualquer custo para os destinatários, visto que o custo é apenas suportado pelas entidades emissoras do correio.

O utilizador é capaz de escolher que remetentes podem enviar documentos em formato digital [21]. Existe várias entidades aderentes de diferentes serviços, tais como fornecedores de eletricidade, telecomunicações, água, gás, entidades bancárias ou serviços públicos [20].

Em termos de segurança, é utilizado certificados digitais de autenticação para garantir o acesso, o tratamento e a conservação dos documentos através da identificação segura de remetentes e destinatários. A adesão ao ViaCTT poderá ser feita de três modos diferente e cada utilizador decide o modo que mais lhe convém. Os três modos são os seguintes:

- **Registo no *website* ViaCTT**: O registo é efetuado no *website* com o preenchimento dos dados de identificação do utilizador e a respetiva validação enviando os documentos de identificação pessoal. Só depois da validação, a conta da ViaCTT ficará ativa.
- **Estação de Correios**: É necessário deslocar-se até uma estação de correios e mostrar a identificação, que por sua vez receberá um código temporário de acesso ao *website*. A conta ficará imediatamente ativa.
- **Portal das Finanças**: Aderir às notificações eletrónicas no Portal das Finanças, depois não será necessário qualquer identificação adicional na ViaCTT e a conta ficará ativa.

Depois de efetuar o registo e a conta estar ativa, será possível receber os documentos por correio das entidades pretendidas. Para receber documentos de qualquer entidade, é necessário proceder à subscrição das mesmas, introduzindo os dados solicitados para o efeito.

Figura 4 - Página inicial do ViaCTT.

Além de receber documentos e fazer a gestão da caixa postal, com a ViaCTT também é possível efetuar pagamentos relativo a faturas e outros documentos, enviar alertas por *e-mail* dos eventos ocorridos na caixa postal, partilhar os documentos recebidos na caixa postal com outros utilizadores, gravação dos documentos fora da caixa postal, etc [22].

Por fim em relação à usabilidade do ViaCTT é um *website* simples, intuitivo, apresenta apenas informação essencial, fornece legendas dos ícones, fornece sempre o histórico de navegação e é possível sair ou aceder ao menu de ajuda em qualquer momento. Mas, porém, não se ajusta à resolução do ecrã e não tem qualquer aplicação móvel. Podemos concluir que em termos de usabilidade a ViaCTT é razoável.

2.2.5 Tradeshift - Electronic Invoicing

O Tradeshift [23] fornece um serviço de faturação eletrónica destinado para qualquer empresa na europa. Esta solução faz a gestão dos documentos das empresas e ajuda na emissão de faturas como também nos pagamentos independentemente da sua origem.

Para começar a utilizar esta solução, é necessário fazer o registo da empresa e do utilizador que fará a gestão dessa mesma empresa. O processo de registo é rápido visto que é pedido apenas os dados essenciais da empresa e do utilizador. Depois do registo e com a sessão iniciada na plataforma é possível criar faturas, notas de crédito, ordens de compras entre outros.

Ao criar uma fatura, ou qualquer outro documento, é possível enviá-lo por *e-mail* para o cliente. No corpo do *e-mail* recebido pelo cliente, terá uma ligação de acesso ao documento enviado e assim é permitido ao cliente visualizá-lo, aceitar ou rejeitar esse documento, trocar mensagens com a empresa emissora, fazer *download* em PDF ou XML, imprimir e/ou marcar o documento como pago.

O utilizador que emite o documento tem quase as mesmas funcionalidades que o cliente, ou seja, apenas não pode aceitar nem rejeitar o documento.

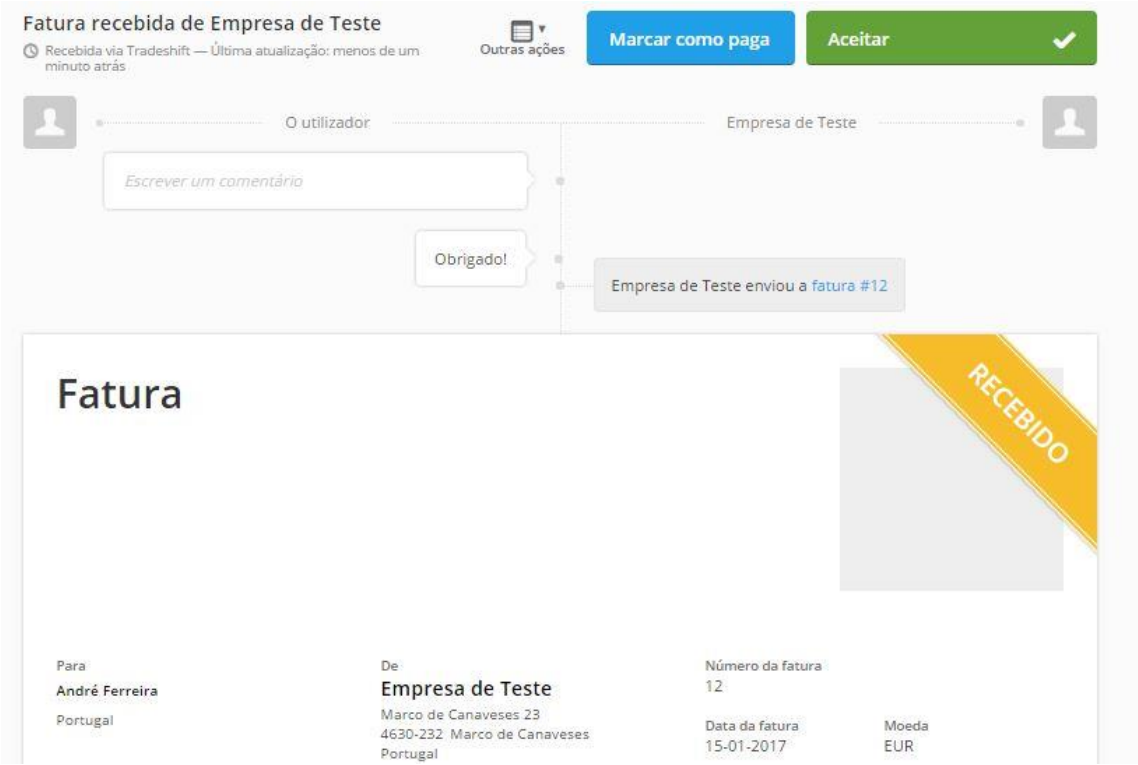


Figura 5 – Página de visualização do documento por parte do cliente.

No Tradeshift são disponibilizadas outras funcionalidades interessantes como listagem de tarefas e criação da rede da empresa. Na lista de tarefas são apresentadas tarefas em falta relacionadas com a empresa, como por exemplo um documento por concluir, falta de logótipo da empresa, adicionar detalhes relevantes sobre a empresa, etc. Quanto à criação da rede esta permite melhorar a comunicação com as empresas que lá se encontram, tornando mais fácil a troca de documentos.

No que diz respeito à usabilidade desta solução é muito satisfatória porque é muito fácil de aprender e utilizar o sistema. Apresenta apenas informação essencial, ajusta-se à resolução do ecrã, é compatível com os principais *browsers* (Chrome Versão 54, Microsoft Edge 38 e Firefox 49.0.2), disponibiliza ajuda em qualquer momento e dispõem de cinco línguas diferentes. Os únicos aspetos negativos encontrados foram de não fazer referência ao histórico de navegação, e de acordo com as pesquisas efetuadas, não existir qualquer aplicação móvel do sistema.

2.3 Conclusão

Ao longo deste capítulo acerca de trabalhos relacionados, verificou-se que através das pesquisas efetuadas que a área de faturas eletrónicas é ainda uma área pouco desenvolvida principalmente em Portugal. Pois foi possível constatar que em Portugal existe poucas soluções de faturas eletrónicas.

Na tabela 1 encontra-se algumas características principais que uma solução de faturação eletrónica pode ter, e assim é possível uma melhor comparação entre os trabalhos relacionados encontrados.

Tabela 1 - Tabela de comparação entre os trabalhos relacionados.

Características	SaphetyDoc	NOS (Área de Cliente)	B2BRouter.net	ViaCTT	Tradeshift
Emissão de faturas	Sim	Não	Sim	Não	Sim
Emissão de outros tipos de documentos	Sim	Não	Não	Não	Sim
Receção de documentos de uma entidade	Sim	Sim	Não	Sim	Sim
Receção de documentos de diferentes entidades	Sim	Não	Não	Sim	Sim
Envio de documentos de uma entidade	Sim	Não	Sim	Não	Sim
Envio de documentos para diferentes entidades	Sim	Não	Sim	Não	Sim
Fazer pagamento dos documentos	Não	Sim	Sim	Sim	Sim
Envio de documentos por correio eletrónico	Sim	Sim	Sim	Sim	Sim
Histórico de eventos dos documentos	Não	Não	Sim	Sim	Sim
Descarregar documento em XML ou PDF	Sim	Sim	Sim	Sim	Sim
Partilhar documentos com várias entidades	Sim	Não	Não	Não	Sim
Associar vários utilizadores à entidade	Sim	Não	Não	Não	Sim
Usabilidade	Aceitável	Boa	Aceitável	Aceitável	Boa
Aplicação móvel	Não	Sim	Sim	Não	Não

Este estudo é muito importante na fase inicial do novo sistema, pois foi possível reunir algumas funcionalidades e ideias das soluções existentes que podem ser implementadas no novo sistema. Também se obtiveram alguns aspetos de usabilidade que deve ser tomado em conta para uma melhor utilização deste novo sistema.

3 Desenho do Sistema

3.1 Introdução

Para o desenvolvimento de um novo sistema, o desenho do sistema é a parte fundamental no processo de desenvolvimento de sistemas. Nesta etapa é importante definir e compreender todo o funcionamento do novo sistema, as necessidades do cliente, o processo de negócio e a comunicação com outros sistemas, visto que uma correta definição e compreensão do sistema são essenciais para o sucesso do desenvolvimento.

Portanto, neste capítulo será apresentado os requisitos legais, requisitos funcionais requisitos não funcionais, casos de utilização, diagrama de *classes*, diagramas de estado, diagramas de atividade, modelo relacional, protótipos e por último a arquitetura do sistema.

3.2 Requisitos Legais

A faturação eletrónica obedece a regras específicas (como a assinatura eletrónica avançada ou a emissão por sistema de intercâmbio eletrónico de dados) e deve ser gerada por um programa certificado. Os requisitos legais exigidos para os programas de faturação e para as condições técnicas de emissão de faturas eletrónicas asseguram o cumprimento da condição de integridade do conteúdo das faturas. Não pode ser confundido o envio da fatura por *e-mail* com a fatura eletrónica. Há requisitos legais a cumprir. Os requisitos a respeitar são as seguintes:

Tabela 2 - Requisitos legais

Nº	Requisito
RL01	A autenticidade da origem de cada fatura eletrónica ou documento equivalente.
RL02	A integridade do conteúdo da fatura eletrónica ou documento equivalente.
RL03	A integridade da sequência das faturas eletrónicas ou documentos equivalentes (validação cronológica).
RL04	Arquivar, em suporte informático, as faturas eletrónicas emitidas e recebidas por via eletrónica.
RL05	A manutenção, durante o período de 10 anos previsto no Código do IVA, da autenticidade, integridade e disponibilidade do conteúdo original das faturas emitidas e recebidas por via eletrónica.
RL06	O não repúdio da origem e receção das mensagens.
RL07	A não duplicação das faturas emitidas e recebidas por via eletrónica.
RL08	Mecanismos que permitam verificar que o certificado emitido pelo emissor da fatura eletrónica não se encontra revogado, caduco, suspenso na respetiva data de emissão.

Existem duas formas de garantir a autenticidade da origem e a integridade do conteúdo das faturas, já referidas anteriormente, que são:

- Utilização da assinatura eletrónica avançada;
- Utilização do sistema de Intercâmbio Eletrónico de Dados (EDI).

A assinatura eletrónica avançada, nos termos do Decreto-Lei n.º 290-D/99, de 2 de Agosto [63], deve preencher os seguintes requisitos:

- Identificar de forma unívoca o titular como autor do documento;
- A sua aposição no documento depende apenas da vontade do titular;
- É criada com meios que o titular pode manter sob seu controlo exclusivo;
- A sua conexão com o documento permite detetar toda e qualquer alteração posterior do conteúdo deste.

O EDI (Intercâmbio Eletrónico de Dados), é um sistema que permite a transferência eletrónica, de computador para computador, de dados comerciais e administrativos utilizando uma norma acordada para estruturar uma mensagem EDI. Uma mensagem EDI é um conjunto de segmentos estruturados utilizando uma norma acordada, preparados num formato legível em computador e que podem ser processados automaticamente e sem ambiguidades. Na prática, este sistema traduz-se por um registo do destinatário da fatura no endereço eletrónico da empresa emitente com um *ID user* e uma *password* de acesso à fatura eletrónica. [29] [30]

3.3 Requisitos Funcionais

Os requisitos funcionais descrevem as operações e atividades que o sistema deve ser capaz de realizar para satisfazer as necessidades dos utilizadores. Durante o levantamento de requisitos foi encontrado requisitos de diferentes tipos, tais como requisitos do sistema, do utilizador e do documento. Com isso obtiveram-se os seguintes requisitos funcionais:

Tabela 3 - Requisitos funcionais

Nº	Requisito	Nº RL
RF01	O sistema deve possibilitar o registo de novas entidades.	
RF02	O sistema deve disponibilizar a subscrição de serviços.	
RF03	O sistema deve possibilitar a receção de documentos provenientes do iGEST.	
RF04	O sistema deve ser capaz de enviar automaticamente os documentos provenientes do iGEST para o destinatário definido.	
RF05	O sistema deve possibilitar a troca de documentos entre entidades.	
RF06	O sistema apenas deve permitir a troca de documentos autorizados pelo remetente e destinatário.	
RF07	O sistema deve possibilitar reportar problemas pelo utilizador.	
RF08	O sistema deve disponibilizar permissões de acesso aos utilizadores da entidade.	
RF09	O sistema deve alertar aos utilizadores do sistema da sua gestão de ligações das suas entidades.	
RF10	O sistema deve notificar os utilizadores da entidade, sempre que a entidade receber um novo documento.	
RF11	O sistema deve guardar e disponibilizar o histórico de eventos ocorridos no sistema.	
RF12	O sistema deve possibilitar a importação de documentos.	
RF13	O sistema deve possibilitar a impressão de documentos.	
RF14	O sistema deve possibilitar o envio de documentos por e-mail.	
FR15	O sistema deve garantir a assinatura eletrónica avançada.	1 e 2
RF16	O sistema não deve permitir duplicar documentos recebidos ou enviados.	7
FR17	O sistema deve ser capaz de verificar se os certificados não se encontram revogado, caduco ou suspenso.	8
RF18	O documento deve ter vários estados.	
RF19	O sistema deve mudar o estado do documento conforme as operações realizadas pelos utilizadores.	

RF20	O sistema deve ser capaz de manter os documentos eletrónicos durante 10 anos, previsto no código do IVA.	5
RF21	O sistema deve ser capaz de provar a origem e a receção dos documentos.	6
RF22	O sistema deve garantir que não acontece quaisquer alterações ao conteúdo do documento quando é transmitido	3
RF23	O utilizador deve ser capaz de anular subscrições não pagas.	
RF24	O utilizador deve ser capaz de enviar e arquivar documentos.	4
RF25	O utilizador deve ser capaz de aprovar ou recusar documento recebidos.	
RF26	O utilizador deve ser capaz de convidar entidades que não estão registadas no sistema.	
RF27	O utilizador deve ser capaz de visualizar os documentos eletrónicos recebidos e enviados.	
RF28	O utilizador deve ser capaz de gerir os utilizadores associados a entidade.	
RF29	O utilizador deve ser capaz de fazer pedidos de ligação com outras entidades, para a troca de documentos eletrónicos.	
RF30	O utilizador deve ser capaz de cancelar um pedido de ligação.	
RF31	O utilizador deve ser capaz de aceitar ou rejeitar pedidos de ligações.	
RF32	O utilizador deve ser capaz de cancelar ligações.	
RF33	O utilizador deve ser capaz de visualizar os pedidos de ligações pendentes e canceladas.	
RF34	O utilizador deve ser capaz de visualizar as entidades com os quais foram estabelecidas ligações.	

3.4 Requisitos Não-Funcionais

Quanto aos requisitos não funcionais referem-se a um conjunto de atributos que o sistema deve ter, tais como os níveis de desempenho, segurança, confiabilidade, disponibilidade e usabilidade. Os requisitos não funcionais foram os seguintes:

Tabela 4 - Requisitos não funcionais

Nº	Requisito
RNF01	A interface do sistema deve ser visualmente agradável para os utilizadores.
RNF02	A navegação no sistema para o utilizador deve ser intuitiva.
RNF03	Qualquer interação com o sistema deve dar *feedback* ao utilizador.
RNF04	O sistema deve mostrar o histórico de navegação para o utilizador.
RNF05	O utilizador deve ser capaz de aprender a utilizar o sistema em menos de 3 minutos.
RNF06	A base de dados deve apoiar qualquer crescimento na recolha de dados.
RNF07	A palavra-passe guardada na base de dados deve ser encriptada.
RNF08	O sistema deve fornecer políticas de privacidade a todos os utilizadores.
RNF09	O sistema deve bloquear o utilizador quando errar três vezes na palavra-passe.
RNF10	O utilizador bloqueado deve receber um *e-mail*, permitindo-lhes recuperar a palavra-passe.
RNF11	O *back-end* do sistema para *web* deve ser desenvolvido com linguagem PHP.
RNF12	O *front-end* do sistema para *web* deve ser desenvolvido com linguagem HTML, CSS e Angular.

3.5 Casos de Utilização

O modelo caso de utilização é um modelo que indica como diferentes tipos de utilizadores interagem com o sistema. Como tal, descreve de forma simples e resumida os tipos de utilizadores ou sistemas (ou seja, atores) que vão interagir com o sistema, as ações que os utilizadores ou sistemas vão realizar no sistema, e por fim as relações entre os atores e as ações do sistema.

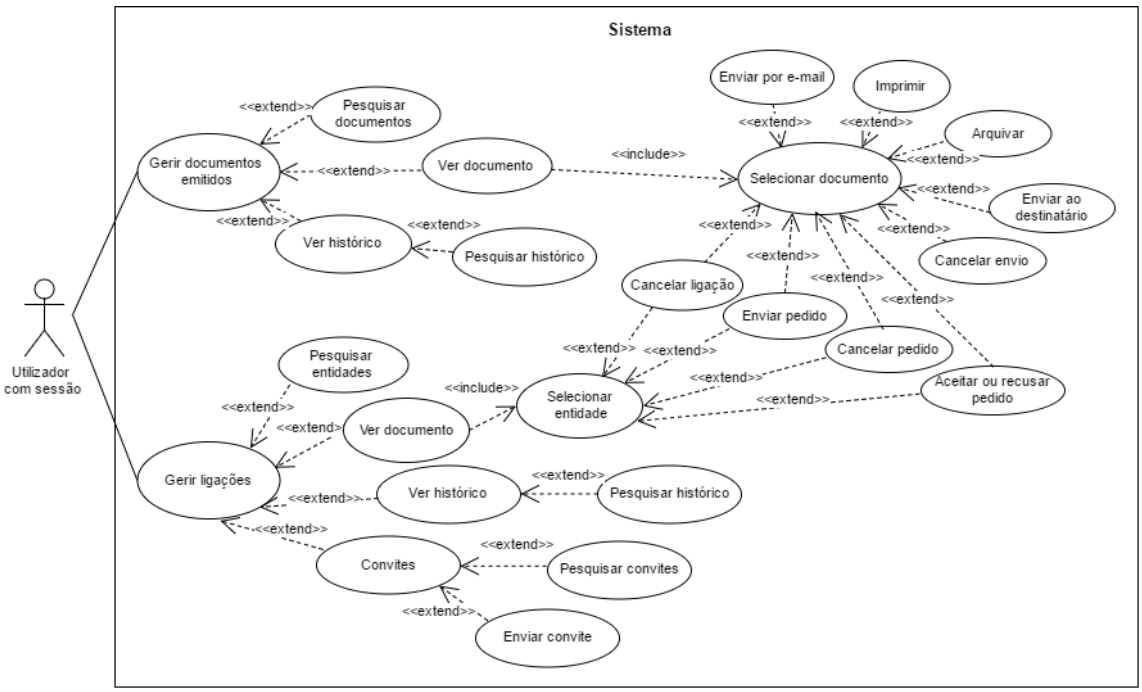


Figura 6 - Parte do diagrama casos de utilização do utilizador com sessão

No âmbito deste sistema foram identificadas várias ações, relações e 4 atores tais como, utilizadores sem sessão, utilizadores com sessão, administradores e um sistema externo. Para cada ator foi realizado um modelo de casos de utilização.

Tabela 5 - Descrição e diagramas de casos de utilização dos atores que interagem com o sistema

Ator	Descrição	Figura
Utilizador sem sessão	Como o próprio nome indica, é aquele que não tem sessão iniciada no sistema e apenas tem acesso as funcionalidades públicas do sistema.	Figura 34
Utilizador com sessão	Também como o próprio nome indica, é o tipo de utilizador que tem a sessão iniciada no sistema.	Figura 36
Administrador	Utilizador responsável por gerir o sistema e por isso tem acesso a toda informação do sistema.	Figura 37
Sistema externo	Refere-se aos sistemas externos que vão interagir com o novo sistema, como por exemplo o sistema de faturação iGEST.	Figura 35

3.6 Diagrama de Classes

O diagrama de classes é um diagrama do UML que representa os tipos de classes de um sistema, como interagem entre si e qual a responsabilidade de cada uma dessas classes.

A criação do diagrama de classes é útil para o desenvolvimento do sistema, visto ter todas as classes que o sistema necessita e é a base para a construção de outros diagramas (como por exemplo o diagrama de estados). O diagrama de classes também é importante para o desenvolvimento da base de dados do sistema, pois cada classe do diagrama representa uma tabela da base de dados.

Para o desenvolvimento do diagrama, começou-se por abstrair todas as classes do sistema e os respetivos atributos. Depois de todas as classes abstraídas, passou-se por realizar a ligação e a cardinalidade entre as classes. Na figura 7 segue-se apenas as classes relacionadas com o utilizador, permissões e notificações do sistema.

Por causa da dimensão do diagrama de classes completo, achou-se por bem colocar em Anexo B.

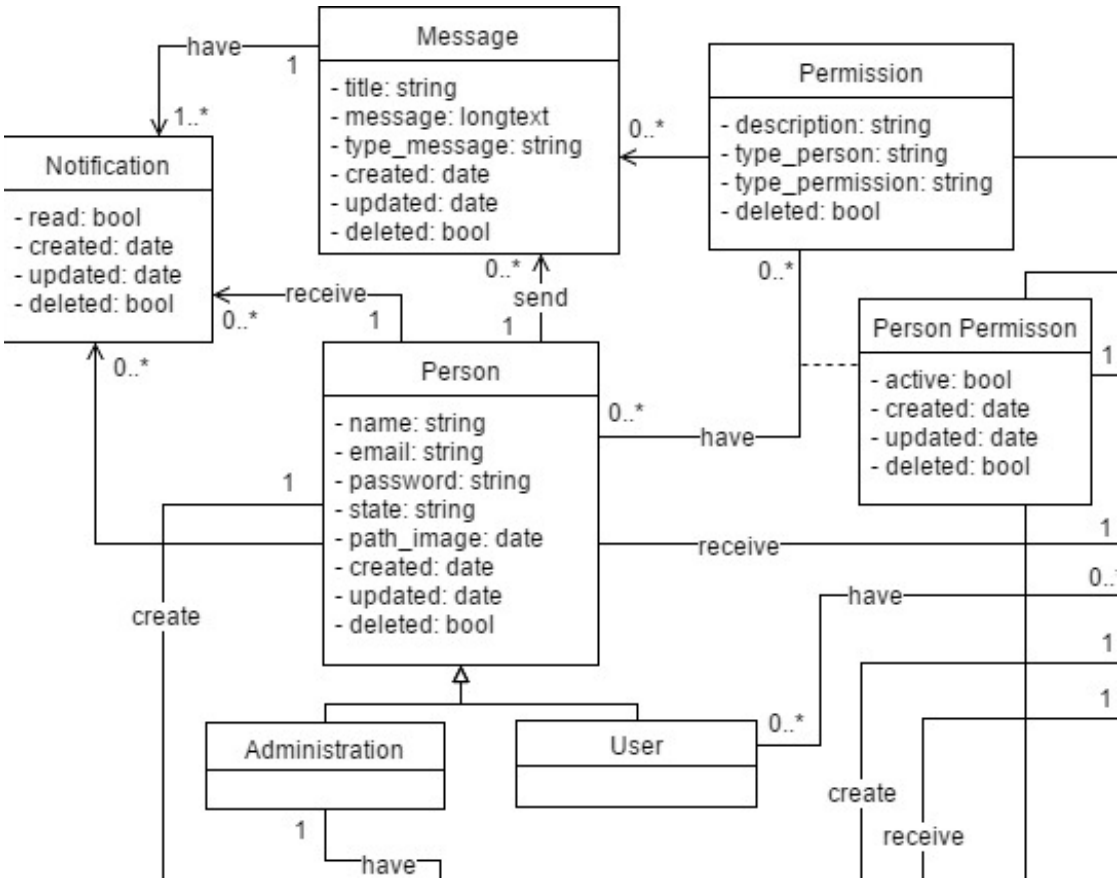


Figura 7 - Classes relacionadas com o utilizador, permissões e notificações do sistema.

3.7 Diagramas de Estado

O diagrama de estado é um tipo de diagrama que descreve o comportamento de um objeto num sistema, deste a sua existência até a sua destruição. Neste tipo de diagrama é representado os possíveis estados do objeto, as transições entre os estados, os eventos que fazem ocorrer as transições e ainda as operações que podem ser executadas dentro dos estados ou durante uma transição.

Neste sistema achou-se por bem a criação de diagramas de estados dos documentos e das ligações, devido a sua importância no sistema. Para os documentos, criou-se um diagrama de estados para os documentos emitidos e outro para os documentos recebidos. Os

documentos emitidos podem ser documentos criados através do novo sistema ou documentos enviados por sistemas externos (como por exemplo, o sistema de faturação iGEST), e os documentos recebidos são documentos enviados pelo emissor do documento e recebidos pelo destinatário do documento através do novo sistema.

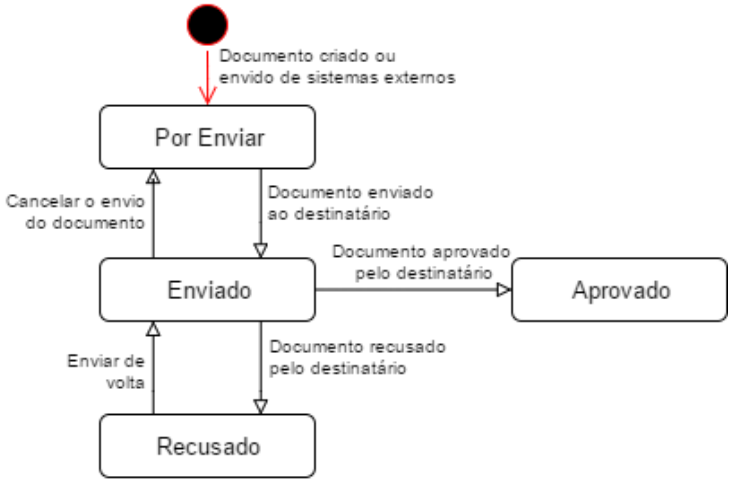


Figura 8 - Diagrama de estados dos documentos emitidos.

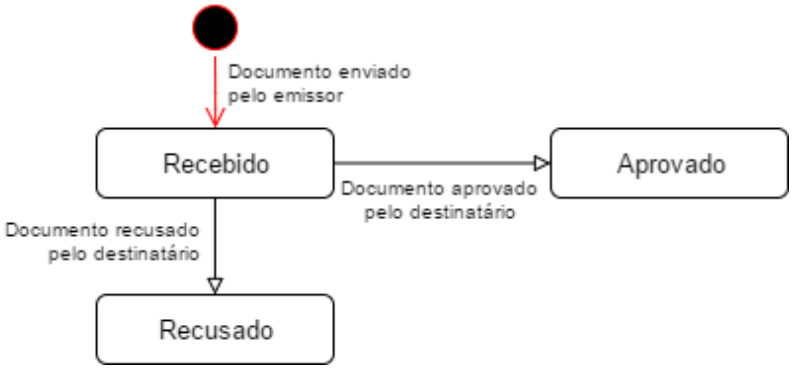


Figura 9 - Diagrama de estados dos documentos recebidos.

Também para as ligações, criou-se um diagrama de estados para as ligações enviadas e outro para as ligações recebidas. As ligações enviadas são os pedidos de ligação enviados para outras entidades para possibilitar a troca de documentos eletrónicos, e as ligações recebidas é a situação contrária.

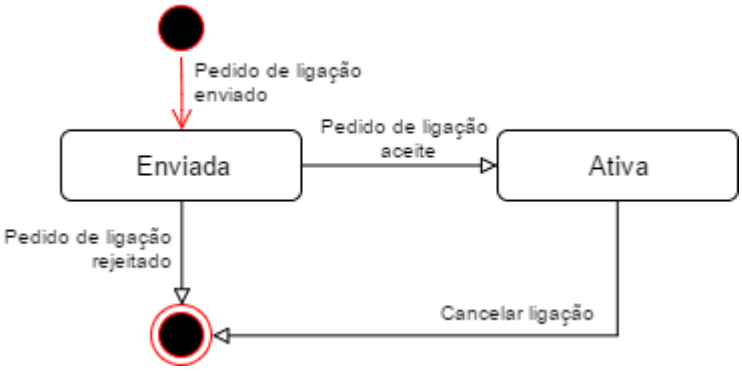


Figura 10 - Diagrama de estados das ligações enviadas.

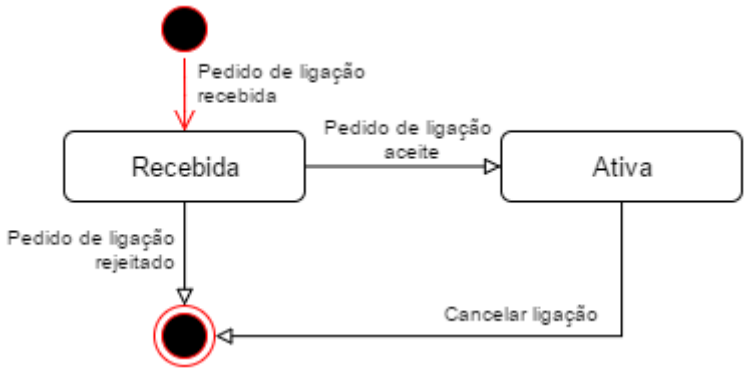


Figura 11 - Diagrama de estados das ligações recebidas.

3.8 Diagramas de Atividade

O diagrama de atividade fornece o comportamento de um sistema entre os seus utilizadores e tem como objetivo mostrar o fluxo de atividade para executar uma determinada tarefa. Os diagramas de atividades correspondem aos conhecidos fluxogramas.

Foram realizados 32 diagramas de atividade e para cada diagrama foi realizado a respetiva descrição. Devido ao grande número de diagramas de atividade, em seguida apenas serão descritos e apresentados os 3 diagramas mais importantes, sendo que os restantes serão apresentados no Anexo B.

Começou-se por descrever a criação da nova adesão no sistema. Inicialmente para criar uma nova adesão o utilizador deve selecionar a página adesão, em que o sistema vai responder a atividade com o formulário para realizar a adesão. O utilizador deve preencher os dados da entidade e do utilizador e submeter para o sistema. O sistema verifica se todos os campos foram preenchidos e se os dados são válidos, e notifica o utilizador caso ocorra algum erro. Se não ocorrer nenhum erro, o sistema verifica se o utiliza já existe. No caso de existir o utilizador, o sistema regista a entidade e associa o utilizador, caso contrário o sistema primeiro regista o utilizador e envia um *e-mail* para inserir a palavra-passe, e só depois regista a entidade e associa o novo utilizador. Por fim apresenta a página para iniciar sessão.

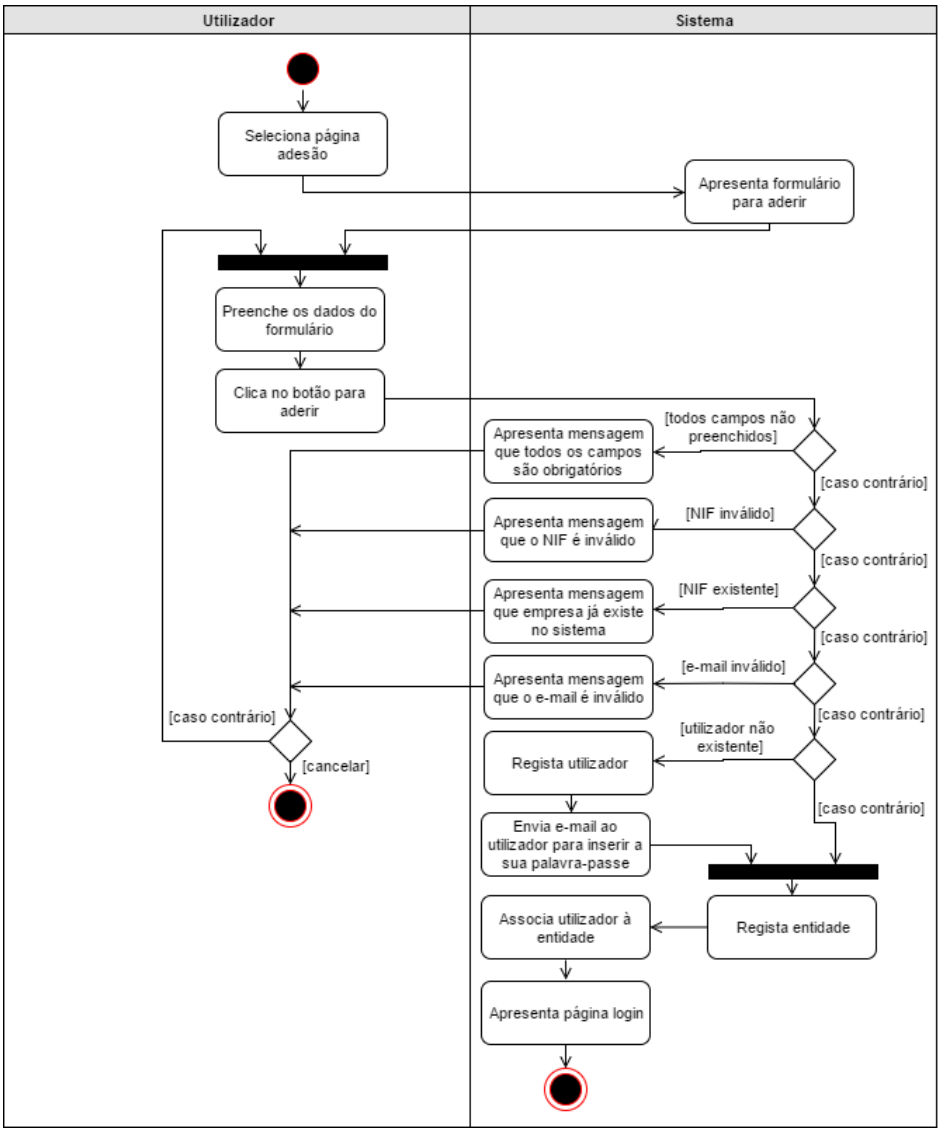


Figura 12 - Diagrama de atividade para nova adesão.

Para o utilizador iniciar sessão no sistema deve aceder a página para iniciar sessão. O sistema verifica se o utilizador já tem sessão iniciada, e caso tenha sessão iniciada o sistema atualiza os dados de sessão e apresenta a página inicial conforme o tipo de utilizador. Uma vez que não exista sessão iniciada, o sistema apresenta o formulário para iniciar sessão.

O utilizador preenche e submete o formulário para o sistema, e este por sua vez verifica se os campos do formulário estão todos preenchidos e se os dados são validos. Se ocorrer algum erro o sistema notifica o utilizador, caso contrário inicia sessão e apresenta a página inicial conforme o tipo de utilizador. Na eventualidade de o tipo de utilizador ser normal, ainda é necessário escolher a entidade se o utilizador estar associado a várias entidades, senão a única entidade é selecionada automaticamente por defeito.

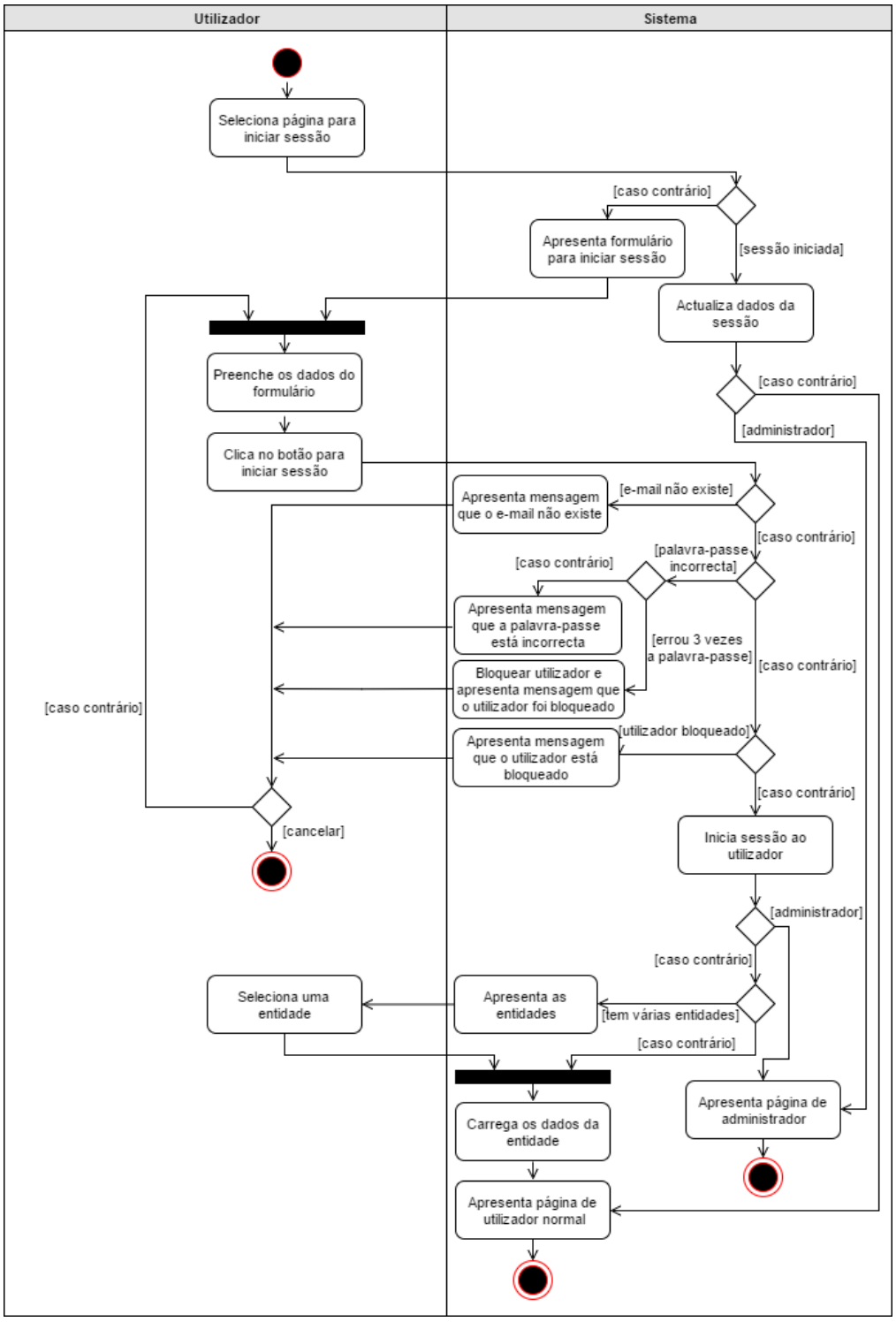


Figura 13 - Diagrama de atividade para iniciar sessão.

Tendo em conta o envio de documento emitidos, o utilizador deve ter sessão iniciada no sistema para enviar documentos. Primeiramente o utilizador deve visualizar os documentos emitidos e o sistema apresenta os documentos emitidos apenas se o utilizador tiver permissões. Depois o utilizador seleciona o documento que pretende enviar e o sistema apenas possibilita o envio do documento caso exista ligação com o destinatário do documento, caso o documento ainda não tenha sido enviado ou se foi enviado não foi aceite. Ainda é

possível enviar o pedido de ligação caso o destinatário esteja registado no sistema, ou então enviar um convite se não tiver registado.

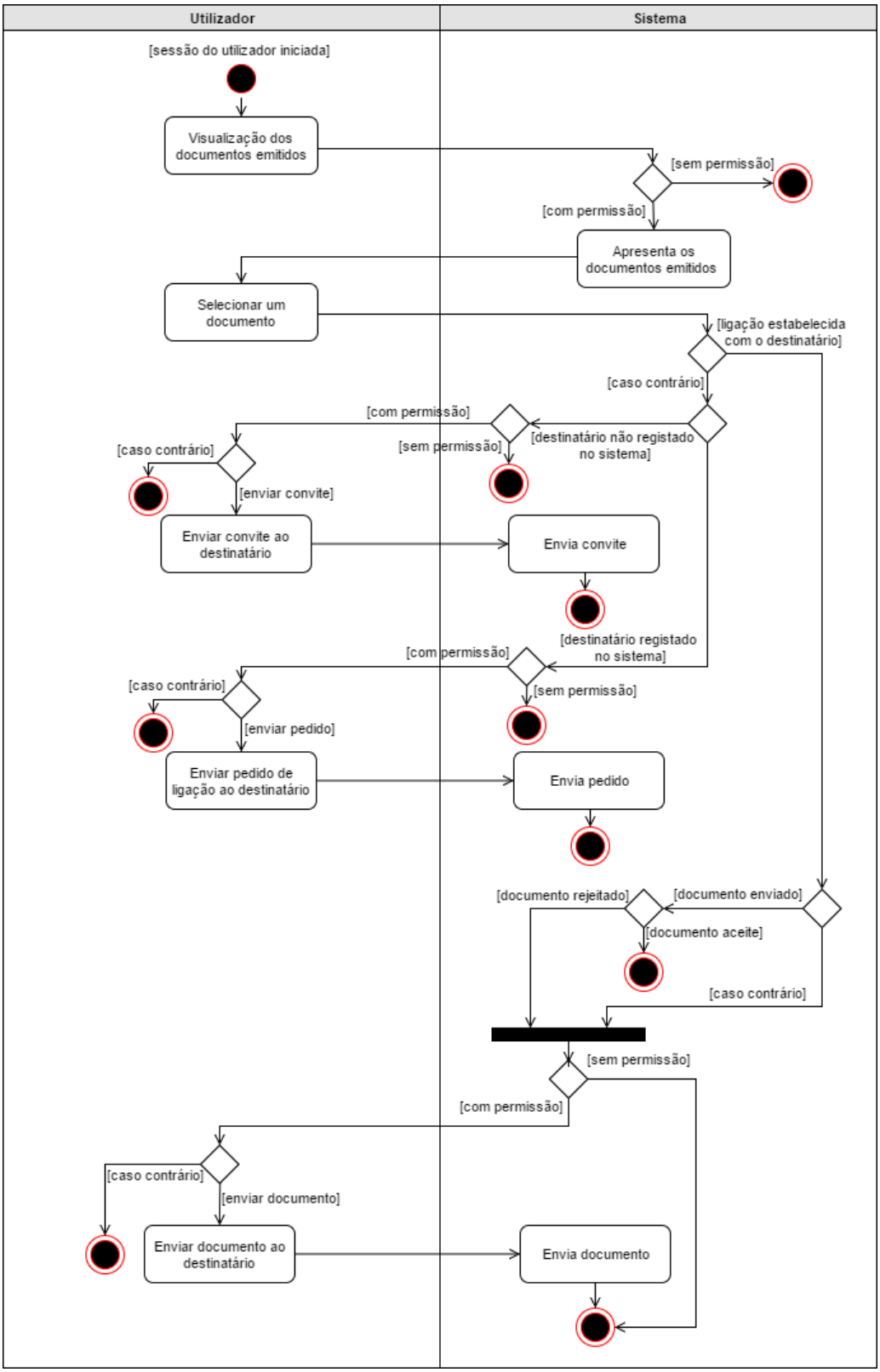


Figura 14 - Diagrama de atividade para enviar documentos emitidos.

3.9 Modelo Relacional

Para o desenvolvimento do novo sistema, foi necessário criar uma base de dados para guardar todos os dados do novo sistema de forma simples e clara. Começou-se por utilizar o diagrama de classes criado anteriormente, dado que uma classe do diagrama de classes corresponde a uma tabela na base de dados. Depois acrescentou-se algumas tabelas importantes, para que os dados fossem armazenados de forma consistente e sem repetições.

Na figura 15 segue-se uma parte da base de dados, ou seja, as tabelas, atributos, e relações necessárias para guardar e manter as informações dos utilizadores e as suas respetivas permissões. Devido a dimensão e de alguma complexidade da base de dados completa do sistema, decide-se apresentar no Anexo C.

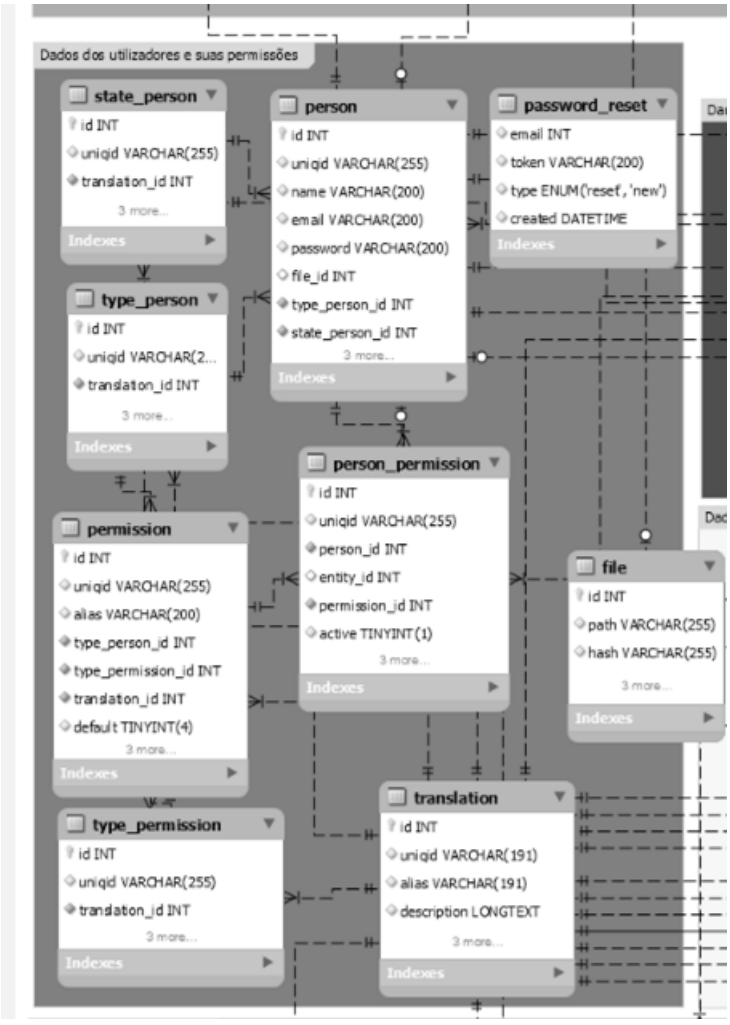


Figura 15 - Parte da base de dados para guardar informações dos utilizadores.

3.10 Protótipos

No desenvolvimento de qualquer novo sistema de *software*, a realização de protótipos de baixa fidelidade é importante para o sucesso futuro do novo sistema. Por vezes apenas o levantamento de requisitos do sistema com o consumidor final não é suficiente, também pode ser necessário a realização de protótipos do novo sistema para perceber os detalhes do mesmo. A realização de protótipos deve ser feita na fase de planeamento do sistema, tendo tempo e custo reduzido.

Assim sendo, para o novo sistema realizou-se protótipos de baixa fidelidade utilizando a plataforma *online moqups* (www.moqups.com). A plataforma *moqups* disponibiliza um conjunto de componentes e *templates* (como por exemplo menus, campos para formulário, listas, tabelas, diagramas, gráficos, botões e ícones) para criar protótipos de forma rápida e simples. A plataforma não só permite criar páginas, como também realizar ligações entre as mesmas para simular a navegação do futuro sistema.

Começou-se por criar os protótipos do sistema para o utilizador normal, o utilizador administrador e por fim para os protótipos com a integração do sistema de faturação iGEST. Esses protótipos foram realizados com base nos princípios de usabilidade, para fornecer uma melhor utilização do sistema aos seus utilizadores. Na figura 16 segue-se um exemplo dos protótipos realizados inicialmente.

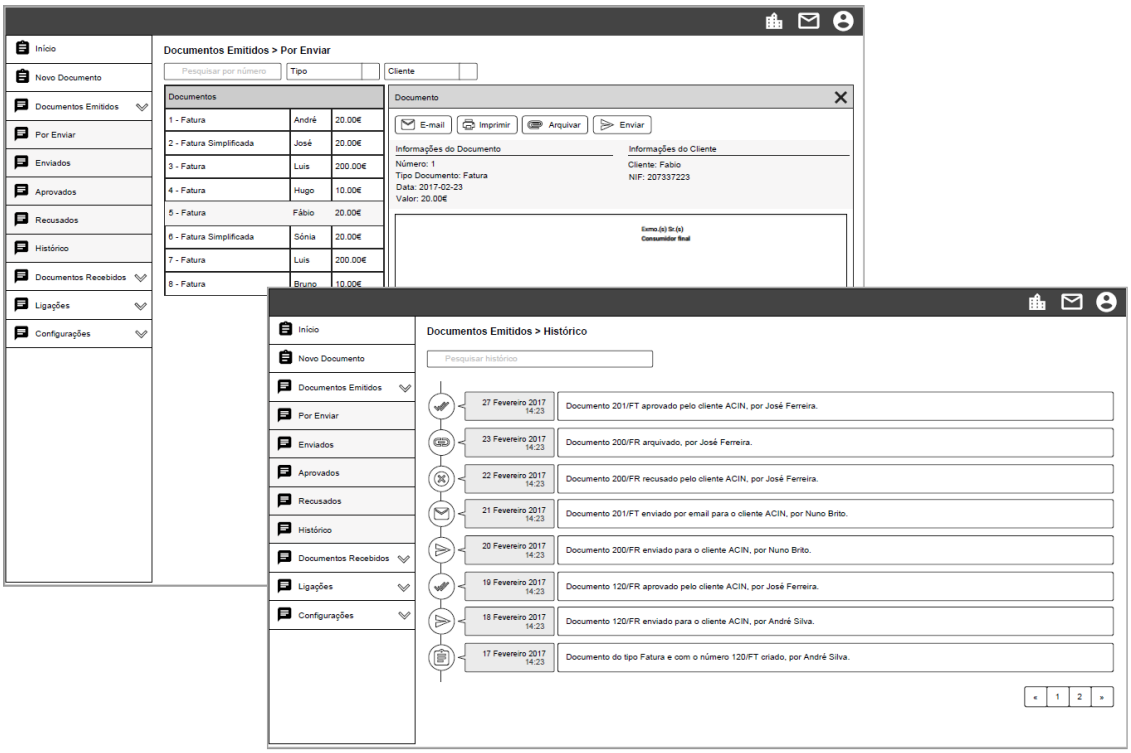


Figura 16 - Protótipo da página de visualização e histórico de documentos emitidos.

Logo após a criação da primeira versão dos protótipos do sistema, foi realizado alguns testes com oito utilizadores. Nesses testes os utilizadores seguiram cenários que envolvem as ações mais importantes do sistema. Os cenários e as tarefas associadas a cada cenário estão apresentados no Anexo D. Durante a realização dos testes com os utilizadores, registou-se o tempo de execução das tarefas de cada cenário, o número de erros cometidos e os comentários indicados pelos utilizadores.

Depois no final dos testes, os utilizadores ainda responderam a um questionário relacionado com a realização das tarefas associadas aos cenários e a usabilidade do sistema em geral. O questionário e os seus resultados encontram-se no Anexo E, no entanto os tempos e erros utilizadores está no Anexo F.

Em seguida, foram realizadas melhorias aos protótipos do sistema de acordo com os problemas e comentários indicados pelos utilizadores. Depois os utilizadores voltaram a testar

até obter uma solução do sistema satisfatória. No Anexo G encontra-se os protótipos finais do sistema.

Quanto aos protótipos do sistema na versão mobile, por questão de tempo apenas foi realizado protótipos de algumas páginas do sistema. Na figura 17 segue-se os protótipos do sistema na versão mobile das páginas novo documento, documento emitidos, visualização de documentos emitidos e histórico de documento emitidos. Contudo, as restantes páginas do sistema em versão *mobile* funcionará de forma semelhante ao sistema na versão normal.

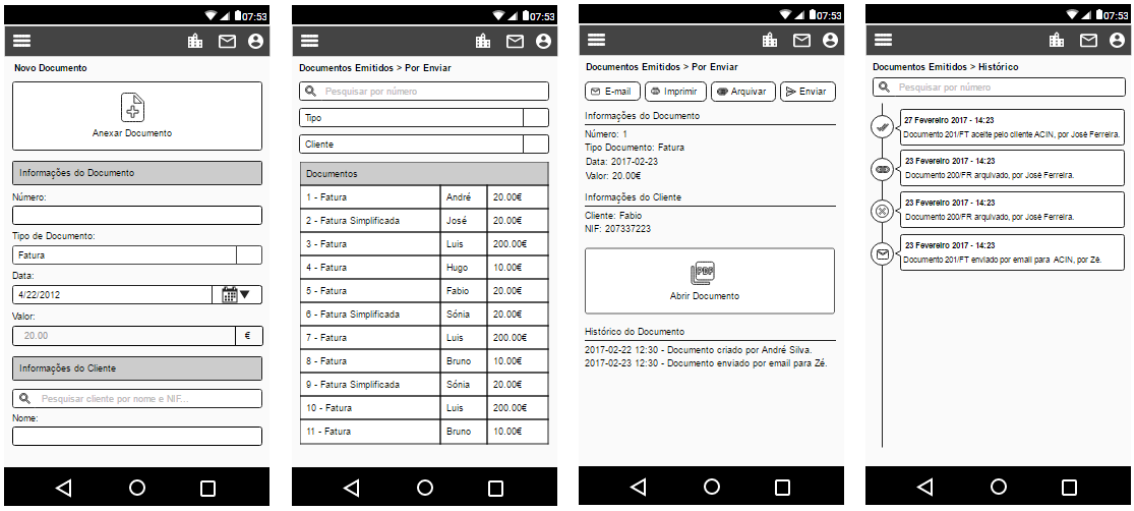


Figura 17 - Protótipos do sistema na versão mobile.

3.11 Arquitetura do Sistema

Nos dias de hoje, cada vez mais os sistemas estão ligados uns com os outros para a partilha de informação. Essa ligação é estabelecida através de uma API (*Application Programming Interface*), que disponibiliza um conjunto de estruturas, operações, protocolos e ferramentas para construção de sistemas. Assim permite que outros sistemas utilizem as suas características.

Como já foi referido anteriormente é pretendido a criação da uma plataforma web, uma aplicação móvel para várias plataformas e ainda a comunicação com a plataforma de faturação iGEST. Por isso, foi criado uma API que contém toda a lógica de negócio do novo sistema, permitindo a comunicação com cada um dos sistemas. A arquitetura completa do sistema está representada na figura 18.

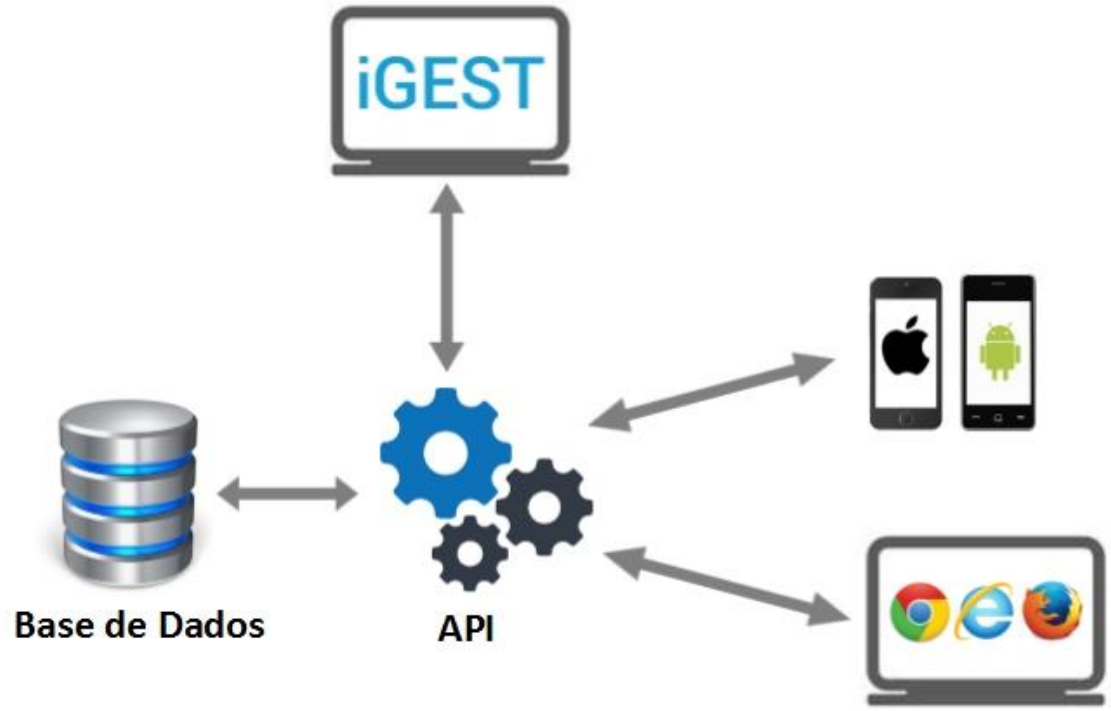


Figura 18 - Arquitetura completa do sistema.

Em termos de utilização da API, cada um dos sistemas utilizam a API quando necessitar apagar, obter, atualizar dados. No futuro caso seja necessário alterar ou adicionar recursos a lógica de negócio do sistema, apenas será feito diretamente na API e assim facilitando o desenvolvimento.

3.12 Conclusão

Ao longo deste capítulo, descreveu-se detalhadamente os requisitos do sistema e como os diferentes utilizadores vão interagir com o sistema. Ainda neste capítulo foi especificado a arquitetura e os protótipos do sistema, como também será armazenado os seus respetivos dados.

O levantamento de requisitos e os diagramas arquiteturais foram importantes para compreender o sistema, e assim desenhar o modelo relacional e a interface do sistema. Além de desenhar a interface, ainda foi realizado teste de usabilidade com utilizadores para melhorar a interface do sistema. Os resultados dos testes de usabilidade realizados aos vários utilizadores foram satisfatórios, visto que, à maioria dos utilizadores realizam as tarefas propostas com sucesso, com poucas falhas e em pouco tempo.

Por fim, mas não menos importante, com a especificação da arquitetura do sistema neste capítulo foi possível compreender o que é pretendido para o novo sistema de faturas eletrónicas. A arquitetura do sistema também será importante para futuras decisões nos próximos capítulos.

4 Framework

4.1 Introdução

Neste capítulo, será apresentado uma descrição geral sobre o que é e para que serve uma *framework,* e ainda serão apresentadas as seis melhores *frameworks* de 2016 existentes no mercado [49] [50]. Na descrição de cada *framework* engloba a sua descrição geral, características, vantagens e desvantagens de utilização. As informações de cada *framework* serão de acordo com as pesquisas realizadas em diferentes fontes.

As *frameworks* descritas ao logo deste capítulo serão apenas *frameworks* PHP, visto que um dos requisitos não funcionais é a utilização de linguagem PHP para o desenvolvimento do *back-end* da nova solução. Por isso as seis *frameworks* apresentadas serão Phalcon, Laravel, Symfony, Yii, CakePHP e CodeIgniter.

Por fim, será decidido qual da *framework* utilizar no *back-end* do projeto com base nas vantagens e desvantagens de cada *framework* e ainda de acordo com os exercícios práticos de utilização.

4.2 O que é e para que serve uma *framework*?

Uma *framework* é uma estrutura com um conjunto de código, que se relaciona entre si para disponibilizar funcionalidades específicas ao programador de *software*. Isso significa que é necessário escrever menos código, que por sua vez garante menos riscos de erros. A utilização de um a *framework* é importante por muitas razões, tais como:

- Facilidade e rapidez no processo de desenvolvimento.
- Reutilização e organização de código.
- Escalabilidade e segurança das aplicações.

4.3 Estudo de *Frameworks*

4.3.1 Phalcon

O Phalcon [51] é uma *framework* PHP *open-source*, escrita com extensão da linguagem C e é otimizada para alto desempenho. Apesar da extensão do C, não é preciso utilizar C, visto que o seu desenvolvimento é em PHP. Utiliza a versão 5.5 do PHP ou superior e em termos de arquitetura, é baseado no modelo MVC (*Model-View-Controller*). Quanto as suas vantagens e desvantagens, podemos considerar:

Tabela 6 - Vantagens e desvantagens da framework Phalcon. [52]

Vantagens
Fornece uma interface gráfica para gerir os controladores, modelos, etc.
Só carrega os ficheiros que serão utilizados.
Alto desempenho.
Utilização do *Phalcon Query Language* (PHQL), que permite consulta a base de dados utilizando linguagem *SQL-like*.
Oferece recursos *cache* no ORM (*Object Relational Mapping*) para evitar o acesso contínuo a base de dados.
Fornece componentes para gestão da cache, paginação, formulários, etc.
Desvantagens

Curva de aprendizagem maior, uma vez que existe poucos tutoriais e cursos *online*.
Processo de instalação longo e difícil.

4.3.2 Lavarel

Laravel [53] é uma *framework* PHP *open-source* utilizada para o desenvolvimento *web* e utiliza como arquitetura o padrão MVC. Algumas características importantes do Laravel é a sua simplicidade, várias formas de aceder a base de dados, várias ferramentas úteis para o desenvolvimento do sistema e é compatível com o PHP 5.6.4 ou superior. Tem como principal objetivo ajudar a desenvolver aplicações *web* seguras, rápidas e com código simples. O Laravel utiliza o *composer* que faz a gestão das dependências e oferece as seguintes vantagens e desvantagens:

Tabela 7 - Vantagens e desvantagens da framework Laravel. [54]

Vantagens
Grandes comunidades disponíveis e em crescimento que fornecem suporte e ajudas.
Testes unitários utilizando o PHPUnit. Ainda disponibiliza testes HTTP, testes de *browser* e testes de base de dados.
Integração com a biblioteca *SwitfMailer*, para o envio de *emails*.
Facilidade de integração de bibliotecas.
É baseado em *ORL Eloquent*.
Pronto para facilitar a criação de serviços *RESTful*.
Desvantagens
Problemas nas atualizações entre versões, ou seja, algumas funcionalidades podem deixar de funcionar nas versões mais recentes.
Complexidade da estrutura dificulta a instalação em geral.

4.3.3 Symfony

O Symfony [55] é uma *framework* PHP *open-source* e segue o modelo MVC para desenvolver aplicações *web*, construídas em cima dos componentes Symfony. Os conjuntos de componentes disponibilizadas pelo Symfony são reutilizáveis e são utilizadas para a construção de aplicações PHP como por exemplo Drupal e Laravel. O Symfony tem como objetivo construir aplicações robustas e tem ferramentas adicionais para ajudar nos testes, *debugging* e documentação. Podemos ainda considerar algumas vantagens e desvantagens do Symfony, tais como:

Tabela 8 - Vantagens e desvantagens da framework Symfony. [56]

Vantagens
Mecanismo de base de dados independentes.
Fácil de instalar e configurar.
Criação de código automático.
Utiliza os princípios DRY (*Dont Repeat Yourself*) e KISS (*Keep It Simple Stupid*).
Utiliza o *Doctrina* ORM, para tornar o acesso a base de dados fácil e flexível.
Utiliza a *Cache* HTTP, tornando os pedidos do sistema mais rápidos.
Integração com o PHPUnit, permitindo testes unitários e funcionais.
Desvantagens
Pouca documentação e comunidades *online*.
Dificuldades na atualização de versões.
Mecanismo de segurança difícil de utilizar.

4.3.4 Yii

Yii [57] é uma *framework* PHP *open-source* de alto desempenho para o desenvolvimento de aplicações. É construído em torno do padrão MVC e fornece um conjunto de recursos seguros para criar aplicações robustas rapidamente. O Yii é completamente orientado a objetos, e é baseado no princípio de codificação DRY (*Don't-Repeat-Yourself*) para fornecer uma base de código limpa e lógica.

A versão mais recente da framework Yii é o Yii 2, e é compatível no mínimo com o PHP 5.4.0 e no máximo com o PHP 7.x. Também é integrado com *jQuery*, e como tal vem com um conjunto de recursos habilitados para AJAX. Podemos ainda considerar algumas vantagens e desvantagens da framework Yii, tais como:

Tabela 9 - Vantagens e desvantagens da framework Yii. [58]

Vantagens
Utilização do *Active Record*, que fornece uma interface de orientada a objetos para controlo da base de dados. Tem suporte com base de dados relacional (por exemplo *MySQL*, *PostgreSQL* e *SQLite*) e base de dados *NoSQL* (por exemplo *Redis* e *MongoDB*).
Integração com a *framework Codeception* que permite realizar testes unitários, testes funcionais e testes de aceitação.
Suporta mecanismos de cache de dados, fragmentos, página e HTTP.
Geração de código baseado na *web* através do Gii e extensões para *debugging*.
Conjunto de ferramentas para simplificar a criação de APIs de serviço *web RESTful*.
Desvantagens
Poucas comunidades *online*.
Sem atualizações desde muito tempo e sem notícias sobre uma nova versão da *framework*.

4.3.5 CakePHP

O CakePHP [59] é uma *framework* de desenvolvimento rápido em PHP, grátis e *open-source*. A versão mais recente do CakePHP, ou seja, versão 3, funciona com o PHP 7 ou no mínimo com o PHP 5.5.9. Como a maioria das *frameworks* PHP o CakePHP também utiliza o padrão MVC e tem uma estrutura apropriada para o desenvolvimento, manutenção, implementação de aplicações.

Esta *framework* é indicada para o desenvolvimento de aplicações *web* de menor complexidade. Em termos de vantagens e desvantagens podemos considerar:

Tabela 10 - Vantagens e desvantagens da framework CakePHP. [60]

Vantagens
Utilização do *composer* para gerir as dependências.
Disponibiliza componentes para validação, segurança, sessão, autenticação, envio de email, entre outros.
Consola Bake, que permite criar modelos, vistas, controladores, casos de testes, componentes e *plugins*.
Estrutura para gerir idiomas das aplicações.
Utiliza ORM, para facilitar o acesso a base de dados.
Integração com o PHPUnit.
Desvantagens
Ferramentas de *debugging* disponíveis pouco úteis.
Documentação disponível é inadequada.

Problemas na atualização de uma versão para outra mais recente.

4.3.6 CodeIgniter

CodeIgniter [61] é uma *framework open-source* para o desenvolvimento de aplicações web em PHP, fornecendo um conjunto de ferramentas simples. Tem o objetivo de desenvolver aplicações mais rápidas escrevendo pouco código. Como outras *frameworks* PHP, o CodeIgniter utiliza o padrão MVC e é compatível com a versão 5.6 do PHP ou superior. Em seguida, segue-se as vantagens e desvantagens de utilização do CodeIgniter:

Tabela 11 - Vantagens e desvantagens da framework CodeIgniter. [62]

Vantagens
Processo de instalação rápido e simples.
Fácil de aprender, manter, desenvolver e tem um excelente desempenho.
Fornece URL's limpos, utilizando uma abordagem baseada em segmentos.
Desvantagens
Não tem o ORM, para facilitar o acesso a base de dados.
Menos bibliotecas integradas e ferramentas em comparação com outras frameworks.

4.4 Escolha da *Framework*

Com o estudo efetuado e conforme os requisitos do sistema, será possível comparar as seis *frameworks* de *back-end* e decidir qual delas utilizar.

De acordo com um dos requisitos funcionais, o sistema deve comunicar com a plataforma iGEST e no futuro possivelmente com outros sistemas, logo por isso é necessário criar uma API *RESTful*. Devido a essa necessidade a *framework* CodeIgneiter é excluída das opções, porque é a única *framework* que não fornece qualquer ferramenta ou componente para criação de serviços *RESTful*. Outras características importantes para a escolha da *framework* para o *back-end* são a curva de aprendizagem, documentação e material de apoio da *framework*. As *frameworks* Phalcon, Symfony e CakePHP têm problemas de documentação e a curva de aprendizagem é acentuada. Logo por causa desses problemas, essas *frameworks* também são excluídas.

Depois de todas as comparações de cada *framework*, verificou-se que as *frameworks* Laravel e Yii são as mais indicadas para desenvolver o *back-end* do novo projeto. Para uma escolha mais correta, foi realizado um pequeno exercício prático com *frameworks* Laravel e Yii. Os exercícios realizados encontrem-se detalhados no Anexo A.

Após a realização dos exercícios práticos utilizando as duas *frameworks*, em termos de tempo obtiveram-se aproximadamente os descritos na tabela 12:

Tabela 12 - Tempos do exercício prático com as frameworks Laravel e Yii.

	Laravel	Yii
Instalação e Configurações	1 hora	1 hora e 20 minutos
Implementação	8 horas	9 horas
Total	**9 horas**	**10 horas e 20 minutos**

Estas diferenças de tempos entre as *frameworks* Laravel e Yii, foram principalmente devido a documentação. Na existência de alguma dificuldade na implementação dos exercícios, notou-se que através das grandes comunidades, documentação e tutoriais

existentes da *framework* Laravel era mais fácil de ultrapassar. Posto isto, de acordo com as vantagens e desvantagens, realização dos exercícios práticos e os requisitos do sistema, a *framework* Laravel é a mais indicada para o desenvolvimento do projeto.

4.5 Conclusão

Ao longo deste capítulo foram estudadas várias *frameworks* PHP, com o objetivo de conhecer cada uma das *frameworks* em geral e quais as suas vantagens e desvantagens de utilização. Ainda foram realizados exemplos práticos de utilização com duas *frameworks*, neste caso Laravel e Yii, para uma escolha mais correta da *framework* a utilizar no projeto.

De acordo com as características, vantagens e desvantagens das *frameworks*, requisitos do sistema e os exemplos práticos de utilização, achou-se que a *framework* Laravel era a mais indicada para o desenvolvimento do projeto. A escolha correta da *framework* será importante a médio e longo prazo para o desenvolvimento rápido e consistente do projeto.

5 Aplicações Móveis

5.1 Introdução

Nos dias de hoje, as aplicações móveis estão cada vez mais presentes no nosso dia-a-dia. Com o crescimento do número de pessoas que acedem a internet via *smartphone* e *tablet*, as aplicações móveis tem a capacidade de atingir um grande número de pessoas [1] [3]. Devido ao grande crescimento das aplicações móveis, as empresas investem cada vez mais no desenvolvimento de aplicações para apresentar os seus produtos de forma rápida e eficiente aos seus clientes [2]. Em termos de desenvolvimento, uma aplicação pode ser do tipo nativa, híbrida ou *web*.

Neste capítulo, segue-se a descrição dos tipos de aplicações existentes e das ferramentas necessárias para desenvolver cada tipo de aplicação. Este estudo será importante para uma decisão adequada sobre qual dos três tipos de aplicação móvel à desenvolver.

5.2 Aplicações Nativas

As aplicações nativas são desenvolvidas para um único sistema móvel específico. Por exemplo, aplicações para *iOS* são desenvolvidas em *Objective-C* e aplicações *Android* em *Java*.

Devido à utilização da linguagem nativa de cada plataforma nas aplicações nativas, o acesso às funcionalidades do sistema e sensores (exemplos: *GPS*, Câmara, etc.) é facilitado, normalmente estas possuem melhor desempenho, armazenam mais dados *offline.* Contudo, podem ser mais caras e levam mais tempo para desenvolver. [24] [25]

5.3 Aplicações Híbridas

Por sua vez uma aplicação híbrida é desenvolvida para múltiplas plataformas utilizando uma única linguagem (por exemplo *C#* ou *HTML5* e *JavaScript*) e de seguida é compilada para ser executada em cada plataforma definida.

O acesso às funcionalidades do sistema e sensores (por exemplos GPS, Câmara, Contactos, etc.) são feitos através de *plug-ins*. Uma das vantagens de uma aplicação híbrida é a rapidez e a facilidade de desenvolvimento. Também é mais fácil de manter, mas no caso de uma aplicação de maior dimensão pode não ter melhor desempenho. Quanto ao desenvolvimento de uma aplicação híbrida, pode ser desenvolvida utilizando tecnologias *web* e depois executada através de um *web view* ou então através de ferramentas como *Apache Cordova*, *Rubymotion* e *Xamarin*. [25] [26]

5.3.1 Apache Cordova

O *Apache Cordova* [33] é uma plataforma *open-source* para o desenvolvimento de aplicações móveis, que podem ser compiladas para vários sistemas como *Android*, *iOS*, *Windows Phone* ou *Windows*. O desenvolvimento de aplicações móveis utilizando o *Apache Cordova* é feito em *HTML5*, *CSS3* e *JavaScript* e ainda são disponibilizadas algumas *frameworks* e ferramentas para ajudar no desenvolvimento das aplicações, como por exemplo:

- Ionic: É uma *framework* gratuita para o desenvolvimento de aplicações móveis com *HTML5*, *CSS* e *JavaScript*. O *Ionic* utiliza *AngularJS* no desenvolvimento do *front-end* e ainda oferece um conjunto de componentes já desenvolvidos, como por exemplo

listas, formulários, botões, *tabs*, etc. Por isso o *Ionic* simplifica, facilita e aumenta a produtividade no desenvolvimento de aplicações móveis. [35]

- GapDebug: É uma ferramenta gratuita para fazer *debugging* das aplicações móveis em vários dispositivos *iOS* e *Android*. [36]
- Monaca: É um ambiente de desenvolvimento para aplicações móveis em múltiplos sistemas e proporciona um desenvolvimento na "nuvem" através do *browser*. O *Monaca* fornece todas as ferramentas necessárias para o desenvolvimento móvel entre plataformas, incluindo codificação, *debugging* e compilação. É disponibilizada uma subscrição grátis com 3 projetos *online*, 250MB de armazenamento, desenvolvimento local e notificações *push*. [37]
- Onsen UI: Tal como o *Ionic*, *Onsen UI* também é uma *framework* gratuita para facilitar o desenvolvimento de aplicações móveis, utiliza *Angular* para o *front-end* e disponibiliza componentes prontos para utilizar. [34]

Também são disponibilizadas *APIs* que permitem aceder aos recursos nativos do dispositivo como notificações, contactos, câmara, etc. A utilização desta plataforma traz algumas vantagens, como por exemplo: [38] [39] [40]

- Código Único: Aplicação apenas é desenvolvida em *HTML*, *CSS* e *JavaScript* e exportado para funcionar em vários sistemas.
- Curva de Aprendizagem: A curva de aprendizagem é reduzida, considerando a utilização do *HTML*, *CSS* e *JavaScript*.

Em termos de desvantagens, apenas fazer referencia a performance das aplicações desenvolvidas. Mas na maioria dos casos as aplicações têm problemas de performances devido a má utilização das tecnologias. Para evitar esses problemas existe algumas dicas que podem ser consideradas, como por exemplo: [38] [39]

- Evitar recarregar a páginas em cada ação do utilizador.
- Fazer o mínimo de pedidos possíveis e utilizar o *localStorage* para armazenar o máximo de dados.
- Evitar animações com *JavaScript*.

5.3.2 RubyMotion

RubyMotion [41] é uma plataforma que permite criar aplicações nativas para *iOS*, *Android* e *OS X*, utilizando *Ruby* como linguagem de programação para o seu desenvolvimento. Para utilização do *RubyMotion* é preciso um computador *Mac* com o sistema *OS X 10.9* ou superior e não funciona em outro qualquer sistema *Windows* ou *Linux*. O *RubyMotion* não é gratuito, mas disponibiliza uma versão gratuita totalmente funcional aos seus clientes com algumas limitações. Essa versão serve para os possíveis compradores avaliarem o *RubyMotion* antes de comprar.

Como a plataforma *Apache Cordova,* o *RubyMotion* também oferece todo o suporte necessário para aceder aos recursos dos sistemas *iOS*, *Android* e *OS X*. Quanto às suas vantagens, o *RubyMotion* oferece os mesmos benefícios que as plataformas de tecnologias *web* para o desenvolvimento de aplicação híbrida se ainda tem um melhor desempenho. Essa melhor performance das aplicações utilizando o *RubyMotion* é devido a compilação estática em código máquina das aplicações. [42]

5.3.3 Xamarin

Xamarin [43] é uma plataforma de desenvolvimento de aplicações móveis *cross-platform* em C#. Esta solução é totalmente grátis e ainda está integrada com o Visual Studio. Como todas as outras plataformas para criação de aplicações híbridas, o *Xamarin* utiliza apenas uma linguagem de programação (neste caso o C#) e fornece a API nativa para acesso aos recursos dos dispositivos. O *Xamarin* inclui um conjunto de ferramentas para diferentes fases do ciclo de desenvolvimento das aplicações, tais como:

- Xamarin Forms: Tem vários *layouts* criados como botões, gráficos, mapas, textos, animações, formulários e listas que podem ser usados para criar interfaces do utilizador de forma rápida e simples. [44]
- Xamarin Test Cloud: É uma solução que fornece ferramentas para fazer testes automatizados de aceitação das aplicações móveis em diferentes sistemas. Com isso é possível garantir que as aplicações móveis executam de forma eficiente e correta em diferentes sistemas com pouco esforço. [45]
- HockeyApp: Permite gerir as aplicações móveis de vários sistemas como *iOS*, *Android, Windows* e *OS X.* Torna fácil a distribuição da versão beta das novas aplicações aos utilizadores e possibilita obter relatórios de falhas, métricas do utilizador, feedback de qualquer etapa do ciclo de vida das aplicações. Também pode ainda incluir aplicações de produção. [46]

5.4 Aplicações *Web*

Uma aplicação *web* é uma aplicação que não precisa de instalação. Muitas vezes é fácil de desenvolver porque usa linguagens *web*. A sua execução é efetuada por qualquer *browser*.

Desenvolver uma aplicação *web* pode ser mais rápido e mais económico que desenvolver uma aplicação nativa ou híbrida, mas, no entanto, pode ser mais lento, menos intuitivo e não pode estar em qualquer loja de aplicações móveis (exemplos: *App Store* [28] ou *Google Play* [27]). Embora a maioria dos recursos do dispositivo móvel estejam acessíveis, ainda existem alguns que não são possíveis de utilizar, tais como notificações e a capacidade de funcionar em segundo plano. [24] [25]

5.5 Conclusão

Através deste estudo sobre os tipos de aplicações móveis, conclui-se que existe cada vez mais soluções fiáveis para o desenvolvimento de aplicações móveis. Segue-se uma análise em detalhe das diferenças entre os três tipos de desenvolvimento de aplicações, sob o ponto de vista de algumas características. Essa análise pode ser efetuada de acordo com a tabela 13.

Tabela 13 - Comparação dos três tipos de desenvolvimentos de aplicações.

Características	Aplicações Nativas	Aplicações Híbridas	Aplicações *Web*
Tempo de Desenvolvimento	Alto	Medio	Baixo
Custo de Desenvolvimento	Alto	Medio	Baixo
Custo de Manutenção	Alto	Medio	Baixo
Desempenho	Alto	Medio	Baixo
Acesso API Nativa	Sim	Sim	Não
Modo Offline	Sim	Sim	Limitado
Interface do Utilizador	Bom	Bom	Media
Microfone	Sim	Sim	Limitado

Acelerômetro	Sim	Sim	Limitado
Armazenamento Local	Sim	Sim	Limitado
Acesso aos Contatos	Sim	Sim	Não
Notificações	Sim	Sim	Não
GPS	Sim	Sim	Limitado
Câmara	Sim	Sim	Limitado
Funcionamento em 2º plano	Sim	Sim	Não

De acordo com a tabela 13 concluímos que o desenvolvimento de uma aplicação nativa leva tempo e é mais trabalhosa e uma aplicação híbrida e *web* acaba sendo mais rápida e mais barata. Em termos de desempenho, as aplicações nativas têm melhor desempenho que as outras aplicações, mas mesmo assim as aplicações híbridas também têm bom desempenho. O acesso aos recursos nativos dos dispositivos por vezes é limitado ou inacessível pelas aplicações *web*.

Em conclusão, os recursos disponíveis, tempo de desenvolvimento e as necessidades do negócio são importantes para decidir que tipo de aplicação desenvolver [47] [48]. Por isso o tipo de aplicação a desenvolver será do tipo híbrida devido a necessidade de criar uma aplicação para vários sistemas em pouco tempo e com poucos recursos. Quando à ferramenta para o desenvolvimento da aplicação híbrida será utilizado o *Apache Cordova* juntamente com *Ionic* dado que utiliza linguagens *web* e a curva de aprendizagem será reduzida comparando com as outras ferramentas.

6 Implementação da API RESTful

6.1 Introdução

Depois de definir as especificações do sistema e decidir qual é a *framework* a utilizar, passou-se à implementação do sistema. Neste capítulo será apresentado todo o processo de desenvolvimento do *back-end* (API *RESTful*) e os testes realizados para testar a API.

Primeiro, decidiu-se por desenvolver o *back-end* (API *RESTful*) e só depois o *front-end* porque como foi referido na secção 3.11, a API (*Application Programming Interface*) contém toda a lógica de negócio do sistema.

O desenvolvimento da API foi realizado de acordo com os protótipos apresentados na secção 3.10, pois só assim foi possível compreender os tipos de pedidos e respostas implementados na API *RESTful* do novo sistema de faturas eletrónicas. Ainda neste capítulo, além dos testes realizados, também serão apresentados os resultados de avaliação desses mesmos testes.

6.2 Desenvolvimento

6.2.1 Arquitetura

Uma API *RESTful* tem como arquitetura um conjunto de recursos (normalmente identificados por URL's ou *endpoints*) e um conjunto verbos HTTP (GET, POST, PUT, DELETE, etc) para manipular os recursos.

Na figura 19 é apresentado um exemplo prático da arquitetura de uma API *RESTful*.

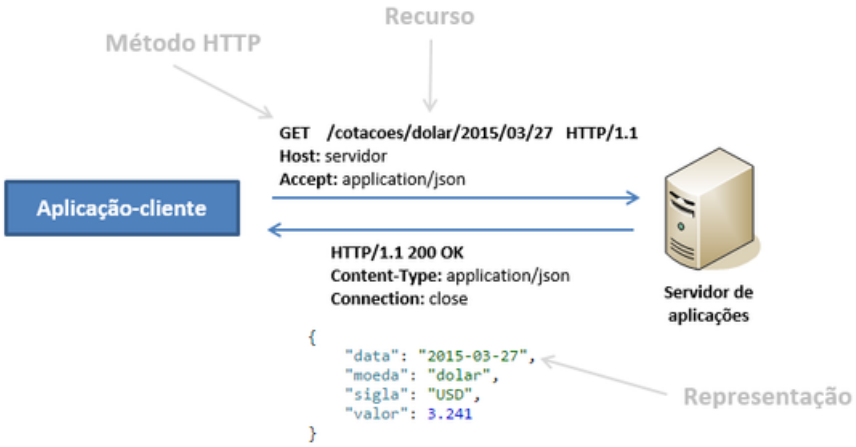


Figura 19 - Arquitetura de uma API RESTful. [64]

Em termos de representação, esses recursos podem apresentar os dados em vários formatos, como por exemplo JSON (*JavaScript Object Notation*) ou XML (*eXtensible Markup Language*). Cada representação tem associado um código de estado (ou *status code*), que indica o resultado da representação.

6.2.2 Recursos

Como já foi referido anteriormente, cada recurso é identificado por um URL. Na API *RESTful* do novo sistema, os URL's têm a estrutura representada na figura 20.

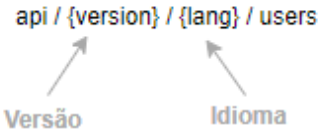


Figura 20 - Estrutura do URL.

Existem recursos públicos e privados. Os recursos públicos não precisam de autenticação, enquanto os recursos privados necessitam de autenticação de uma plataforma, utilizador ou administrador autorizado.

Na tabela 14 mostra alguns exemplos de URL's públicos e privados existentes para obter recursos da API do sistema.

Tabela 14 - Exemplos de URL's da API do sistema.

Tipo	URL	Descrição
Privado	api/v1/pt/users	Obter todos os utilizadores do sistema.
Privado	api/v1/pt/users/12	Obter utilizador com o identificador 12.
Privado	api/v1/pt/admins	Obter todos os administradores do sistema.
Privado	api/v1/pt/admins/13	Obter administrador com o identificador 13.
Público	api/v1/pt/states/persons	Obter todos os estados de pessoa.
Público	api/v1/pt/types/persons	Obter todos os tipos de pessoa.

6.2.3 Autenticação

A utilização do **JWT (*JSON Web Token*)** foi a forma utilizada para autenticar o acesso aos recursos privados da API RESTful do sistema, pois deste modo é possível controlar e identificar quem acede a esses recursos. O **JWT (*JSON Web Token*)** é um padrão (RFC 7165) que define de forma segura o envio de objetos JSON entre aplicações. O JWT é composto por 3 partes, tais como *cabeçalho*, conteúdo e *assinatura*. [65]

O **cabeçalho** (ou *header*) tem o tipo de *token* (typ) que é JWT, e o algoritmo de encriptação (alg) que é HS256. Na figura 21 segue-se um exemplo:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Figura 21 – Exemplo de cabeçalho do *token*.

Por sua vez, o **conteúdo** (ou *payload*) contém os atributos úteis que normalmente são utilizados por protocolos de seguranças nas APIs (mas não são obrigatórios), os atributos de utilização do JWT e informações úteis para a aplicação e ainda atributos para partilhar informações entre aplicações. O cabeçalho e o conteúdo depois são codificados utilizando o *Base64Url*. Na figura 22 segue-se um exemplo do conteúdo:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Figura 22 - Exemplo de conteúdo do *token*.

Para criar a **assinatura** (ou *signature*) é realizada a soma do cabeçalho e do conteúdo e depois codificado com o algoritmo HMAC SHA256. Na figura 23 segue-se um exemplo:

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Figura 23 - Exemplo de assinatura do *token*.

Depois por fim, o JWT é gerado com a junção do cabeçalho, conteúdo e assinatura separados por pontos. Na figura 24 segue-se um exemplo final do *token*.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gR
G9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab3
0RMHrHDcEfxjoYZgeFONFh7HgQ
```

Figura 24 - Exemplo final do *token*.

O *token* final é obrigatório para aceder qualquer recurso privado. O *token* deve ser enviado através do ***Authorization*** no cabeçalho do pedido utilizando a *flag* ***Bearer***, como está representado na figura 25.

```
Authorization: Bearer <token>
```

Figura 25 - Estrutura do *token* no acesso dos recursos privados.

6.2.4 Verbos HTTP

Os verbos HTTP são importantes para o funcionamento de qualquer API, pois é através destes que o servidor sabe o que fazer em cada solicitação de um recurso do servidor. No caso da API do novo sistema, apesar de existirem vários verbos HTTP, apenas são utilizados os seguintes:

- **GET:** É utilizado para obter informação no servidor.
- **POST:** É utilizado para criar ou adicionar informação no servidor.
- **PUT:** É utilizado para criar ou atualizar informação no servidor.
- **DELETE:** É utilizado para remover informação no servidor.

6.2.5 Códigos de Estado de Resposta

Também não menos importantes que os verbos HTTP, os códigos de estado de respostas são a maneira de informar ao cliente o resultado da solicitação de um recurso do servidor. Existem vários códigos de estados e estão agrupados por 5 grupos, tais como:

- Códigos informativos (tem 1 no primeiro dígito).
- Códigos de sucesso (tem 2 no primeiro dígito).
- Códigos de redireccionamento (tem 3 no primeiro dígito).
- Códigos de erro do cliente (tem 4 no primeiro dígito).
- Códigos de erro do servidor (tem 5 no primeiro dígito).

Apesar de existir uma grande variedade de códigos de estado, depende de API para API a utilização dos mesmos. De acordo com as necessidades do novo sistema e seguindo o padrão IEFT [66], está descrito na tabela 15 os códigos de estado que a API *RESTful* retorna nos resultados de acesso aos recursos do servidor.

Tabela 15 - Descrição dos códigos de estado da API.

Código	Descrição
200	Indica que a solicitação ao recurso foi bem-sucedida.
201	Tal como o código 200, também indica que a solicitação ao recurso foi bem-sucedida, mas que foi criada nova informação no servidor.
400	Indica que não pode processar o recurso devido a sintaxe incorreta do pedido.
401	Indica que a solicitação não foi processada devido a autenticação inválida.
404	O recurso solicitado não foi encontrado, mas pode ser disponibilizado novamente no futuro.
405	Indica que o verbo HTTP utilizado não está disponível para o *endpoint* solicitado.
500	Indica que ocorreu um erro do servidor ao processar o recurso solicitado.

6.2.6 Representação

Em termos de representação dos resultados da API *RESTful* do novo sistema, apenas é utilizado o formato JSON (*JavaScript Object Notation*). O formato JSON cada vez mais é utilizado pelas APIs, visto que é um formato mais fácil de analisar, de ler e é compatível com a maioria das linguagens de programação.

A estrutura JSON retornada pela API RESTful nas suas respostas tem como base as chaves *success*, *response*, *message* e *errors*. Cada uma dessas chaves tem o seguinte valor/significado:

- ***Success***: Tem *true* se a solicitação do recurso foi bem-sucedida e *false* caso contrário.
- ***Response***: Tem os dados retornados pelo servidor quando a solicitação é bem-sucedida.
- ***Message***: Possui a mensagem de sucesso da solicitação bem-sucedida.
- ***Errors***: Indica um ou mais erros quando as solicitações não são bem-sucedidas.

Na figura 26 segue-se o exemplo da estrutura JSON de uma solicitação bem-sucedida ao servidor e na figura 27 uma solicitação em que ocorreu erro de sintaxe.

```
HTTP 201 Created - Ex: api/v1/pt/logo
{
    "success": true,
    "message": {
        "code": "e081",
        "msg": "Logotipo da entidade guardado com sucesso."
    },
    "response": {
        "data": {
            "id": "599d90e4689bf",
            "name": "Alexandra Yasmin Reis de Pereira",
            "file": "http://localhost:8000/file/bGGl5GCCVfKHmTjQ4ARyXfQAzeY3UwZVt5elvSDI44CQMUPrzp"
        }
    }
}
```

Figura 26 - Resposta JSON de sucesso ao inserir logótipo na entidade.

```
HTTP 400 Bad Request
{
    "success": false,
    "errors": {
        "name": {
            "code": "e048",
            "msg": "O nome é obrigatório."
        },
        "email": {
            "code": "e024",
            "msg": "O e-mail deve ser um endereço de e-mail válido."
        }
    }
}
```

Figura 27 - Resposta JSON de erro ao criar novo administrador.

6.2.7 Estrutura de Ficheiros

A estrutura de uma aplicação padrão do Laravel 5.4, fornece uma organização sem quaisquer restrições de ficheiros para os seus desenvolvedores. Apesar de disponibilizar uma estrutura um pouco diferente das outras *frameworks* MVC (*Model-View-Controller*), é fácil de organizar e compreender.

Na figura 28 segue-se a estrutura de ficheiros da API utilizando a *framework* Laravel 5.4.

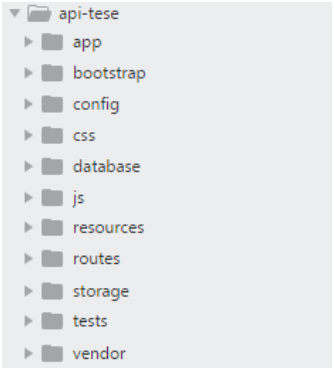


Figura 28 - Estrutura de ficheiros da API.

As pastas da estrutura de ficheiros da API têm o seguinte conteúdo:

- ***App***: Contém o código principal da API.
- ***Bootstrap***: Contém os ficheiros de inicialização e configuração do *autoloading*.
- ***Config***: Contém os ficheiros de configuração da API.
- **CSS**: Contém os ficheiros css.
- ***Database***: Contém os *migrations* e os *seeds* da base de dados.
- **JS**: Contém os ficheiros *javascript*.
- ***Resources***: Contém os ficheiros dos idiomas, as vistas e ficheiros LESS ou SASS.
- ***Routes***: Contém os ficheiros com as definições das rotas da API.
- ***Storage***: Contém o ficheiro de *log*, os ficheiros de *caches* e os ficheiros gerados pela API.
- ***Tests***: Contém os testes HTTP da API.
- ***Vendor***: Contém as dependências do *composer*.

A pasta ***app*** é a pasta mais importante na estrutura de ficheiros da API, pois como foi referido anteriormente contém todo o código principal da API. Na figura 29 apresenta-se a estrutura de ficheiros da pasta ***app***.

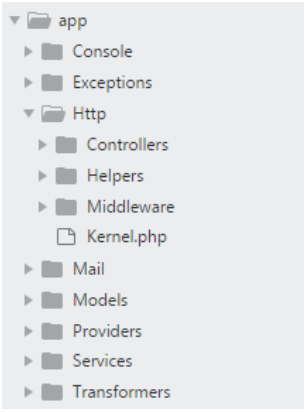


Figura 29 – Estrutura de ficheiros da pasta *app*.

Dentro da pasta ***app***, os ficheiros mais importantes para o desenvolvimento da API RESTful são os *Controllers*, *Middleware*, *Mail*, *Models*, *Services* e *Transformers*, que passa-se a explicar de seguida:

- ***Controllers***: Os controladores agrupam a lógica de gestão de solicitações, ou seja, são responsáveis por receber, processar e enviar resposta aos pedidos dos utilizadores.
- ***Middleware***: Contém os mecanismos necessários para filtrar os pedidos HTTP efetuado na API.
- ***Mail***: Contém a lógica dos *e-mails* enviados pela API.
- ***Models***: Os modelos interagem com a base de dados e para cada tabela da base de dados existe um modelo. Nos modelos também são definidas as relações entre as tabelas.
- ***Services***: Os serviços contêm toda a lógica do sistema e fazem a ligação entre os controladores e os modelos.
- ***Transformers***: São os responsáveis por formatar e organizar os JSON de resposta de cada pedido da API.

6.2.8 Documentação

Qualquer utilizador que pretende utilizar uma API depara-se imediatamente com os problemas de como utilizar, tipos de erros, tipos de respostas, formatação das respostas, etc. Logo uma documentação correta e completa de uma API é importante para a sua utilização, pois contém instruções de como utilizar e integrar com a API.

Para gerar a documentação da API *RESTful* do novo sistema utilizou-se a APIDOC [68], que permite criar documentação através de anotações no código-fonte específicas para cada linguagem de programação. A documentação da API foi criada ao longo do desenvolvimento de cada *endpoint* e na figura 30 encontram as anotações para criar a documentação do *endpoint* GET para obter os dados de autenticação da entidade no sistema.

```
/**
 * @api {get} v1/{lang}/authentication Authentication
 * @apiVersion 0.0.0
 * @apiName GetActiveAuthentication
 * @apiGroup Apps
 *
 * @apiPermission Apps
 *
 * @apiHeader {String} Authorization Token
 * @apiHeader {Integer} nif          NIF entity
 *
 * @apiParam (Parameter-URL) {String = "en", "pt"} lang Language
 *
 * @apiSuccessExample Success-Response:
 *   HTTP 200 OK - Ex: api/v1/pt/authentication
 *   {
 *       "success": true,
 *       "response": {
 *           "data": {
 *               "id": "599ee84e992f7",
 *               "date_strat": "2017-08-24 15:53:02",
 *               "settings": [
 *                   {
 *                       "id": "599d90e45fe39",
 *                       "alias": "send_docs_auto",
 *                       "description": "Enviar os documentos automaticamente para os clientes.",
 *                       "active": 0
 *                   }
 *               ]
 *           }
 *       }
 *   }
```

Figura 30 - Anotações para criar documentação da API.

Depois de criadas as anotações, utilizou-se o código "*apidoc -i myapp/ -o apidoc/ -t mytemplate/*" para gerar a documentação da API. A documentação da API *RESTful* do novo sistema de faturas eletrónicas está disponível em https://igest.acin.pt/api-tese/apidoc/.

6.3 Testes e Resultados

Ainda durante a fase de implementação, iniciou-se a etapa de testes. A etapa de testes é importante para a implementação funcional e robusta da API do sistema, pois é nesta etapa que é possível descobrir erros que não foram descobertos durante a etapa de desenvolvimento da API.

6.3.1 Testes HTTP

Para testar a API do sistema utilizou-se os testes HTTP da *framework* PHPUnit [67]. Esta *framework* é uma estrutura de testes unitários para a linguagem PHP e encontra-se integrada com o Laravel 5.4 [53].

Decide-se testar a API do sistema utilizando testes HTTP, dado que estes permitem realizar solicitações HTTP para uma API e depois verificar os respetivos resultados. Quanto ao

resultado é possível verificar o código de estado fornecido, o cabeçalho da resposta e o conteúdo e estrutura do JSON. Ainda é possível realizar solicitações HTTP utilizando os verbos HTTP.

Foi criado um total de 85 testes que equivale a um teste por cada pedido da API desenvolvido, e em alguns casos mais que um teste, com o objetivo de testar todos os pedidos da API e os seus casos possíveis. A *framework* PHPUnit permite gerar um documento em XML com os resultados dos testes realizados, que por sua vez converte-se para HTML para uma melhor interpretação desses resultados. Na tabela 16 segue-se as validações e os resultados dos testes realizados.

Tabela 16 - Testes HTTP da API.

Teste HTTP	Validações	Resultado
AdministrationAdminsTest		
testPOSTAdminsCreate	18	OK (1.937s)
testDELETEAdminsDelete	13	OK (1.532s)
testPUTAdminsUpdate	15	OK (1.235s)
testGETAllAdmins	23	OK (1.440s)
AdministrationContractsTest		
testGETAllContractRequests	20	OK (1.118s)
testPOSTNewContracts	24	OK (0.918s)
testGETContractsCompleted	8	OK (0.667s)
testGETContractsToFinish	10	OK (0.858s)
testGETEntitiesContracts	10	OK (0.750s)
testPOSTPayContractRequests	19	OK (1.128s)
AdministrationEntityTest		
testGETAllEntities	25	OK (1.611s)
testPUTEntitiesUpdate	15	OK (1.017s)
testGETUsersEntities	10	OK (0.750s)
AdministrationUsersTest		
testPOSTUsersCreate	18	OK (1.492s)
testDELETEUsersDelete	13	OK (0.831s)
testPUTUsersUpdate	15	OK (0.957s)
testGETAllAdmins	23	OK (1.413s)
testGETEntitiesUsers	12	OK (1.439s)
AppsTest		
testGETAuthentication	13	OK (0.620s)
testPOSTAppAuthentication	17	OK (0.747s)
testPUTAuthenticationUpdate	19	OK (0.787s)
testDELETEAuthenticationDelete	14	OK (0.584s)
ConnectionsTest		
testPOSTCreateConnections	18	OK (1.091s)
testPOSTAcceptConnections	16	OK (0.976s)
testDELETERejectConnections	16	OK (0.983s)
testDELETECancelConnectionRequest	16	OK (0.987s)
testDELETECancelConnections	16	OK (0.962s)
testGETNewConnections	21	OK (1.190s)
testGETActiveConnections	21	OK (1.102s)

testGETReceivedConnections	21	OK (1.057s)
testGETSentConnections	21	OK (1.114s)
ContractTest		
testPOSTNewContractRequests	20	OK (1.003s)
testPOSTCancelContractRequests	22	OK (1.027s)
testGETAllContractRequests	16	OK (1.072s)
DocumentsIssuedTest		
testPOSTSendDocuments	32	OK (1.283s)
testPOSTImportDocuments	33	OK (1.351s)
testPOSTSendIssuedDocuments	22	OK (1.516s)
testPOSTCancelSendingIssuedDocuments	18	OK (2.122s)
testGETDocumentsIssuedToSend	31	OK (2.038s)
testGETDocumentsIssuedSent	31	OK (1.953s)
testGETDocumentsIssuedRejected	31	OK (1.969s)
testGETDocumentsIssuedAccepted	31	OK (1.949s)
DocumentsReceivedTest		
testPOSTAcceptReceivedDocuments	18	OK (1.122s)
testPOSTRejectReceivedDocuments	18	OK (1.144s)
testGETDocumentsReceived	31	OK (1.960s)
testGETDocumentsReceivedRejected	31	OK (1.992s)
testGETDocumentsReceivedAccepted	31	OK (2.057s)
EntityTest		
testPOSTEntityCreate	18	OK (1.404s)
testPOSTEntityCreateWithEntityInvitation	5	OK (0.877s)
testPOSTEntityUsersCreate	24	OK (1.498s)
testPUTEntityUsersUpdate	22	OK (1.669s)
testDELETEEntityUsersDelete	24	OK (2.128s)
testGETEntityUsers	15	OK (1.261s)
testPOSTToReport	13	OK (0.713s)
EventTest		
testGETDocumentIssuedEvents	20	OK (1.749s)
testGETDocumentReceivedEvents	20	OK (1.737s)
testGETUsersEvents	16	OK (1.339s)
testGETEntitiesEvents	16	OK (1.140s)
testGETConnectionsEvents	16	OK (1.445s)
testGETAdministratorsEvents	12	OK (1.385s)
FileTest		
testPOSTLogoUpload	19	OK (1.182s)
testLogoDelete	17	OK (1.201s)
testPOSTPhotoUpload	15	OK (1.378s)
testPhotoDelete	13	OK (2.051s)
InvitationTest		
testPOSTSendInvitations	23	OK (1.355s)
testGETEntityInvitations	17	OK (2.315s)
PersonTest		
testGETUserEntities	25	OK (2.168s)
testPUTUserEntitiesUpdate	21	OK (1.854s)
testPUTProfileUpdate	20	OK (1.561s)

testGETProfile	9	OK (0.964s)
testPOSTRecoverPassword	12	OK (1.071s)
testPOSTLogin	19	OK (1.609s)
ServiceTest		
testGETPeriods	9	OK (0.442s)
testGETModulesMonthly	9	OK (0.559s)
testGETModulesSemester	9	OK (0.563s)
testGETModulesYearly	9	OK (0.538s)
testGETModulesUndefined	11	OK (0.594s)
StatesTest		
testGETContractRequestStates	8	OK (0.360s)
testGETDocumentStates	8	OK (0.436s)
testGETPersonStates	8	OK (0.417s)
TypesTest		
testGETDocumentTypes	8	OK (0.165s)
testGETEntityTypes	8	OK (0.220s)
testGETFactDocumentTypes	8	OK (0.162s)
testGETModulesTypes	8	OK (0.157s)
testGETPersonTypes	8	OK (0.148s)
TOTAL:	**1478**	**OK (100.707s)**

A realização dos testes HTTP tornou-se importante para o desenvolvimento e manutenção da API, pois permite voltar a testar os pedidos da API em qualquer momento e assim assegura sempre o bom funcionamento dos mesmos.

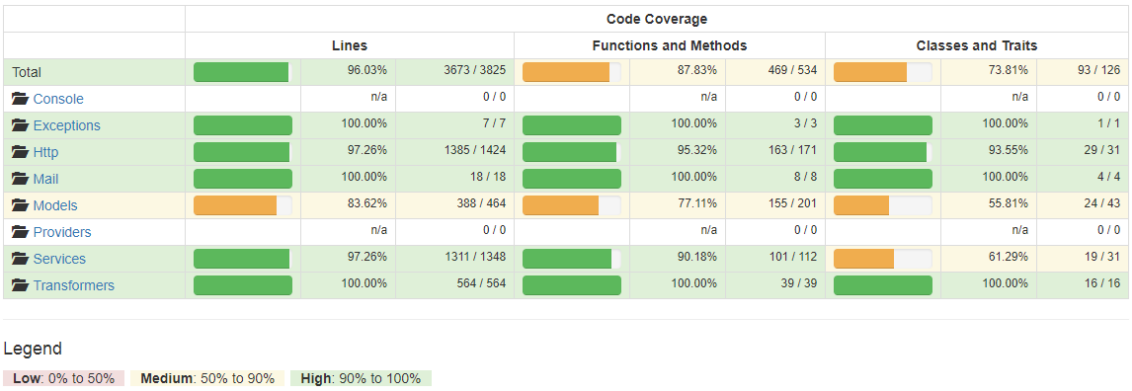
6.3.2 Resultados de avaliação dos testes HTTP

A realização dos testes HTTP, importante como já foi referido anteriormente, mas por vezes isso não é suficiente. Os testes não são bem-sucedidos apenas de acordo com o volume de testes e o número de validações, mas também pela quantidade de código que os testes abrangem. Se os testes garantirem que abrangem a maioria do código desenvolvido, haverá menos probabilidade de conter erros na API e assim é possível afirmar que os testes foram realizados com sucesso.

Para avaliar os testes HTTP realizados na API, utilizou-se o *code coverage* do PHPUnit. Ao adicionar na execução dos testes HTTP o código *"--coverage-html <file>"*, gerou-se o relatório em HTML da cobertura de código dos testes HTTP da API. No relatório gerado, é possível analisar métricas de *software* de cobertura de código do tipo:

- **Por linha:** Verifica se cada linha executável do código foi executada.
- **Por métodos e funções:** Verifica se cada método e função foram executados, mas só é considerado um método ou função coberto se todas as suas linhas são cobertas.
- **Por classes e *trait*:** Verifica se cada *classe* ou *trait* é coberta, mas só é considerado uma *classe* ou *trait* coberta se todos os seus métodos e funções são cobertos.

Na figura 31 encontram os resultados de cobertura de código dos testes HTTP com as métricas de cobertura de código descritas anteriormente e este está disponível em https://igest.acin.pt/api-tese/report_tests/index.html.



	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		96.03%	3673 / 3825		87.83%	469 / 534		73.81%	93 / 126
Console		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Exceptions		100.00%	7 / 7		100.00%	3 / 3		100.00%	1 / 1
Http		97.26%	1385 / 1424		95.32%	163 / 171		93.55%	29 / 31
Mail		100.00%	18 / 18		100.00%	8 / 8		100.00%	4 / 4
Models		83.62%	388 / 464		77.11%	155 / 201		55.81%	24 / 43
Providers		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Services		97.26%	1311 / 1348		90.18%	101 / 112		61.29%	19 / 31
Transformers		100.00%	564 / 564		100.00%	39 / 39		100.00%	16 / 16

Legend

Low: 0% to 50% **Medium**: 50% to 90% **High**: 90% to 100%

Figura 31- Avaliação dos testes HTTP utilizando o Code Coverage.

De acordo com os resultados do *code coverage*, **96.03% das linhas executáveis**, **87.83% dos métodos e funções** e **73.81% das *classes* e *traits*** da API são cobertas pelos testes HTTP. Ainda é importante referir, que grande parte do código da API (de acordo com a secção 6.2.7), em termos de linhas executáveis os *middlewares*, *transformers* e *mails* são cobertos 100%, os *services* 97.26% e os *controllers* 97.10%.

Além das métricas de cobertura de código, também é possível analisar de acordo com os testes a complexidade, risco e a cobertura de código insuficiente das *classes* ou métodos, entre outros parâmetros que estão disponíveis em https://igest.acin.pt/api-tese/report_tests/dashboard.html.

6.4 Conclusão

Neste capítulo foram apresentados os aspetos mais importantes da implementação da API *RESTful* do sistema, explicando detalhadamente o seu funcionamento e a arquitetura. Começou-se por explicar a arquitetura da API desde a solicitação de um pedido até a sua resposta. A API *RESTful* do sistema de faturas eletrónicas está disponível em https://igest.acin.pt/api-tese/api e o código correspondente está disponível no Github em https://github.com/Oxyde13/api-tese.

Quanto aos testes HTTP realizados, revelaram-se fulcrais para descobrir erros durante o desenvolvimento da API e no futuro também será importante para a sua manutenção, dado que permite testar cada um dos pedidos existentes. Deste modo podemos concluir que os testes HTTP foram realizados com sucesso, pois no geral cobrem 96.03% das linhas executáveis do código desenvolvido.

Assim sendo, é possível seguir para a etapa de implementação do *front-end* do sistema de faturas eletrónicas, com garantias do funcionamento correto e robusto da API *RESTful*.

7 Implementação do *Front-End*

7.1 Introdução

Depois da implementação da API *RESTful* do sistema e respetivos testes, iniciou-se o desenvolvimento visual do sistema de faturas eletrónicas. Ao longo deste capítulo será apresentado uma breve descrição sobre a *framework* Angular e o processo de desenvolvimento do *font-end* do sistema. É importante referir que a utilização da *framework* Angular é um requisito não funcional (apresentado na secção 3.4) e que toda a informação apresentada sobre esta *framework* neste capítulo é referente a versão 4.

Também neste capítulo, será descrito o funcionamento da integração do novo sistema de faturas eletrónicas com um sistema de faturação, neste caso o iGEST. Para o novo sistema, a integração com outros sistemas de faturação é fulcral para a sua sobrevivência pois deste modo permite receber faturas e depois entregar ao respetivo destinatário. Em termos de apresentação visual do sistema, tal como a API *RESTful*, foi desenvolvido de acordo com os protótipos definidos na secção 3 de desenho do sistema.

É de referir ainda que, relativamente à API *RESTful* desenvolvida anteriormente, durante o desenvolvimento do *front-end* foi necessário ajustar alguns pedidos, e em certos casos até criar novos pedidos.

7.2 Introdução ao Angular

7.2.1 O que é o Angular?

O Angular é uma *framework* JavaScript que facilita o desenvolvimento de aplicações *web*, utilizando linguagens como HTML, *JavaScript* e *TypeScript* (que é compilado para *JavaScript*). A *framework* Angular disponibiliza várias bibliotecas que facilitam o desenvolvimento das aplicações, pois foi criado especificamente para ajudar a desenvolver aplicações com uma única página *"single page applications"*. A sua estrutura permite reduzir consideravelmente a duplicação de código.

As aplicações com uma única página, funcionam dentro do *browser* e não necessitam de recarregamento da página durante a sua utilização. Essas aplicações fazem utilização dos pedidos AJAX (*Asynchronous Javascript and XML*) para obter dados do servidor sem atualizar qualquer página. Na figura 32 encontra-se a estrutura de uma aplicação com uma única página. [69]

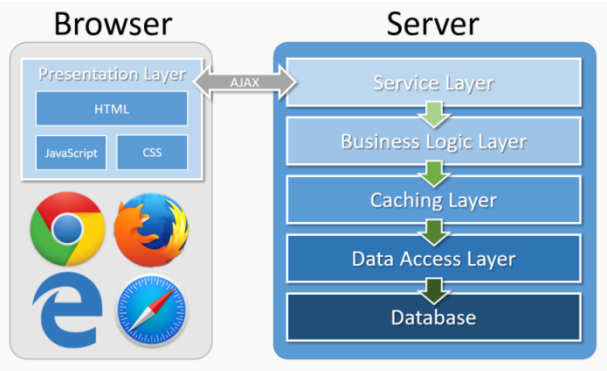


Figura 32 - Estrutura de uma aplicação com uma única página.

7.2.2 Características

Principalmente a partir da versão 2, o Angular tem como principais características e benefícios a criação de aplicações *cross-platform*, a velocidade/ desempenho, a produtividade e por fim o desenvolvimento completo das aplicações. Cada uma dessas características oferece e possibilita o seguinte:

Aplicações *cross-platform*: Possibilita o desenvolvimento de aplicações nativas utilizando *Ionic Framework*, *NativeScript* e *React Native*.

Velocidade e Desempenho: Oferece divisão automática de código, permitido apenas carregar o que é necessário e é altamente otimizado na sua geração de código.

Produtividade: Oferece ferramentas de linha de comandos (CLI Angular) para criar modelos, componentes, serviços e ainda criação de testes de forma rápida e simples. Disponibiliza IDEs (*Integrated Development Environment*) robustos que fornecem erros instantâneos e outros comentários sobre o código.

Desenvolvimento Completo: Possibilita a criação de testes unitários e testes funcionais de forma rápida e estável.

7.3 Desenvolvimento

7.3.1 Arquitetura

A arquitetura do Angular concentra-se nos principais blocos tais como módulos, componentes, *templates*, *metadata*, ligações de dados, diretivas, serviços e injeção de dependência. Na figura 33 encontra-se cada um desses blocos e como interagem.

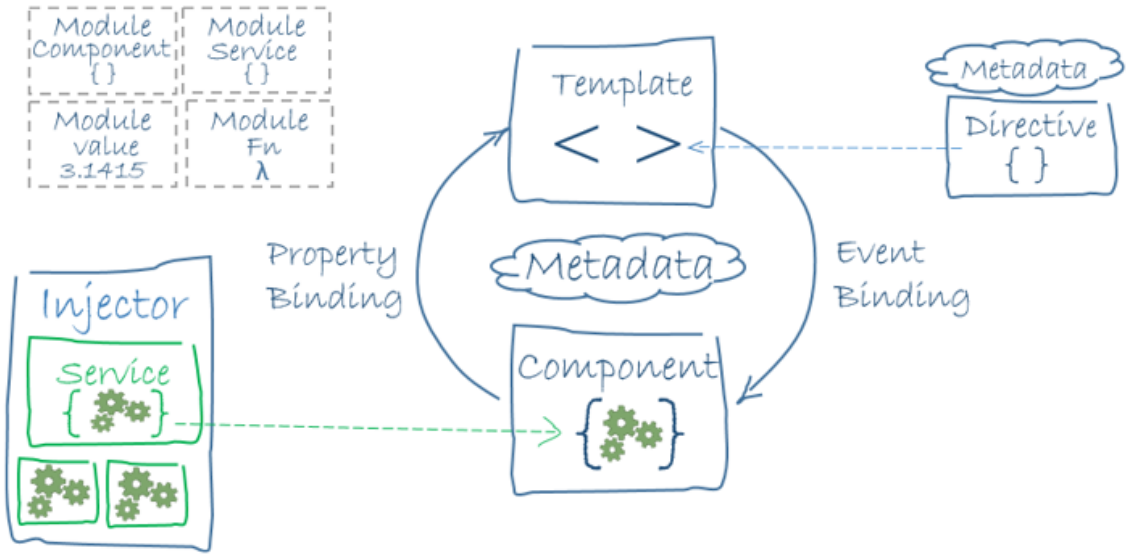


Figura 33 - Arquitetura geral do Angular. [69]

7.3.1.1 Módulos

É através dos módulos que as aplicações Angular se tornam modulares, pois com os módulos é possível separar o funcionamento ou fluxos diferentes das aplicações. Uma aplicação angular tem pelo menos um módulo, chamado de AppModule. Em qualquer módulo Angular existe um decorador *@NgModule*, que é utilizado para definir as importações, declarações e opções de inicialização desse módulo.

Em termos do sistema de faturas eletrónicas, existe mais três módulos além do módulo principal AppModule. Cada um desses módulos tem o seguinte:

- **AppModule**: Contém todo o fluxo e funcionamento relacionado com a aplicação na zona pública, tais como o registar, iniciar sessão e recuperar palavra-passe.
- **UserLayoutModule**: Contém todo o fluxo e funcionamento relacionado com a aplicação para um utilizador normal.
- **AdminLayoutModule**: Contém todo o fluxo e funcionamento relacionado com a aplicação para um administrador.
- **SharedModule**: Contém os componentes menu, cabeçalho, perfil e reportar partilhados pelos módulos UserLayoutModule e AdminLayoutModule.

Na figura 34 segue-se o módulo AppModule com as suas importações, declarações e opções de inicialização.

```
@NgModule({
    declarations: [
        AppComponent, LoginComponent,
        RecoverPasswordComponent, SignUpComponent
    ],
    imports: [
        BrowserModule, BrowserAnimationsModule,
        HttpClientModule, FormsModule,
        ReactiveFormsModule, NG2DataTableModule,
        RouterModule.forRoot([
            { path: '', redirectTo: 'login', pathMatch: 'full' },
            { path: 'login', component: LoginComponent },
            { path: 'recover-password', component: RecoverPasswordComponent },
            { path: 'sign-up', component: SignUpComponent },
            { path: 'index', loadChildren: './user-layout/user-layout.module#UserLayoutModule'},
            { path: 'admin', loadChildren: './admin/admin.module#AdminModule'},
            { path: '**', redirectTo: 'login'}
        ], { useHash: true }),
        ToastModule.forRoot(),
        LoadingModule.forRoot({
            backdropBackgroundColour: 'rgba(0,0,0,0.1)',
            backdropBorderRadius: '0px',
            primaryColour: '#ffffff',
            secondaryColour: '#ffffff',
            tertiaryColour: '#ffffff'
        }),
        MyDatePickerModule, SelectModule, CurrencyMaskModule
    ],
    providers: [
        {provide: ToastOptions, useClass: CustomOption},
        ConfigurationsService, EntitiesUserService, AuthenticationService,
        AuthGuard, CountriesService, EntityService, UsersService,
        StatesService,TypesService, InvitationsService,
        DocumentsIssuedService, ConnectionsService, PaginationService,
        PersonService, DocumentsReceivedService, StatisticService
    ],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

Figura 34 - Módulo AppModule

7.3.1.2 Componentes

Os componentes são pequenas partes das aplicações Angular, permitindo assim reduzir a repetição de código no desenvolvimento das aplicações. Um componente deve pertencer a um módulo, para que esse componente possa utilizar outros componentes.

Tal como os módulos, os componentes também têm um decorador *@Component* que permite definir uma *classe* como um componente e fornece *metadata* adicionais que determinam como os componentes devem ser processados.

Quanto ao sistema, foi criado dezoito componentes no total de modo a evitar repetição de código. Os componentes foram criados através do CLI Angular, e associado a cada componente existe um ficheiro HTML (*template*), SCSS e um *TypeScript*. Na figura 35 segue-se o ficheiro *TypeScript* do componente *login* com a declaração do decorador *@Component* e da respetiva *classe*.

```typescript
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {
    formLogin: FormGroup;

    constructor(
        private toastr: ToastsManager,
        private vcr: ViewContainerRef,
        private http: HttpClient,
        private config: ConfigurationsService,
        private auth: AuthenticationService,
        private route: Router
    ) {
        this.auth.validateSession();
        this.toastr.setRootViewContainerRef(vcr);
    }

    ngOnInit() {
        this.formLogin = new FormGroup({...
        });
    }

    onLoggedin() {
        this.config.showLoading();

        if (this.formLogin.valid == false) {
            this.validateFormLogin();
            this.config.hideLoading();
            return;
        }

        this.auth.login(this.formLogin.value).subscribe(...
        );
    }

    validateFormLogin() {
        let email    = this.formLogin.controls.email.errors,
            password = this.formLogin.controls.password.errors

        if (email && email.required) {
            this.toastr.error('O e-mail é obrigatório.');
            return;
        }
```

Figura 35 - Código *TypeScript* do componente *login*.

7.3.1.3 Templates

É nos *templates* que a visão dos componentes é definida. O *template* é formado em HTML regular, apenas adicionando algumas ligações de dados e/ou diretivas específicas do Angular (mais a frente nesta secção será descrito cada um destes).

Na figura 36 encontra-se um *template* (entre vários existentes na aplicação) do componente de recuperar palavra-passe.

```html
<div class="container">
    <div class="row">
        <div class="col-sm-12">
            <div class="form-container-public">
                <form (submit)="onRecoverPassword()" [formGroup]="formRecoverPass" class="form-normal">
                    <p class="title">Recuperar Palavra-Passe</p>
                    <div class="form-group inner-addon left-addon">
                        <i class="glyphicon glyphicon-user"></i>
                        <input type="email" formControlName="email" placeholder="E-mail">
                    </div>
                    <div class="form-group">
                        <a routerLink="/login">Voltar?</a>
                    </div>
                    <div class="form-group">
                        <button type="submit">Recuperar</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
```

Figura 36 - *Template* HTML do componente recuperar palavra-passe.

7.3.1.4 Metadata

Os *metadata* são responsáveis por definir como o Angular deve processar uma *classe*. Um componente ou um módulo é apenas uma *classe*, que passa a ser componente ou módulo

quando é adicionado *metadata* utilizando o decorador *@Component* ou *@NgModule* respetivamente.

Para o desenvolvimento do sistema de faturas eletrónicas foi utilizado *metadata* para definir os módulos e os componentes. Na figura 37 segue-se um exemplo de *metadata* para cada um dos casos referidos anteriormente.

```
@Component({
  selector: 'app-profile',
  templateUrl: './profile.component.html',
  styleUrls: ['./profile.component.scss']
})
export class ProfileComponent implements OnInit { }
```

```
@NgModule({
    imports: [
        CommonModule, FormsModule,
        FormsModule, ReactiveFormsModule,
        MyDatePickerModule, SelectModule,
    ],
    declarations: [
        MenuComponent, HeaderComponent,
        ProfileComponent, ReportComponent
    ],
    exports: [
        MenuComponent, HeaderComponent,
        ProfileComponent, ReportComponent
    ]
})
export class SharedModule { }
```

Figura 37 - Definição do componente e do módulo utilizando *metadata*.

7.3.1.5 Ligação de Dados

A ligação de dados é um mecanismo suportado pelo Angular para interligar partes do *template* HTML com partes de um componente. Para tal interligação, é adicionado marcas de ligação no *template* HTML. As ligações podem ser apenas de uma direção ou em ambas direções entre o *template* HTML e o componente.

Para qualquer aplicação desenvolvida em Angular são indispensáveis as ligações de dados entre o *template* HTML e o componente, por isso no novo sistema não foi diferente. Na figura 38 encontra-se um bloco de código do *template* HTML com várias marcas de ligação de dados.

```
<ul class="timeline" *ngIf="activities.length > 0">
    <li *ngFor="let activity of activities; let i = index; let odd = odd" [ngClass]="{'timeline-inverted' : odd}">
        <div class="timeline-badge" [ngClass]="getColorIcon(activity)">
            <i class="glyphicon" [ngClass]="getClassIcon(activity)"></i>
        </div>
        <div class="timeline-panel">
            <div class="timeline-heading">
                <p>
                    <small class="text-muted">
                        <i class="glyphicon glyphicon-time"></i> {{ activity.date }} - {{ activity.hour }}
                    </small>
                </p>
            </div>
            <div class="timeline-body">
                <p>{{ activity.event }}</p>
            </div>
        </div>
    </li>
</ul>
```

Figura 38 - Bloco de código do *template* HTML do componente de atividades.

7.3.1.6 Diretivas

No Angular existe três tipos de diretivas, tais como diretivas de componente, estruturais e de atributos. Cada uma dessas diretivas significa o seguinte:

- **Diretivas de Componente**: diretivas com um *template* HTML.
- **Diretivas Estruturais**: diretivas que alteram o *layout* adicionando ou removendo elementos no DOM. Exemplo: *ngFor e *ngIf
- **Diretivas de Atributos**: diretivas que alteram a aparência ou o comportamento de um elemento ou componente.

De acordo com nas necessidades do sistema, apenas foi utilizado diretivas de componente e estruturais. Na figura 39 encontra-se código HTML com diretivas de componente (por exemplo <app-menu></app-menu>) e diretivas estruturais (por exemplo *ngIf).

```
<ngx-loading [show]="config.loading" ></ngx-loading>
<div id="page-wrapper" [ngClass]="{'open': config.toggle}" [ngStyle]="!config.showMenu && {'padding-left': '0px'}" >
    <div id="sidebar-wrapper" *ngIf="config.showMenu" >
        <app-menu></app-menu>
    </div>

    <div id="content-wrapper">
        <div class="page-content">
            <app-header></app-header>
            <router-outlet></router-outlet>
        </div>
    </div>
</div>
```

Figura 39 - Código HTML com diretivas de componente e estruturais.

7.3.1.7 Serviços

Um serviço é tipicamente uma *classe* que pode englobar qualquer função, valor, ou recurso que a aplicação pode necessitar. Ou seja, tudo pode ser um serviço desde que seja bem definido.

Para o novo sistema foi criado quinze serviços, em que um desses serviços é um serviço de configuração da aplicação e os restantes são serviços de dados. O serviço de configuração contém variáveis e funções de configuração da aplicação e os serviços de dados têm funções que realizam pedidos HTTP à API *RESTful* do sistema. Na figura 40 segue-se um exemplo de um serviço.

```
import { ConfigurationsService } from './../services/configurations.service';

@Injectable()
export class EntityService {

    constructor( ...
    ) {}

    getEntitiyData(){
        let entity  = localStorage.getItem('entity'),
            headers = new HttpHeaders({ 'Authorization': localStorage.getItem('token')});

        return this.http.get(
            this.config.baseURL + 'users/entities/' + entity + '?fillCollections=district,country',
            {headers: headers}
        ).map(
            data => {
                return data['response'];
            }
        );
    }
```

Figura 40 - Serviço com os pedidos relacionados com a entidade.

7.3.1.8 Injeção de Dependência

A injeção de dependência é fundamental no desenvolvimento de aplicações em Angular, pois permite injetar dependências em diferentes componentes sem estes necessitarem de saber como essas dependências foram criadas. As dependências são injetadas no construtor e grande parte das dependências injetadas nos componentes são serviços.

Na figura 41 segue-se apenas um exemplo de tantos outros existentes no novo sistema de faturas eletrónicas.

```
@Component({
  selector: 'app-sign-up',
  templateUrl: './sign-up.component.html',
  styleUrls: ['./sign-up.component.scss']
})
export class SignUpComponent implements OnInit {

    constructor(
        private countriesService: CountriesService,
        private config: ConfigurationsService,
        private entityService: EntityService,
        private vcr: ViewContainerRef,
        private toastr: ToastsManager,
        private auth: AuthenticationService
    ) {
        this.auth.validateSession();
        this.config.showLoading();
        this.toastr.setRootViewContainerRef(vcr);
    }
```

Figura 41 - Exemplo de injeção de dependência no componente registar.

7.3.2 Estrutura de Ficheiros

A estrutura de uma aplicação Angular (igual ou superior à versão 2), não é uma estrutura comum de acordo com as maiorias das *frameworks,* mas é uma boa base para experiências rápidas. Para ter uma estrutura correta e de acordo com o Angular convém criar o projeto e seus ficheiros com o CLI Angular. O CLI Angular é uma interface de linha de comandos que facilita a criação de aplicações Angular.

Na figura 42 encontra-se a estrutura de ficheiros do novo sistema. Foi utilizado o CLI Angular para a sua criação inicial e geração de componentes, rotas, serviços, etc.

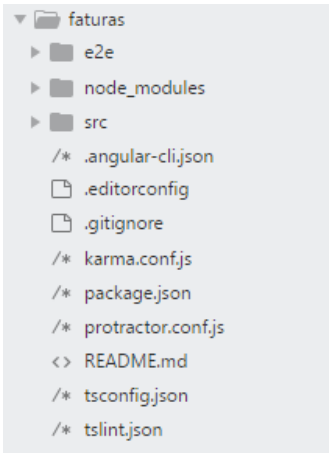


Figura 42 - Estrutura de ficheiros base de uma aplicação Angular.

As pastas e ficheiros da estrutura base das aplicações Angular têm o seguinte conteúdo:

- ***e2e***: Contém os testes unitários da aplicação.
- ***node_modules***: Contém as dependências/bibliotecas da aplicação.
- ***src***: É a pasta base da aplicação, pois contém todos os módulos, componentes, *templates*, serviços da aplicação.
- **.angular-cli.json**: Contém as configurações do CLI Angular.
- **.editorconfig**: Contém as configurações simples para o IDE.
- **.gitignore**: Contém os ficheiros que devem ser ignorados pelo Git.
- **karma.conf.js**: Contém as configurações dos testes unitários.
- **package.json**: Contém a lista das dependências/bibliotecas da aplicação.
- **protractor.conf.js**: Contém as configurações dos testes funcionais.
- **tsconfig.json**: Contém as configurações do compilador *TypeScript*.

- **tslint.json**: Contém configurações do Linting. O Linting mantém o código consistente.

Devido à sua importância, por conter todos os ficheiros da aplicação, na figura 43 encontra-se a estrutura de ficheiros da pasta *src*.

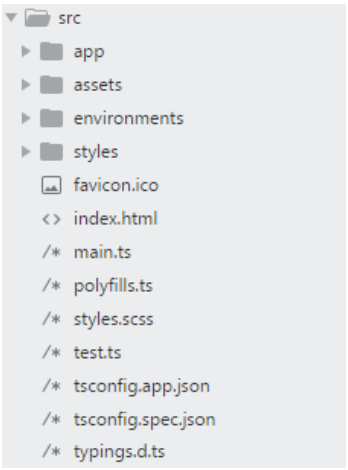


Figura 43 - Estrutura de ficheiros da pasta src.

A pasta *src* tem o seguinte conteúdo:

- ***app***: Contém todos os módulos, componentes, *templates*, serviços, etc da aplicação.
- ***assests***: Contém os ficheiros extras da aplicação, como por exemplo imagens.
- ***environments***: Contém configurações necessárias para o ambiente de produção e desenvolvimento.
- **styles**: Contém os ficheiros SCSS.
- **index.html**: Página HTML utilizada quando a aplicação é acedida.
- **main.ts**: Ponto de entrada principal da aplicação.
- **polyfills.ts**: Contém as configurações para suportar diferentes navegadores.
- **styles.css**: Contém estilos centrais da aplicação.
- **test.ts**: Ponto de entrada principal para os testes unitários.
- **tsconfig**: Contém configurações do compilador *TypeScript*.

7.4 Funcionamento

Em termos de funcionamento do novo sistema de faturas eletrónicas, decide-se explicar em detalhe com ajuda de *screenshots* a integração com o sistema de faturação iGEST. Como já foi referido nos objetivos (secção 1.3), a integração com outros sistemas é umas das características principais e mais importantes da nova solução de faturas eletrónicas.

Uma entidade que encontra-se registada no sistema iGEST e com sessão iniciada, tem nas configurações da entidade a possibilidade de integrar com vários sistemas, incluindo o sistema de faturas eletrónicas. O processo de ativação da integração do iGEST com o sistema de faturas eletrónicas é rápido, intuitivo e apenas é necessário um clique (ver figura 44).

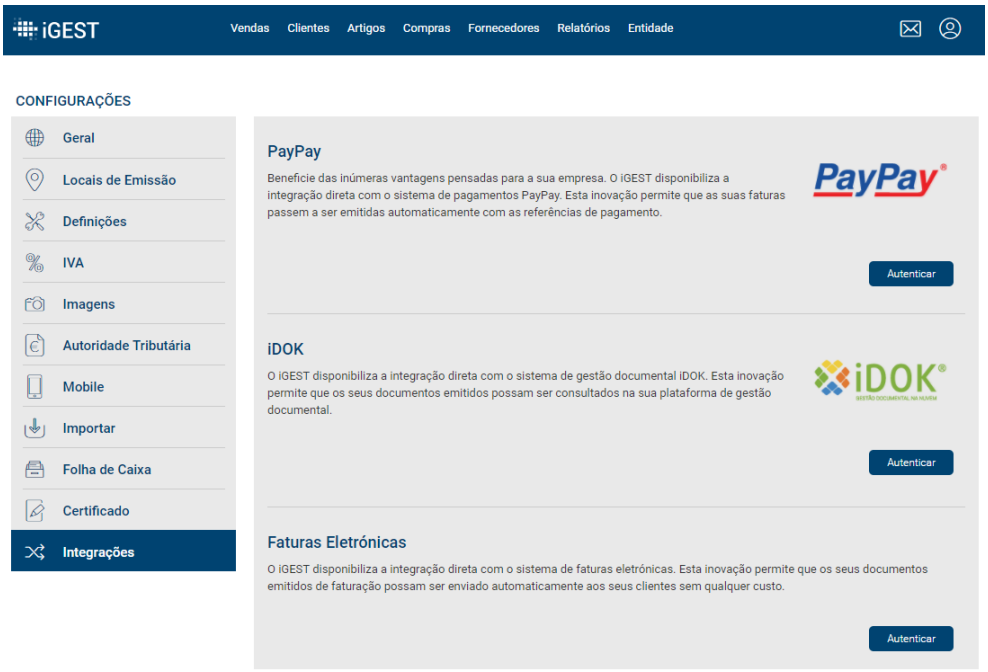


Figura 44 - Sistemas de integração com o iGEST.

Ao clicar no botão autenticar no sistema de faturas eletrónicas, será realizado o pedido para autenticação. A autenticação será realizada com sucesso, apenas caso a entidade que pretende realizar a autenticação esteja registada e tenha subscrição ativa no sistema de faturas eletrónicas. Depois da integração ativa entre o iGEST e o sistema de faturas eletrónicas, é possível ativar/desativar configurações extras (como por exemplo possibilitar o envio de documento automaticamente para o sistema) e/ou desativar a integração (ver figura 45).

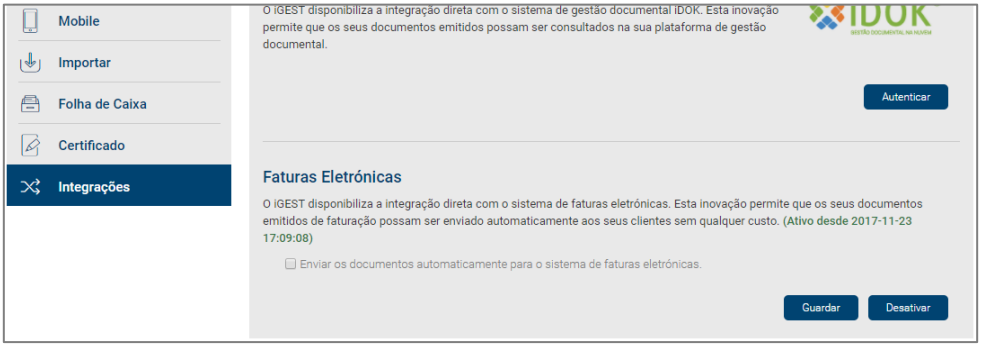


Figura 45 - Integração entre iGEST e sistema de faturas eletrónicas ativa.

Com a integração efetuada com sucesso, na emissão de documentos de faturação no iGEST é possível enviar os documentos para o sistema de faturas. Os documentos enviados para o sistema de faturas devem ser do tipo “Fatura” ou “Fatura Simplificada” e deve ter cliente associado. Caso exista a configuração ativa “Enviar os documentos automaticamente para o sistema de faturas eletrónicas”, depois da emissão do documento este será enviado automaticamente. Caso contrário, é apresentado um botão para enviar manualmente (ver figura 46).

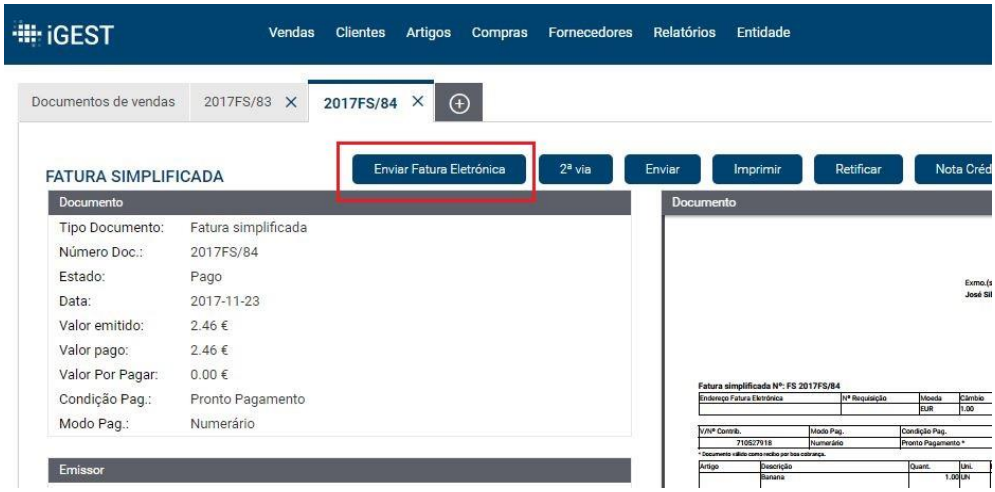


Figura 46 - Envio do documento para o sistema manualmente.

Depois do envio da fatura para o sistema, fica associado ao documento informações sobre a data de envio da fatura para o sistema e o estado de envio da fatura ao destinatário. Inicialmente quando a fatura é enviada para o sistema, esta fica no estado por enviar ao cliente (ver figura 47).

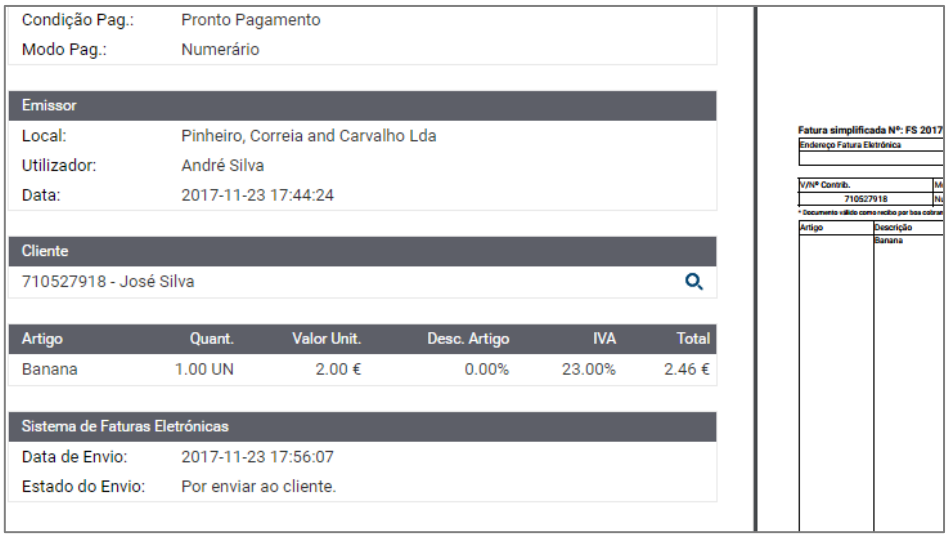


Figura 47 - Informações do documento.

Ao aceder ao sistema de fatura eletrónicas, o documento enviado encontra-se nos documentos emitidos no estado por enviar (ver figura 48). A partir desse momento, é possível enviar o documento ao cliente caso este encontrar-se registado no sistema e que exista ligação ativa entre a entidade emissora e o destinatário do documento. Se o cliente não tiver registado é possível enviar um convite para o cliente aderir ao sistema, caso contrário é possível enviar um pedido de ligação para possibilitar a troca de faturas eletrónicas.

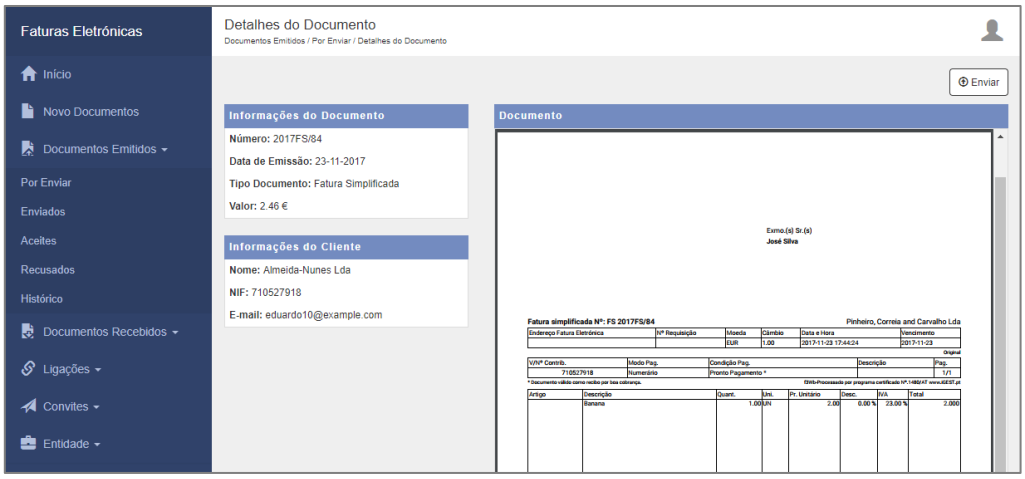


Figura 48 - Detalhes do documento no estado por enviar.

Ao clicar no botão enviar, o documento é enviado ao cliente e fica a aguardar aceitação ou rejeição do documento por parte do cliente. O documento passa para os documentos enviados e em caso de engano, é possível cancelar o envio do documento caso este ainda não tenha sido aceite pelo cliente (ver figura 49).

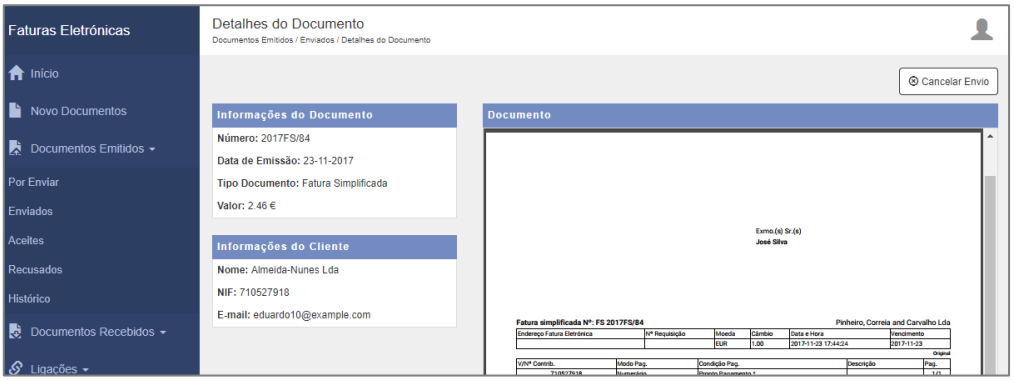


Figura 49 - Detalhes do documento no estado enviado.

O cliente destinatário do documento, ao aceder ao sistema terá um novo documento nos documentos recebidos que poderá aceitar ou rejeitar o documento (ver figura 50). Apesar do cliente destinatário ainda não ter aceitado o documento, este já tem acesso às informações gerais do documento e da respetiva fatura (documento PDF).

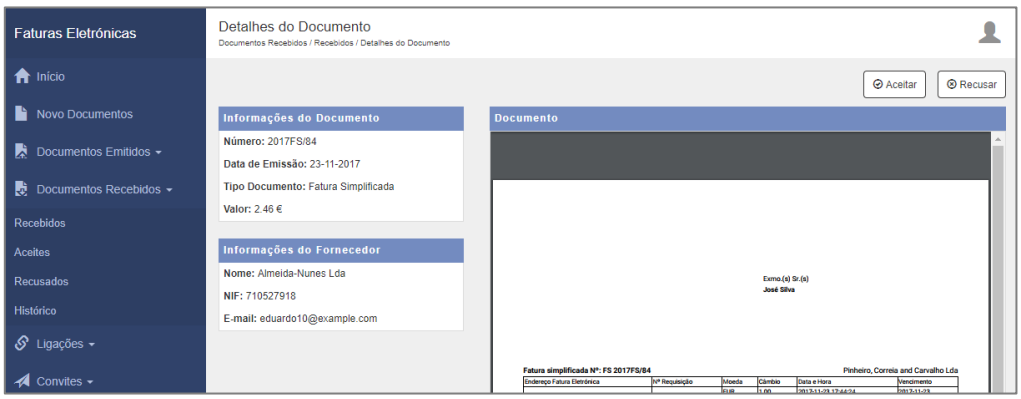


Figura 50 - Detalhe do documento no estado recebido.

Se o cliente aceitar o documento, o processo de envio da fatura fica concluído e a fatura fica entregue (ver figura 51). Caso contrário se o cliente recusar o documento, este é enviado de volta para o emissor do documento.

Figura 51 - Documento entregue ao seu destinatário.

Durante todo o processo de envio da fatura, o estado de envio no iGEST também foi atualizado. Neste caso específico como a fatura já foi aceite pelo destinatário, nos detalhes do documento no iGEST o estado de envio fica entregue ao cliente (ver figura 52).

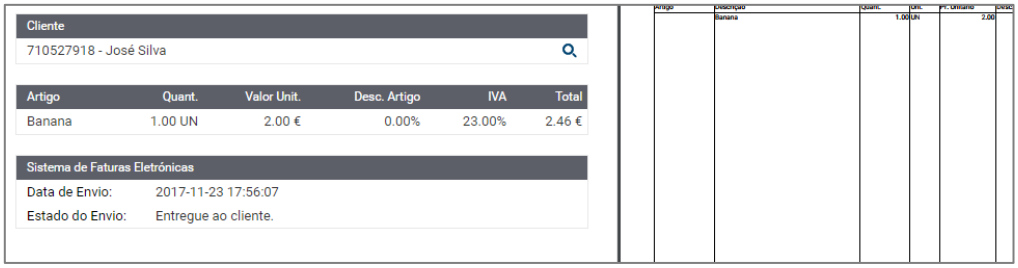


Figura 52 - Detalhes do documento no iGEST quando a fatura é entregue ao cliente.

7.5 Conclusão

Neste capítulo foi apresentado os aspetos principais da implementação do *front-end* do sistema de faturas eletrónicas, explicando como foi desenvolvido e estruturado. Inicialmente apresentou-se uma breve introdução da *framework* Angular e as suas características, seguindo-se o desenvolvimento com a explicação em detalhe da arquitetura do sistema e da estrutura de ficheiros. Podemos concluir que em termos de arquitetura é um sistema que será fácil de alterar e manter no futuro.

O sistema de faturas eletrónicas está disponível em https://igest.acin.pt/faturas/ e o seu código desenvolvido disponível em https://github.com/Oxyde13/faturas-eletronicas. É importante referir que o sistema está totalmente funcional, pois permite importar documentos, receber e enviar documentos, gerir ligações entre entidades, enviar convites, gerir utilizadores, etc. Em termos de *layout*, o sistema adapta-se à resolução do ecrã.

Por fim, apresentou-se detalhadamente o funcionamento da integração do iGEST com o sistema de faturas eletrónicas. Visto que o sistema desenvolvido é idêntico ao sistema desenhado na fase inicial do projeto e de acordo com os testes de usabilidade realizados com vários utilizadores (ver anexo D, E e F), podemos concluir que o sistema é eficiente, fácil de utilizar e de aprender.

8 Conclusões

8.1 Trabalho Efetuado

Um dos objetivos concretizados neste projeto de mestrado foi o desenvolvimento de um sistema que facilitasse a troca de faturas eletrónicas com todos os direitos legais, entre a entidade emissora e o destinatário da fatura. O sistema ficou dividido em duas partes, entre as quais *back-end* (API *RESTful*) e *front-end*. No *back-end* encontra-se os termos gerais e todas as regras de negócio do sistema, e por sua vez no *front-end* está a interface do sistema que interage diretamente com o utilizador final.

Inicialmente o desenvolvimento deste sistema começou com a pesquisa de trabalhos relacionados, para verificar o que existia atualmente no mercado de forma a desenvolver algo mais simples, intuitivo e inovador. De acordo com as pesquisas efetuadas, concluiu-se que no mercado de faturas eletrónicas, não existe muitos sistemas deste tipo. Dos poucos sistemas encontrados de faturas eletrónicas, ou eram limitados em termos de funcionalidades ou então restritos para uma entidade específica. Com a noção do que já existia, iniciou-se o desenho do sistema. O desenho do sistema foi a etapa essencial para o sucesso de desenvolvimento do sistema, pois foi realizado o levantamento dos requisitos, casos de utilização, diagramas de classes, diagramas de estados, diagrama de atividades, modelo relacional, protótipos e arquitetura geral do sistema. É importante referir que, não foram realizadas praticamente quaisquer alterações ao desenho do sistema realizado inicialmente. Ainda na fase do desenho do sistema, realizou-se também testes de usabilidade com cenários de utilização e um questionário sobre os protótipos realizados, para obter uma versão completa e robusta. Com todos estes processos referidos, conclui-se que o desenho do sistema foi realizado ao máximo pormenor.

Após o desenho do sistema, principalmente de acordo com os requisitos e arquitetura do sistema, iniciou-se a pesquisa de *frameworks* PHP a utilizar no *back-end* do sistema. Inicialmente optou-se por pesquisar as características, vantagens e desvantagens de cada *framework*. Depois das seis *frameworks* estudadas e de acordo com as necessidades do *back-end* do sistema, aplicou-se um exemplo prático de utilização com as duas melhores *frameworks*. Concluiu-se com o estudo realizado, que a *framework* Laravel era a mais adequada para o desenvolvimento do back-end do sistema. Também realizou-se o estudo das tecnologias existentes para o desenvolvimento de aplicações para Android e iOS, e concluiu-se que o Apache Cordova era o mais adequado para o desenvolvimento das aplicações móveis.

Depois dos passos anteriores, passou-se para o desenvolvimento do *back-end* (API *RESTful*) do sistema. Foram implementados cem pedidos na API e cada pedido foi implementado para dar suporte aos requisitos e os protótipos realizados no desenho do sistema. Também para cada pedido foram efetuados testes HTTP para garantir o funcionamento correto e robusto da API *RESTful*, dos quais foram executados com sucesso. Além disso, os testes HTTP abrangem 96.03% do código desenvolvido. Depois de terminar os testes e garantir que a grande maioria do código desenvolvido foi bem testado e estar a funcionar corretamente, passou-se para a implementação da interface do sistema (*front-end*). No desenvolvimento de cada funcionalidade, esta era testada pelo programador e no final ainda foi testado o funcionamento global do sistema por outras pessoas de modo a garantir um bom funcionamento e sem quaisquer erros. É importante referir que a interface do sistema de faturas eletrónicas adapta-se a resolução do ecrã (*responsive*).

Em relação ao código desenvolvido utilizou-se o sistema de controlo de versões Git e o serviço web Github que oferece outras funcionalidades extras aplicadas ao Git, permitindo a gestão eficiente de versões do código desenvolvido. O código do *back-end* e *front-end* do sistema estão disponíveis em:

- *Back-end*: https://github.com/Oxyde13/api-tese
- *Front-end*: https://github.com/Oxyde13/faturas-eletronicas

Também foram disponibilizados num servidor de testes da empresa ACIN iCloud Solutions a API *RESTful* (*back-end*) e a interface do sistema de faturas eletrónicas (*front-end*). Este estão disponíveis em:

- *Back-end*: https://igest.acin.pt/api-tese/
- *Front-end*: https://igest.acin.pt/faturas/

Após conclusão de todo o processo de implementação e de testes do sistema, é possível afirmar que o sistema encontra-se totalmente funcional para a receção, gestão e envio de faturas eletrónicas ao destinatário de forma fácil, eficiente, com baixo custo, sem qualquer impressão e envio da fatura em papel ao destinatário. Como qualquer sistema, ao longo da sua vida devem ser realizadas melhorias e o sistema de faturas eletrónicas também ficará pendente de melhorias no futuro.

8.2 Trabalho Futuro

Como foi referido no subcapítulo anterior, um sistema durante a sua existência necessitará sempre de melhorias. Sendo o sistema de faturas eletrónicas de uma área recente e em constante crescimento, como é óbvio, existem várias melhorias que podem ser implementadas tais como:

- Possibilitar o envio automático dos documentos emitidos aos destinatários quando estes já têm ligação ativa.
- Por parte do destinatário do documento, possibilitar a verificação dos documentos que ainda não foram enviados pelas entidades emissoras.
- Por parte do destinatário do documento, possibilitar o pedido de envio dos documentos que ainda não foram enviados pelas entidades emissoras.
- Possibilitar o registo automático com os sistemas integrados (por exemplo o iGEST) com o sistema de faturas eletrónicas.
- Possibilitar arquivar documentos emitidos ou recebidos.
- Possibilitar a adesão de indivíduos independentes (não apenas entidades e/ou empresas) para a receção de documento.
- Possibilitar através dos sistemas integrados, o envio de documento emitidos para o cliente destinatário.
- Possibilitar através dos sistemas integrados, o envio de convites.
- Possibilitar através dos sistemas integrados, o envio de pedidos de ligação.
- Possibilitar nas configurações dos sistemas integrados o envio automático dos documentos emitidos aos destinatários.
- Implementar aplicação móvel para Android e iOS com o Apache Cordova.

Estas apenas são algumas das funcionalidades que podem ser implementadas no futuro, mas de certeza que existe outras tantas para melhorar o sistema de futuras eletrónicas. Ainda é importante referir, que a solução desenvolvida será lançada para o mercado brevemente.

8.3 Considerações Finais

O objetivo principal deste projeto foi a criação de um sistema que permita às empresas o envio e a receção de faturas eletrónicas, de forma fácil e de baixo custo. Para isso era importante e necessário a integração com o sistema de faturação iGEST, que é responsável pela emissão das faturas. É importante referir que com um sistema deste tipo, a utilização do papel é reduzida e assim contribui para melhorar o meio ambiente.

Quanto a nível pessoal, este foi um projeto extremamente exigente pela sua complexidade em termos funcionais e legais. Inicialmente, devido à criação de uma API *RESTful* pela primeira vez, foi complexo perceber todos os conceitos iniciais mas ao longo do desenvolvimento foram se tornando cada vez mais compreensíveis. Depois, a utilização de *frameworks* como Laravel e Angular também teve as suas dificuldades que prontamente foram sendo ultrapassadas com ajuda de tutoriais e pesquisas na internet. Em geral, gostaria de salientar, que através do desenvolvimento deste projeto aumentei os meus conhecimentos a nível de programação com PHP, *JavaScript*, *TypeScritp*, HTML, SCSS e ainda no desenvolvimento de APIs.

Por fim, mas não menos importante, todas as indicações fornecidas pelos meus orientadores foram determinantes para concluir este projeto com sucesso.

9 Referências

[1] Ferreira, R. (2017). *Receitas do Google Play crescem 82%, mas a AppStore ainda lidera destacadíssima | Future Behind*. [online] Future Behind. Available at: https://www.futurebehind.com/google-play-app-store-ios-2016/ [Accessed 21 Jan. 2017].

[2] Trends, S., Loyalty, 7. and Gazdecki, A. (2017). *7 Statistics That Prove Mobile Apps Are Crucial For Customer Loyalty*. [online] Small Business Trends. Available at: https://smallbiztrends.com/2016/09/using-apps-to-build-customer-loyalty.html [Accessed 21 Jan. 2017].

[3] Business 2 Community. (2017). Mobile Apps Usage – Statistics and Trends [Infographic]. [online] Available at: http://www.business2community.com/infographics/mobile-apps-usage-statistics-trends-infographic-01248837#uH8wsRew8RkOyAxb.97 [Accessed 21 Jan. 2017].

[4] iGEST - Emita faturas e guias de transporte com o seu telemóvel. (2017). iGEST. [online] Available at: https://www.igest.pt/igest/ [Accessed 21 Jan. 2017].

[5] Económico, J., Barroso, B., Simões, L., Lourinho, J., Alves, M., Laxmidas, S., Ferreira and Ricardo Junqueiro, A. (2017). *Iniciativa "Papel Zero": hipermercados podem deixar de emitir faturas*. [online] O Jornal Económico. Available at: http://www.jornaleconomico.sapo.pt/noticias/iniciativa-papel-zero-hipermercados-podem-deixar-emitir-faturas-107176 [Accessed 24 Jan. 2017].

[6] Shane, S. and more, R. (2017). WhyYourSmall Business Needs a Mobile App. [online] Entrepreneur. Available at: https://www.entrepreneur.com/article/269978 [Accessed 23 Jan. 2017].

[7] Archer, S. andArcher, S. (2017). Mobile Apps Are More Important Than Ever in Business. [online] Tech.co. Available at: http://tech.co/mobile-apps-important-business-2016-07 [Accessed 23 Jan. 2017].

[8] Login.saphety.com. (2017). Doc | Saphety — Customer Secure Login. [online] Available at: https://login.saphety.com/pt/doc/ [Accessed 23 Jan. 2017].

[9] www3.saphety.com. (2017). Saphety - Global Network Solutions. [online] Available at: http://www3.saphety.com/ [Accessed 23 Jan. 2017].

[10] Doc.saphety.com. (2017). SaphetyDoc. [online] Available at: http://doc.saphety.com/site/Help/saphetydoc/PT/ [Accessed 23 Jan. 2017].

[11] Nos.pt. (2017). Madeira - NOS. [online] Available at: http://www.nos.pt/particulares/madeira/Pages/home.aspx [Accessed 23 Jan. 2017].

[12] Nos.pt. (2017). Fatura eletrónica e débito direto - NOS. [online] Available at: http://www.nos.pt/particulares/ajuda/faturas-pagamentos/Pages/fatura-eletronica-debito-direto.aspx [Accessed 23 Jan. 2017].

[13] Play.google.com. (2017). Cite a Website - Cite This For Me. [online] Available at: https://play.google.com/store/apps/details?id=pt.nos.selfcare&hl=pt_PT [Accessed 23 Jan. 2017].

[14] NOS, C. and NOS Comunicacoes, S. (2017). Cliente NOS na AppStore. [online] AppStore. Available at: https://itunes.apple.com/pt/app/cliente-nos/id1105930725?mt=8 [Accessed 23 Jan. 2017].

[15] B2brouter.net. (2017). Free On Line Electronic Invoice Service - Portal B2B Router.net. [online] Available at: https://www.b2brouter.net/ [Accessed 23 Jan. 2017].

[16] Play.google.com. (2017). Citação um website - Cite This For Me. [online] Available at: https://play.google.com/store/apps/details?id=com.b2brouter.b2brouter [Accessed 23 Jan. 2017].

[17] INVINET SISTEMES 2003, S. (2017). B2BRouter na AppStore. [online] AppStore. Available at: https://itunes.apple.com/pt/app/b2brouter/id1074858903?mt=8 [Accessed 23 Jan. 2017].

[18] B2brouter.net. (2017). Free On Line Electronic Invoice Service - Portal B2B Router.net. [online] Available at: https://www.b2brouter.net/en/help [Accessed 23 Jan. 2017].

[19] Anon, (2017). [online] Available at: https://www.viactt.pt/website/index.html [Accessed 23 Jan. 2017].

[20] Anon, (2017). [online] Available at: https://www.viactt.pt/website/entidades_aderentes.html [Accessed 23 Jan. 2017].

[21] Anon, (2017). [online] Available at: https://www.viactt.pt/website/o_que_e.html [Accessed 23 Jan. 2017].

[22] Anon, (2017). [online] Available at: https://www.viactt.pt/website/como_funciona.html [Accessed 23 Jan. 2017].

[23]Tradeshift.com. (2017). Tradeshift | Free, easy electronic invoice management: Tradeshift. [online] Available at: https://tradeshift.com/solutions/electronic-invoicing/ [Accessed 23 Jan. 2017].

[24] Machado, p. (2017). *Aplicações Móveis: Nativas ou Web?*. [online] Devmedia.com.br. Available at: http://www.devmedia.com.br/aplicacoes-moveis-nativas-ou-web/30392 [Accessed 24 Jan. 2017].

[25] Murarolli, P. andGirotti, M. (2015). *Inovações tecnológicas nas perspectivas computacionais*. 1st ed. São Paulo - Brasil.

[26] Concretesolutions.com.br. (2017). *Nativo x Híbrido – A Discussão Final! (Parte 1):: Concrete Solutions*. [online] Available at: http://www.concretesolutions.com.br/2016/03/16/nativo-x-hibrido/ [Accessed 24 Jan. 2017].

[27] Play.google.com. (2017). *Cite a Website - Cite This For Me*. [online] Available at: https://play.google.com/store?hl=pt_PT [Accessed 24 Jan. 2017].

[28] Itunes.apple.com. (2017). *AppStore Descargas no iTunes*. [online] Available at: https://itunes.apple.com/pt/genre/ios/id36?mt=8 [Accessed 24 Jan. 2017].

[29] Dre.pt. (2017). *Pesquisa - DRE*. [online] Available at: https://dre.pt/web/guest/pesquisa/-/search/520642/details/normal [Accessed 24 Jan. 2017].

[30] Dre.pt. (2017). *Pesquisa - DRE*. [online] Available at: https://dre.pt/web/guest/pesquisa/-/search/243356/details/normal [Accessed 24 Jan. 2017].

[31] Anon, (2017). [online] Available at: http://www.vda.pt/xms/files/Newsletters/Flash_Fiscal_-_Novas_regras_sobre_faturacao_eletronica_-24.09.2012-.pdf [Accessed 24 Jan. 2017].

[32] Anon, (2017). [online] Available at: http://www.occ.pt/fotos/downloads/files/1221846186_60a66_Consultorio.pdf [Accessed 24 Jan. 2017].

[33] Cordova.apache.org. (2017). *Apache Cordova*. [online] Available at: https://cordova.apache.org/ [Accessed 25 Jan. 2017].

[34] Onsen UI 2: Beautiful HTML5 Hybrid Mobile App Framework and Tools. (2017). [online] Available at: https://onsen.io/ [Accessed 25 Jan. 2017].

[35] Ionic Framework. (2017). *Ionic Framework*. [online] Available at: http://ionicframework.com/ [Accessed 25 Jan. 2017].

[36] Genuitec. (2017). *Gap Debug :: Mobile App Debugger*. [online] Available at: https://www.genuitec.com/products/gapdebug/ [Accessed 25 Jan. 2017].

[37] Monaca.io. (2017). *Cloud-Powered HTML5 Hybrid Mobile App Development Tools | Monaca*. [online] Available at: https://monaca.io/ [Accessed 25 Jan. 2017].

[38] Matheus Danemberg. (2017). *Desvendando o Phonegap/Cordova*. [online] Available at: http://blog.danemberg.com/desvendando-o-phonegap-cordova/ [Accessed 26 Jan. 2017].

[39] 6 aspectos essenciais para decidir entre aplicações mobile híbridas e nativas. (2017). [online] Medium. Available at: https://medium.com/@waldyrfelix/6-aspectos-essenciais-para-decidir-entre-aplicações-mobile-híbridas-e-nativas-51bce0dace68#.4pt38b2h7 [Accessed 26 Jan. 2017].

[40] Apteligent. (2017). *Top 5 reasons to try Cordova / Phonegap to build your mobile app*. [online] Available at: https://www.apteligent.com/2015/01/top-5-reasons-to-try-cordova/ [Accessed 26 Jan. 2017].

[41] Ruby-lang.org. (2017). *Linguagem de Programação Ruby*. [online] Available at: https://www.ruby-lang.org/pt/ [Accessed 26 Jan. 2017].

[42] Rubymotion.com. (2017). *FrequentlyAskedQuestions | RubyMotion*. [online] Available at: http://www.rubymotion.com/support/faq/ [Accessed 26 Jan. 2017].

[43] Xamarin.com. (2017). *Mobile App Development & App Creation Software - Xamarin*. [online] Available at: https://www.xamarin.com/ [Accessed 29 Jan. 2017].

[44] Xamarin.com. (2017). *Mobile Application Development to BuildApps in C# - Xamarin*. [online] Available at: https://www.xamarin.com/platform [Accessed 29 Jan. 2017].

[45] Xamarin.com. (2017). *Mobile App Testing On Hundreds Of Devices - XamarinTestCloud*. [online] Available at: https://www.xamarin.com/test-cloud [Accessed 29 Jan. 2017].

[46] Xamarin.com. (2017). *Mobile AppTracking, Monitoring, and Crash Reporting — Xamarin Insights*. [online] Available at: https://www.xamarin.com/insights [Accessed 29 Jan. 2017].

[47] MobiLoud. (2017). *Native, Web orHybridApps? What'sTheDifference? | MobiLoud*. [online] Available at: https://www.mobiloud.com/blog/native-web-or-hybrid-apps/ [Accessed 30 Jan. 2017].

[48]Pinto, M. (2017). *Apps nativas vs Web apps: Quais devo escolher? - Pplware*. [online] Pplware. Available at: https://pplware.sapo.pt/internet/apps-nativas-vs-web-apps-quais-devo-escolher/ [Accessed 30 Jan. 2017].

[49] EasyWay To createyour Mobile and Website Development. (2017). *Top 5 Best PHP Frameworks for 2016 To Become a Master Developer*. [online] Available at: https://webandmobiletech.wordpress.com/2016/01/01/top-5-best-php-frameworks-for-2016-to-become-a-master-developer/ [Accessed 6 Feb. 2017].

[50] Best Responsive web design& mobile apps Development Company India. (2017). *Why Laravel is the best PHP framework for 2016?*. [online] Available at: http://www.amarinfotech.com/why-laravel-is-best-php-framework-in-2016.html [Accessed 6 Feb. 2017].

[51] Phalcon PHP. (2017). *Phalcon - High Performance PHP Framework*. [online] Availableat: https://phalconphp.com/pt/ [Accessed 6 Feb. 2017].

[52] PHP, P. (2017). *Welcome — Phalcon 3.0.2 documentation (English)*. [online] Docs.phalconphp.com. Available at: https://docs.phalconphp.com/en/latest/index.html [Accessed 6 Feb. 2017].

[53] Otwell, T. (2017). *Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: https://laravel.com/[Accessed 4 Oct. 2017].

[54] Otwell, T. (2017). *Installation - Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: https://laravel.com/docs/5.4[Accessed 6 Feb. 2017].

[55] Symfony.com. (2017). *Symfony, High Performance PHP Framework for Web Development*. [online] Available at: https://symfony.com/[Accessed 6 Feb. 2017].

[56] Symfony.com. (2017). *Symfony 3.2 Documentation*. [online] Available at: https://symfony.com/doc/current/index.html [Accessed 6 Feb. 2017].

[57] Yiiframework.com. (2017). *Yii PHP Framework: Best for Web 2.0 Development*. [online] Availableat: http://www.yiiframework.com/ [Accessed 6 Feb. 2017].

[58] Yiiframework.com. (2017). *TheDefinitive Guide to Yii 2.0*. [online] Available at: http://www.yiiframework.com/doc-2.0/guide-index.html [Accessed 6 Feb. 2017].

[59] CakePHP - Therapiddevelopmentphp framework. (2017). *CakePHP - Buildfast, growsolid | PHP Framework | Home*. [online] Available at: https://cakephp.org/ [Accessed 6 Feb. 2017].

[60] Book.cakephp.org. (2017). *Welcome*. [online] Available at: https://book.cakephp.org/3.0/en/index.html [Accessed 6 Feb. 2017].

[61] Codeigniter.com. (2017). *CodeIgniter Web Framework*. [online] Available at: https://codeigniter.com/[Accessed 6 Feb. 2017].

[62] Codeigniter.com. (2017). *CodeIgniter User Guide — CodeIgniter 3.1.3 documentation*. [online] Available at: https://codeigniter.com/user_guide/ [Accessed 6 Feb. 2017].

[63] Pesquisa - DRE. (2017). [online] Dre.pt. Available at: https://dre.pt/web/guest/pesquisa/-/search/232885/details/normal?q=Decreto-Lei+n. 62/2003 [Accessed 17 Feb. 2017].

[64] Social.technet.microsoft.com. (2017). Visual Studio 2015: Implementando uma aplicação Web API - artigos TechNet - Brasil (Português) - TechNet Wiki. [online] Available at: https://social.technet.microsoft.com/wiki/pt-br/contents/articles/30528.visual-studio-2015-implementando-uma-aplicacao-web-api.aspx [Accessed 19 Sep. 2017].

[65] Jwt.io. (2017). JWT.IO. [online] Available at: https://jwt.io/ [Accessed 21 Sep. 2017].

[66] Tools.ietf.org. (2017). RFC 7231 - Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. [online] Available at: https://tools.ietf.org/html/rfc7231#page-47 [Accessed 21 Sep. 2017].

[67] Phpunit.de. (2017). PHPUnit – The PHP Testing Framework. [online] Available at: https://phpunit.de/ [Accessed 4 Oct. 2017].

[68] Apidocjs.com. (2017). apiDoc - Inline Documentation for RESTful web APIs. [online] Available at: http://apidocjs.com/ [Accessed 4 Oct. 2017].

[69] Angular.io. (2017). Angular Docs. [online] Available at: https://angular.io/docs [Accessed 27 Nov. 2017].

[70] Solutions, A. (2017). ACIN iCloud Solutions. [online] ACIN iCloud Solutions - Pioneira na criação de soluções tecnológicas. Available at: http://www.acin.pt/pt [Accessed 5 Dec. 2017].

10 Anexos

10.1 Anexo A – Exemplo de aplicação das frameworks

Utilizando as frameworks Laravel e Yii, foi realizado um pequeno exercício que tem como base a criação de um formulário para inserir ou editar dados do utilizador e depois apresentar uma listagem com todos os dados dos utilizadores. Nessa listagem também é possível remover os utilizadores.

Em termos de base de dados apenas foi criado uma tabela utilizador com os campos "*id*", "*firstname*", "*lastname*" e "*email*". Em cada aplicação do exercício utilizando as duas frameworks, foi registado os tempos de instalação/configuração e implementação.

10.1.1 Laravel

Deste modo, começou-se por instalar o xampp v5.6.30, composer v1.3.2 e depois o Laravel v5.4 através do comando "*composer create-project --prefer-distlaravel/laravel laravel_teste*" utilizando o composer. Com o projeto criado, obtivemos a seguinte estrutura de pastas:

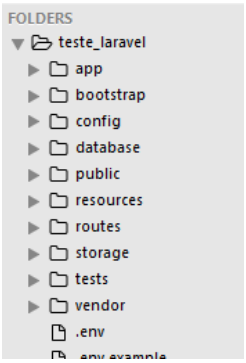


Figura 53 - Estrutura de pastas do laravel

Depois de a instalação estar concluída, passou-se por criar uma base de dados "test" completamente vazia no *phpMyAdmin* e de seguida configurar no ficheiro ".env" os dados de acesso à base de dados.

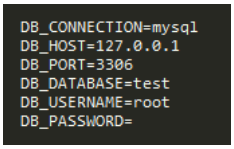


Figura 54 - Ficheiro ".env" com as configurações da base de dados

No Laravel é possível criar ficheiros de migração, que permite criar e alterar facilmente a base de dados do projeto. Logo através do comando "*php artisan make:model Utilizador -m*" foi criado o modelo e o respectivo ficheiro de migração da tabela utilizador.

O ficheiro de migração tem o método *up* e *down*. O método up é utilizado para adicionar novas tabelas, campos, índices e o método *down* serve para inverter as operações do método *up*. Logo no método *up* foi indicado os campos da tabela utilizador e depois com o comando "*php artisan migrate*" passou-se os dados do ficheiro de migração para a base de dados no *phpMyAdmin*.

```php
class CreateUtilizadorsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('utilizador', function (Blueprint $table) {
            $table->increments('id');
            $table->string('firstname');
            $table->string('lastname');
            $table->string('email');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('utilizador');
    }
}
```

Figura 55 - Ficheiro de migração da tabela utilizador

Quanto ao modelo do utilizador, criou-se todos os métodos relacionados com o utilizador tais como criar, editar e remover utilizador.

```php
class Utilizador extends Model
{
    protected $table = 'utilizador';
    protected $fillable = ['firstname', 'lastname', 'email', 'created_at', 'updated_at'];

    /**
     * Método para criar novo utilizador
     *
     * @param  string  $firstname
     * @param  string  $lastname
     * @param  string  $email
     */
    public function addUser($firstname, $lastname, $email) {
        // Inserir novo utilizador
        $this->create([
            'firstname' => $firstname,
            'lastname'  => $lastname,
            'email'     => $email
        ]);
    }


    /**
     * Método para remover utilizador
     *
     * @param  int  $id
     */
    public function deleteUser($id) {
        // Remover utilizador
        $this->destroy($id);
    }

    /**
     * Método para editar utilizador
     *
     * @param  int     $id
     * @param  string  $firstname
     * @param  string  $lastname
     * @param  string  $email
     */
    public function editUser($id, $firstname, $lastname, $email) {
        // Editar dados utilizador
        $this->where('id', $id)->update([
            'firstname' => $firstname,
            'lastname'  => $lastname,
            'email'     => $email
        ]);
    }
```

Figura 56 - Modelo do utilizador

Com todas as operações relacionadas com a base de dados concluídas, foi criado a parte visual do exercício e todo o seu funcionamento. Para tal foi necessário criar um controlador, uma vista e por fim indicar no ficheiro "*web.php*" todas as rotas *web* para o sistema funcionar. Para criar o controlador utilizou-se o comando "*php artisan make:controller HomeController*".

```php
class HomeController extends Controller {
    /**
     * Listar os utilizadores
     */
    public function show() {
        $users = Utilizador::all();
        return view('welcome', compact('users'));
    }

    /**
     * Criar novo utilizador
     */
    public function addUser(Utilizador $user) {
        // Validar dados do formulário
        $this->validate(request(),[
            'firstname' => 'required',
            'lastname'  => 'required',
            'email'     => 'required|email'
        ]);

        // Criar utilizador
        $user->addUser(request('firstname'), request('lastname'), request('email'));
        return back();
    }

    /**
     * Remover utilizador
     * @param  int  $id
     */
    public function deleteUser(Utilizador $user, $id) {
        // Remover cliente
        $user->deleteUser($id);
        return back();
    }

    /**
     * Mostrar os dados do utilizador para editar
     * @param  int  $id
     */
    public function showEditUser($id) {
        // Dados de todos os utilizadores
        $users = Utilizador::all();
        // Dados apenas do utilizador que será editado
        $user  = Utilizador::find($id);
        return view('welcome', compact('users', 'user'));
    }

    /**
     * Editar utilizador
     * @param  int  $id
     */
    public function editUser(Utilizador $user, $id) {
        // Validar dados do formulário
        $this->validate(request(),[
            'firstname' => 'required',
            'lastname'  => 'required',
            'email'     => 'required|email'
        ]);
        // Editar dados do utilizador
        $user->editUser($id, request('firstname'), request('lastname'), request('email'));
        return redirect('/');
    }
}
```

Figura 57 - Controlador HomeController

No ficheiro "*web.php*" foi definido todas as rotas possíveis do sistema e para cada rota foi associada um método do controlador.

Tabela 17 - Todas as rotas do sistema e descrição.

Rota	Descrição
/	Apresenta a página principal com o formulário para criar novo utilizar e a tabela com todos os utilizadores.
/deleteUser/{id}	Remove o utilizador e depois apresenta a página principal.
/showEditUser/{id}	Apresenta a página para editar os dados do utilizador.
/addUser	Adicionar novo utilizador e depois apresenta a página principal.
/editUser/{id}	Editar dados do utilizador e depois apresenta a página principal.

```
11
12  */
13  Route::get('/', 'HomeController@show');
14  Route::get('/deleteUser/{id}', 'HomeController@deleteUser');
15  Route::get('/showEditUser/{id}', 'HomeController@showEditUser');
16
17  Route::post('/addUser', 'HomeController@addUser');
18  Route::post('/editUser/{id}', 'HomeController@editUser');
19
```

Figura 58 - Ficheiro "web.php" com todas as rotas definidas

Quanto a vista do sistema, foi criado um formulário que adapta-se tanto para criar e editar utilizadores, uma zona para apresentar os erros dos campos do formulário e por fim uma tabela com todos os dados dos utilizadores existentes. Através do comando "*php artisan make:view welcome*" foi criado a vista welcome.

```
<div class="col-md-4 formContent">
    <!-- Formulário para inserir ou editar utilizador -->
    <div class="form-group">
        <form method="POST" action="{{ !isset($user) ? '/teste_laravel/public/addUser' : '/teste_laravel/public/editUser/'.$user->id }} ">
            {{ csrf_field() }}
            <legend>{{ !isset($user) ? 'Form Insert' : 'Form Edit' }}</legend>
            <fieldset>
                <div class="form-group">
                    <label for="firstname">Firstname:</label>
                    <input type="firstname" class="form-control" id="firstname" name="firstname" value="{{ isset($user) ? $user->firstname : '' }}">
                </div>
                <div class="form-group">
                    <label for="lastname">Lastname:</label>
                    <input type="lastname" class="form-control" id="lastname" name="lastname" value="{{ isset($user) ? $user->lastname : '' }}">
                </div>
                <div class="form-group">
                    <label for="email">Email:</label>
                    <input type="email1" class="form-control" id="email" name="email" value="{{ isset($user) ? $user->email : '' }}">
                </div>
                <button type="submit" class="btn btn-default">{{ !isset($user) ? 'Insert' : 'Edit' }}</button>
                @if (isset($user))
                    <a href="/teste_laravel/public/"><button type="button" class="btn btn-default">Cancel</button></a>
                @endif
            </fieldset>
        </form>
    </div>

    <!-- Apresentar os erros dos campos do formulário -->
    @if (count($errors))
        <div class="form-group">
            <div class="alert alert-danger">
                <ul>
                    @foreach ($errors->all() as $error)
                        <li>{{ $error }}</li>
                    @endforeach
                </ul>
            </div>
        </div>
    @endif
</div>
```

Figura 59 - Formulário para criar ou editar utilizadores na vista welcome

```
<!-- Listagem dos utilizadores -->
<div class="col-md-8">
    <table class="table">
        <thead>
            <tr>
                <th>Firstname</th>
                <th>Lastname</th>
                <th>Email</th>
                <th>Edit</th>
                <th>Delete</th>
            </tr>
        </thead>
        <tbody>
            @if (count($users))
                @foreach ($users as $user)
                    <tr>
                        <td>{{ $user->firstname }}</td>
                        <td>{{ $user->lastname }}</td>
                        <td>{{ $user->email }}</td>
                        <td><a href="/teste_laravel/public/showEditUser/{{ $user->id }}"><span class="glyphicon glyphicon-pencil" aria-hidden="true"></span></a></td>
                        <td><a href="/teste_laravel/public/deleteUser/{{ $user->id }}"><span class="glyphicon glyphicon-trash" aria-hidden="true"></span></a></td>
                    </tr>
                @endforeach
            @else
                <tr><td>No results</td></tr>
            @endif
        </tbody>
    </table>
</div>
```

Figura 60- Tabela com todos os dados dos utilizadores na vista welcome

Em termos de apresentação do pequeno exercício utilizando a *framework* Laravel, segue-se as seguintes imagens.

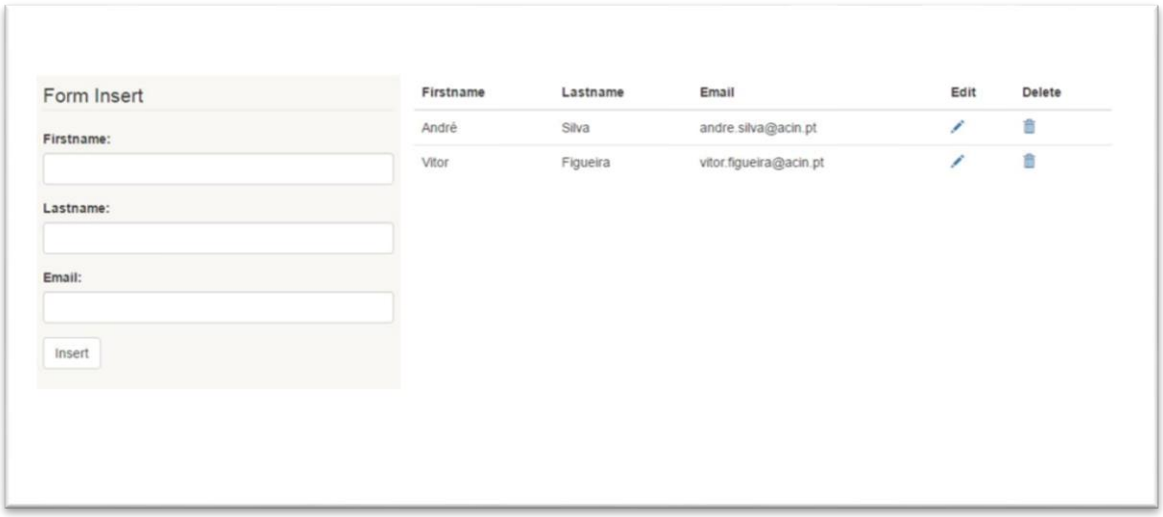


Figura 61 - Página principal que permite inserir, editar ou remover utilizadores

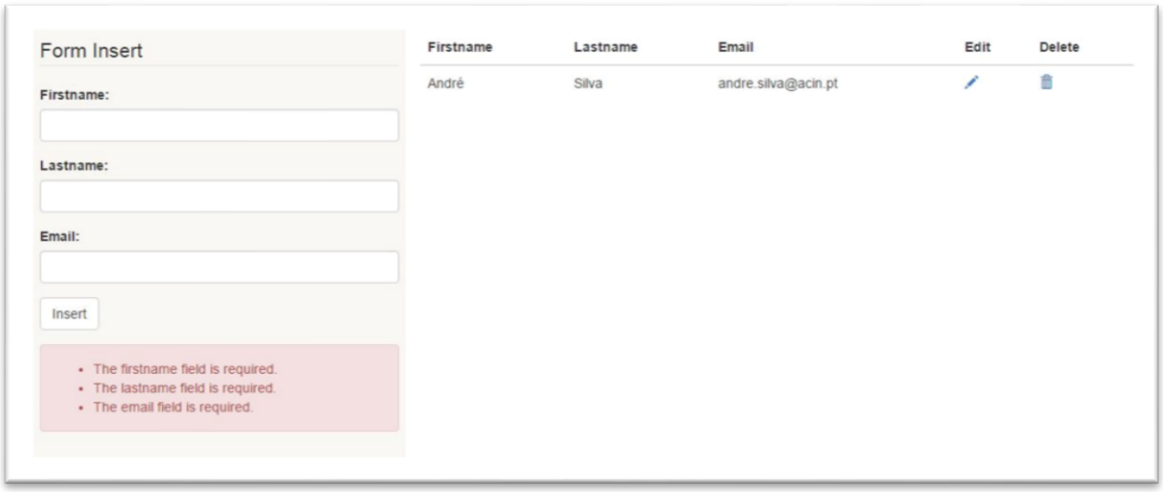


Figura 62 - Validação dos campos do formulário

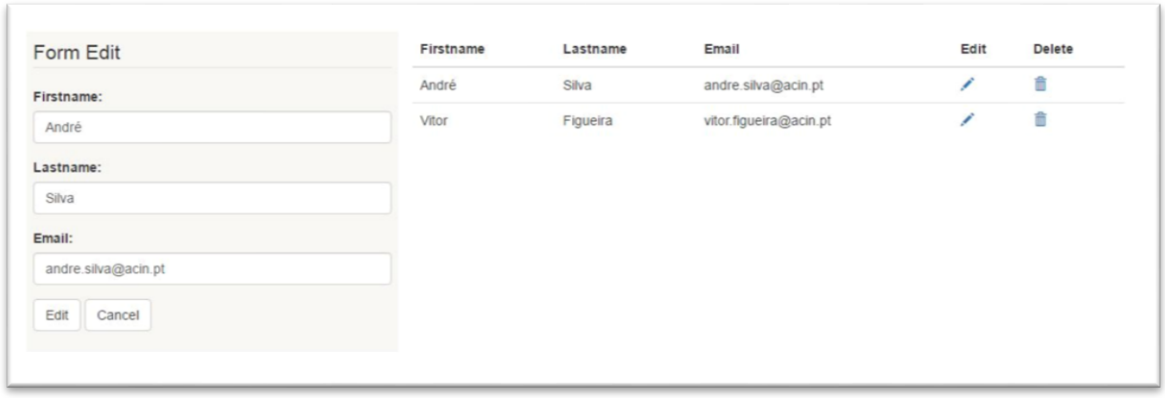


Figura 63 - Página para editar utilizadores

10.1.2 Yii

Com o xampp e o composer instalado no exercício anterior com o Laravel, passou-se logo para a instalação da *framework* Yii com o comando "*composer create-project --prefer-distyiisoft/yii2-app-basic basic*" através do composer. Deste modo, tivemos a seguinte estrutura de pastas:

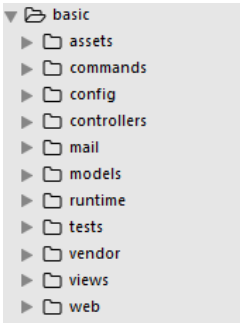


Figura 64 - Estrutura de pastas do Yii

Depois da instalação, configurou-se no ficheiro "*db.php*" dentro da pasta config o acesso a base de dados. Neste exercício também foi utilizado a base de dados "*test*" e a tabela utilizador.

```php
<?php

return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=test',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
];
```

Figura 65 - Ficheiro "db.php" com as configurações da base de dados

Ainda antes de qualquer desenvolvimento do pequeno exercício, foi necessário indicar a chaves secreta de validação em "*cookieValidationKey*" no ficheiro "web.php". Também nesses mesmo ficheiro foi configurado o "*urlManager*" para obter-se URL's mais limpos e mais fáceis de interpretar.

```php
$params = require(__DIR__ . '/params.php');

$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'components' => [
        'urlManager' => ['showScriptName' => true, 'enablePrettyUrl' => true],
        'request' => [
            // !!! insert a secret key in the following (if it is empty) - this is
            'cookieValidationKey' => 'andresilva13',
        ],
        'cache' => [
            'class' => 'yii\caching\FileCache',
        ],
        'user' => [
            'identityClass' => 'app\models\User',
            'enableAutoLogin' => true,
        ],
```

Figura 66 - Ficheiro "web.php" com as configurações gerais do Yii

Com todas as configurações iniciais efetuadas, passou-se a implementação do exercício criando modelos, controladores e vistas de acordo com as necessidades. Em primeiro, criou-se o modelo "*Utilizador.php*" que trata de toda a manipulação dos dados do utilizador e o modelo "*UserForm.php*" que corresponde o formulário para criar e editar utilizadores.

```php
class Utilizador extends ActiveRecord {

    /**
     * Listar todos os utilizadores
     */
    public function getUtilizadores() {
        $users = Utilizador::find()->all();
        return $users;
    }

    /**
     * Listar um utilizador
     */
    public function getUtilizador($id) {
        $user = Utilizador::find()->where(['id' => $id])->one();
        return $user;
    }

    /**
     * Remover utilizador
     */
    public function removeUtilizador($id) {
        $user = Utilizador::findOne($id);
        $user->delete();
    }

    /**
     * Criar novo utilizador
     */
    public function newUtilizador($firstname, $lastname, $email) {
        $user = new Utilizador;
        $user->firstname = $firstname;
        $user->lastname  = $lastname;
        $user->email     = $email;
        $user->insert();
    }

    /**
     * Editar utilizador
     */
    public function editUtilizador($id, $firstname, $lastname, $email) {
        $user = Utilizador::getUtilizador($id);
        $user->firstname = $firstname;
        $user->lastname  = $lastname;
        $user->email     = $email;
        $user->save();
    }
}
```

Figura 67 - Modelo Utilizador

```php
class UserForm extends Model
{
    public $firstname;
    public $lastname;
    public $email;

    /**
     * @return array the validation rules.
     */
    public function rules()
    {
        return [
            // name, email, subject and body are required
            [['firstname', 'lastname', 'email'], 'required'],
            // email has to be a valid email address
            ['email', 'email'],
        ];
    }
}
```

Figura 68 - Modelo UserFrom

De seguida, criou-se a vista "*index.php*" com o formulário para criar ou editar utilizadores e a tabela para apresentar os dados dos utilizadores.

```php
<!-- Formulário para inserir ou editar utilizador -->
<div class="form-group">
    <?php $form = ActiveForm::begin(['id'     => 'user-form',
                                     'action' => isset($user) ? URL::to(['site/edituser', 'id' => $user['id']]) : URL::to(['site/insertuser']) ]); ?>
        <legend><?=!isset($user) ? 'Form Insert' : 'Form Edit'?></legend>
        <fieldset>
            <?= $form->field($model, 'firstname')->textInput(['autofocus' => true, 'value' => isset($user) ? $user['firstname'] : '']) ?>

            <?= $form->field($model, 'lastname')->textInput(['autofocus' => true, 'value' => isset($user) ? $user['lastname'] : '']) ?>

            <?= $form->field($model, 'email')->textInput(['value' => isset($user) ? $user['email'] : '']) ?>

            <div class="form-group">
                <?= Html::submitButton(!isset($user) ? 'Submit' : 'Edit', ['class' => 'btn btn-primary', 'name' => 'user-button']) ?>
                <?= isset($user) ? Html::a('Cancel', URL::to(['/'], true), ['class' => 'btn btn-primary', 'name' => 'user-button']) : '' ?>
            </div>
        </fieldset>
    <?php ActiveForm::end(); ?>
</div>
```

Figura 69 - Formulário para criar ou editar utilizadores

```php
<!-- Listagem dos utilizadores -->
<div class="col-md-8">
    <table class="table">
        <thead>
            <tr>
                <th>Firstname</th>
                <th>Lastname</th>
                <th>Email</th>
                <th>Edit</th>
                <th>Delete</th>
            </tr>
        </thead>
        <tbody>
            <?php
            if (count($users)) {
                foreach ($users as $user) {
                    ?>
                    <tr>
                        <td><?=$user['firstname']?></td>
                        <td><?=$user['lastname']?></td>
                        <td><?=$user['email']?></td>
                        <td><a href="<?=URL::to(['site/edit', 'id' => $user['id']])?>"><span class="glyphicon glyphicon-pencil" aria-hidden="true"></span></a></td>
                        <td><a href="<?=URL::to(['site/remove', 'id' => $user['id']])?>"><span class="glyphicon glyphicon-trash" aria-hidden="true"></span></a></td>
                    </tr>
                    <?php
                }
            } else {
                ?>
                <tr><td>No results</td></tr>
                <?php
            }
            ?>
        </tbody>
    </table>
</div>
```

Figura 70 - Tabela com todos os dados dos utilizadores

E por fim, foi criado o controlador "*SiteController.php*" que é responsável por receber todos os pedidos e decidir qual modelos e vistas utilizar.

```php
class SiteController extends Controller
{
    /** Displays index */
    public function actionIndex() {
        $model = new UserForm();
        $users = Utilizador::getUtilizadores();
        return $this->render('index', ['model' => $model, 'users' => $users]);
    }

    /** Displays remove */
    public function actionRemove() {
        Utilizador::removeUtilizador(Yii::$app->request->get('id'));
        return $this->goHome();
    }

    /** Displays edit */
    public function actionEdit() {
        $model = new UserForm();
        $users = Utilizador::getUtilizadores();
        $user  = Utilizador::getUtilizador(Yii::$app->request->get('id'));
        return $this->render('index', ['model' => $model, 'users' => $users, 'user' => $user]);
    }

    /** Displays insert user */
    public function actionInsertuser() {
        $model = new UserForm();
        if ($model->validate()) {
            return $this->refresh();
        }

        $request = Yii::$app->request;
        $firstname = $request->post('UserForm')['firstname'];
        $lastname  = $request->post('UserForm')['lastname'];
        $email     = $request->post('UserForm')['email'];
        Utilizador::newUtilizador($firstname, $lastname, $email);

        return $this->goHome();
    }

    /** Displays edit user */
    public function actionEdituser()
    {
        $model = new UserForm();
        if ($model->validate()) {
            return $this->refresh();
        }

        $request   = Yii::$app->request;
        $id        = $request->get('id');
        $firstname = $request->post('UserForm')['firstname'];
        $lastname  = $request->post('UserForm')['lastname'];
        $email     = $request->post('UserForm')['email'];

        Utilizador::editUtilizador($id, $firstname, $lastname, $email);
        return $this->goHome();
    }
}
```

Figura 71 - Controlador SiteController

Em termos de apresentação e funcionamento do pequeno exercício utilizando a *framework* Yii, segue-se as seguintes imagens.

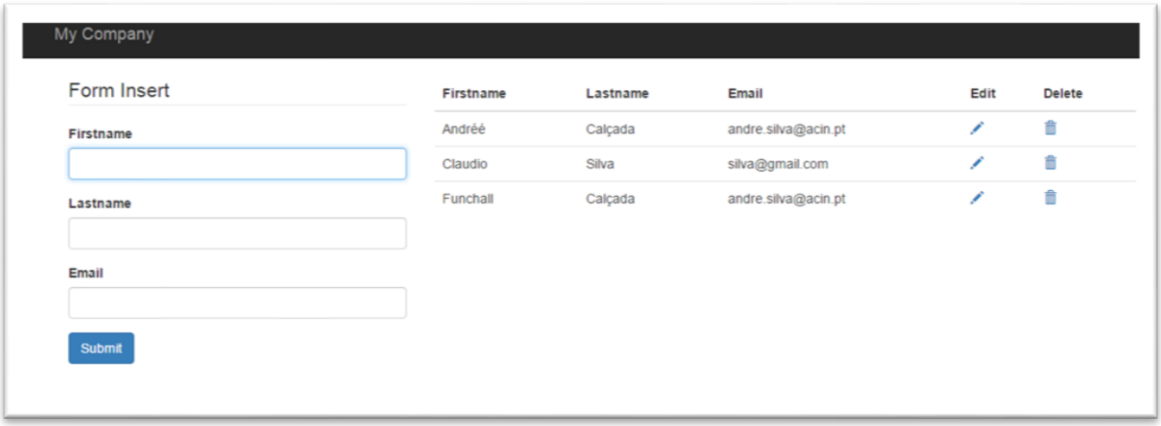


Figura 72- Página principal que permite inserir, editar ou remover utilizadores

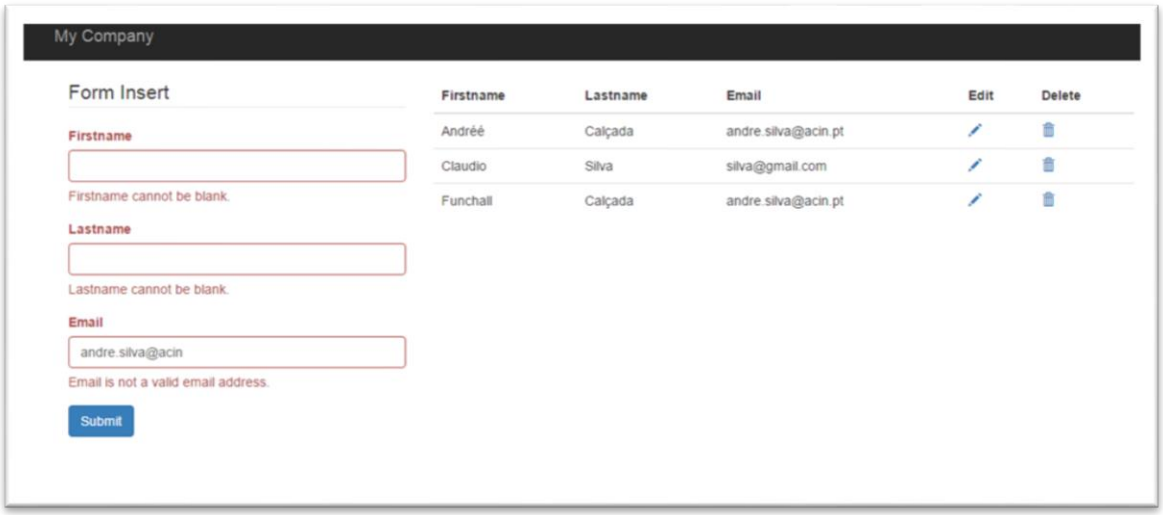


Figura 73 - Validação dos campos do formulário

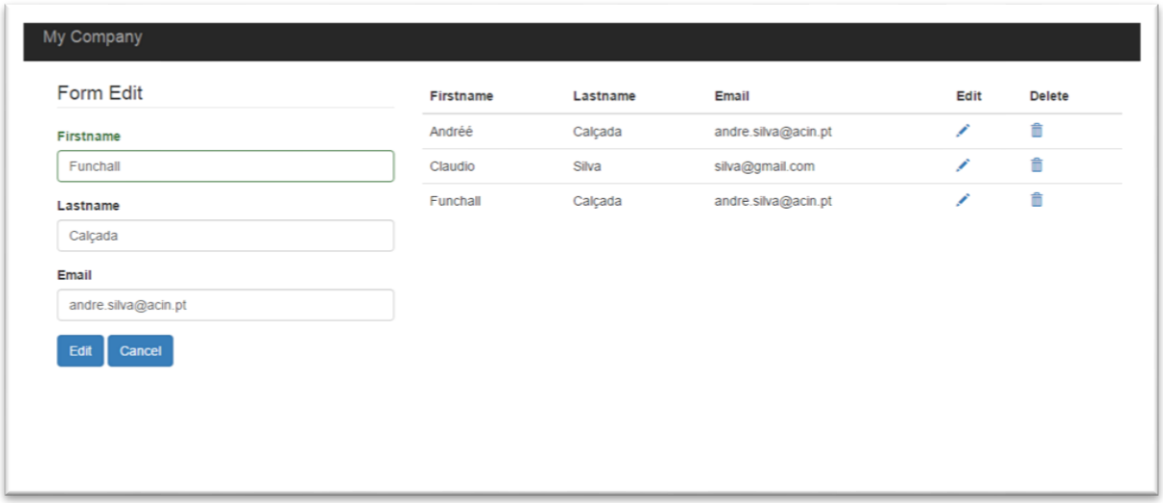


Figura 74 - Página para editar utilizadores

10.2 Anexo B – Diagramas

10.2.1 Casos de Utilização

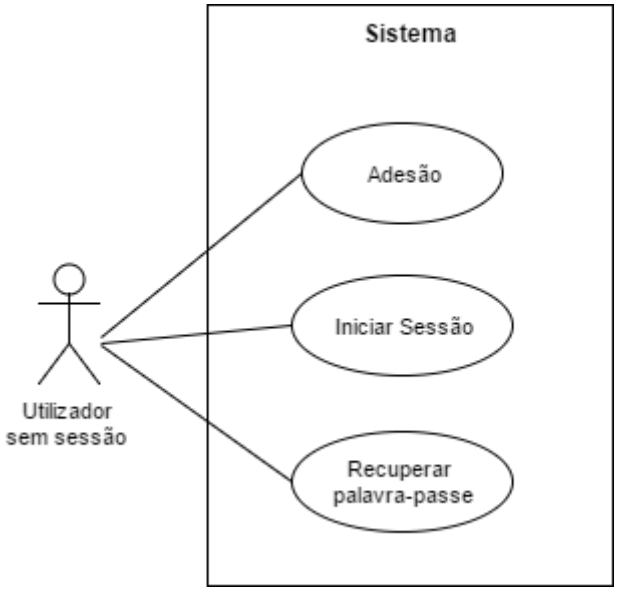


Figura 75 - Diagrama casos de utilização do utilizador sem sessão

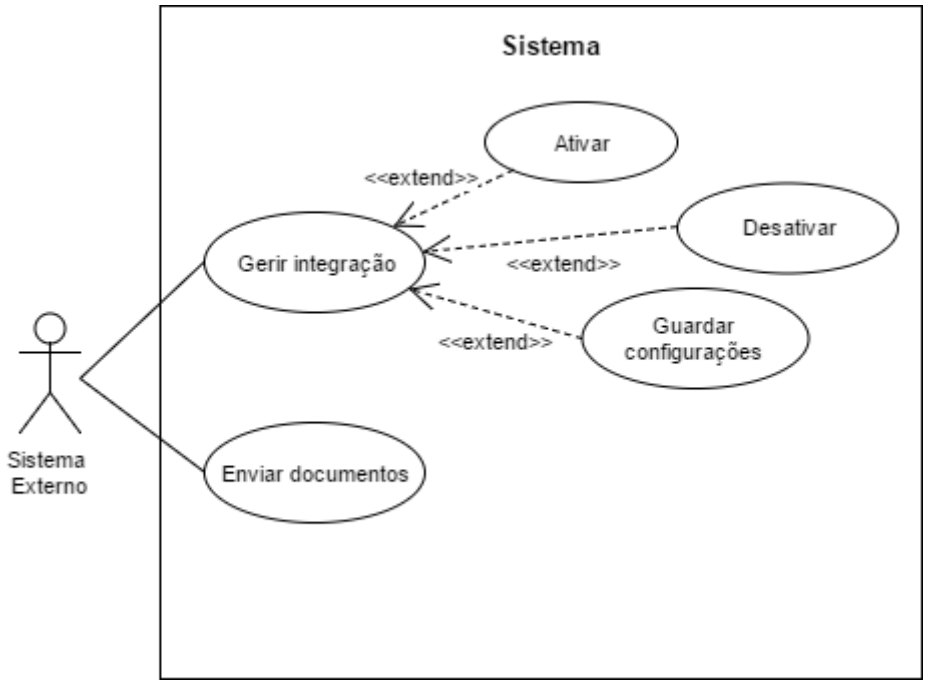


Figura 76 - Diagrama casos de utilização do sistema externo

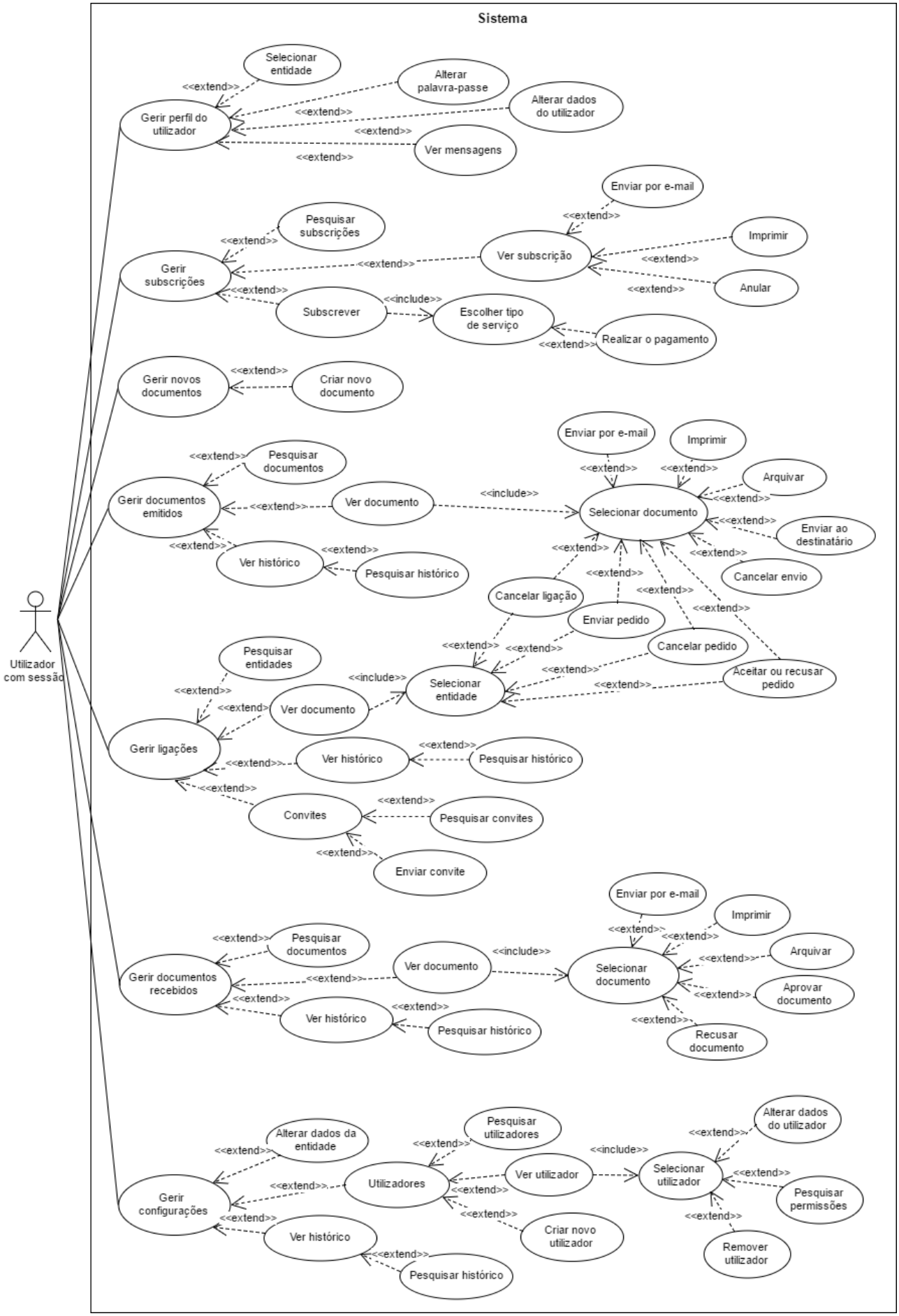


Figura 77 - Diagrama casos de utilização do utilizador com sessão

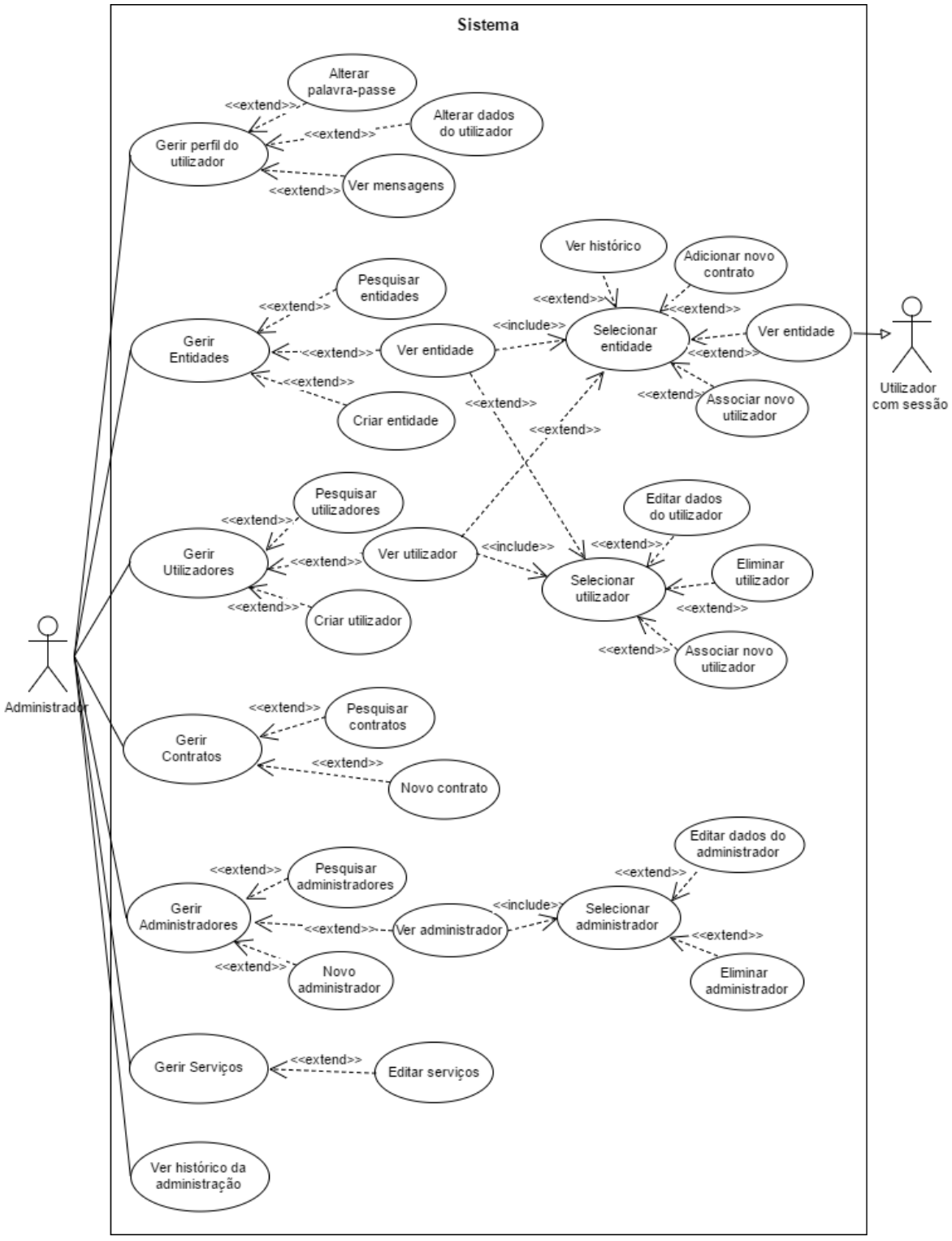


Figura 78 - Diagrama casos de utilização do administrador

10.2.2 Diagrama de Classes

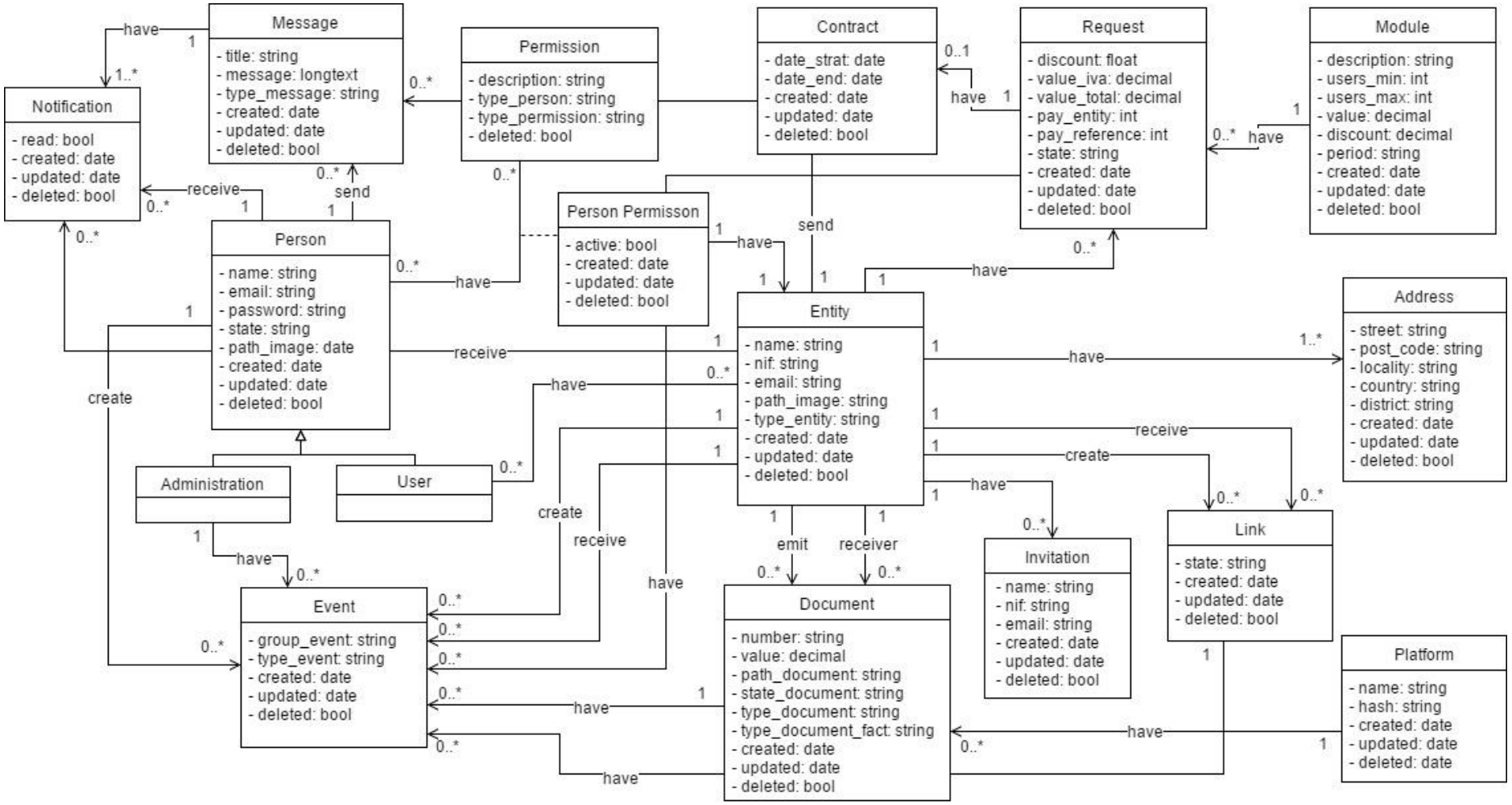


Figura 79 - Diagrama de classes completo do sistema

10.2.3 Diagramas de Atividade

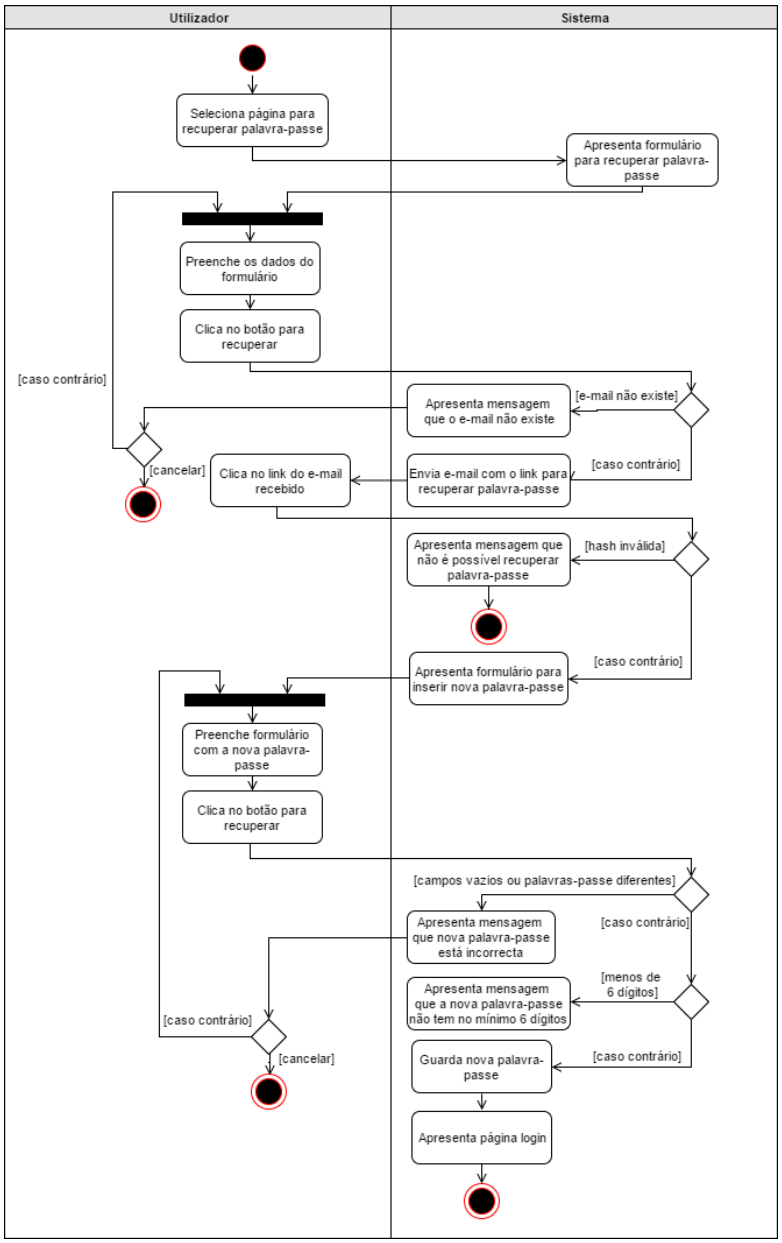


Figura 80 - Diagrama de atividades para recuperar palavra-passe

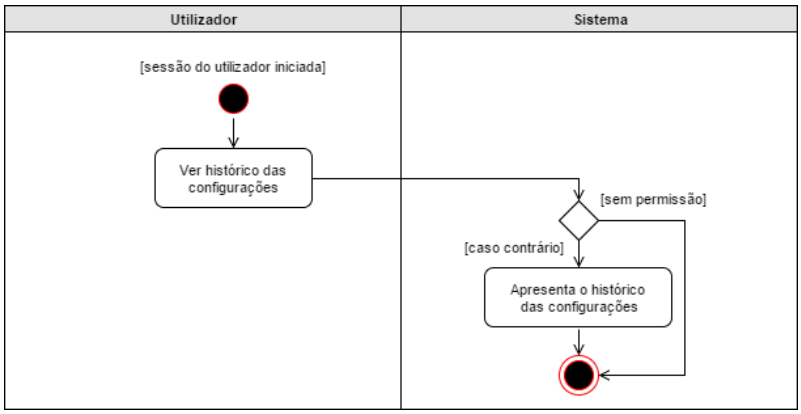


Figura 81 - Diagrama de atividades para ver histórico das configurações

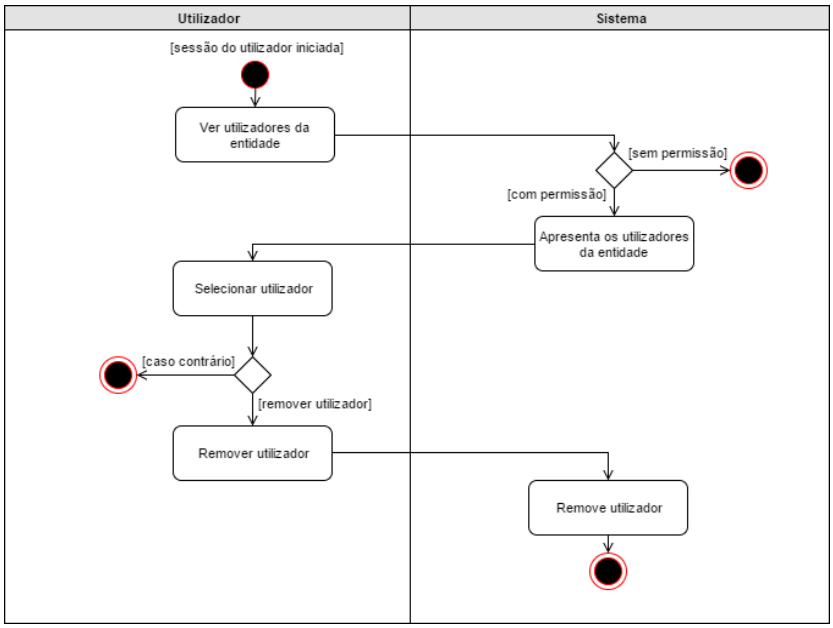


Figura 82 - Diagrama de atividade para remover utilizador da entidade

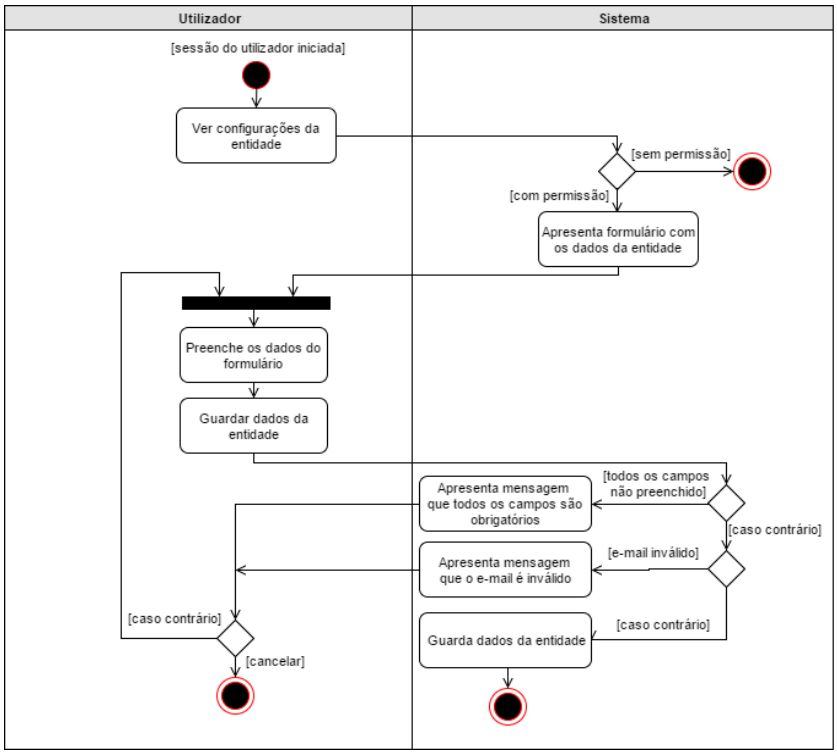


Figura 83 - Diagrama de atividade para alterar configurações da entidade

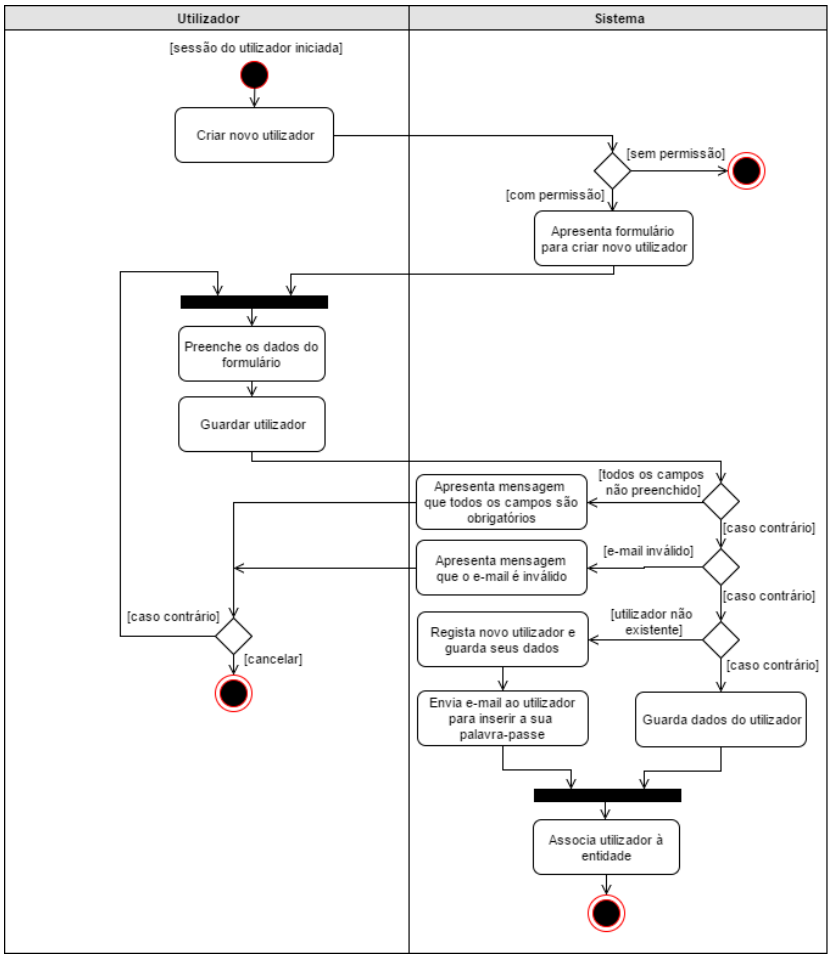


Figura 84 - Diagrama de atividade para associar novo utilizador a entidade

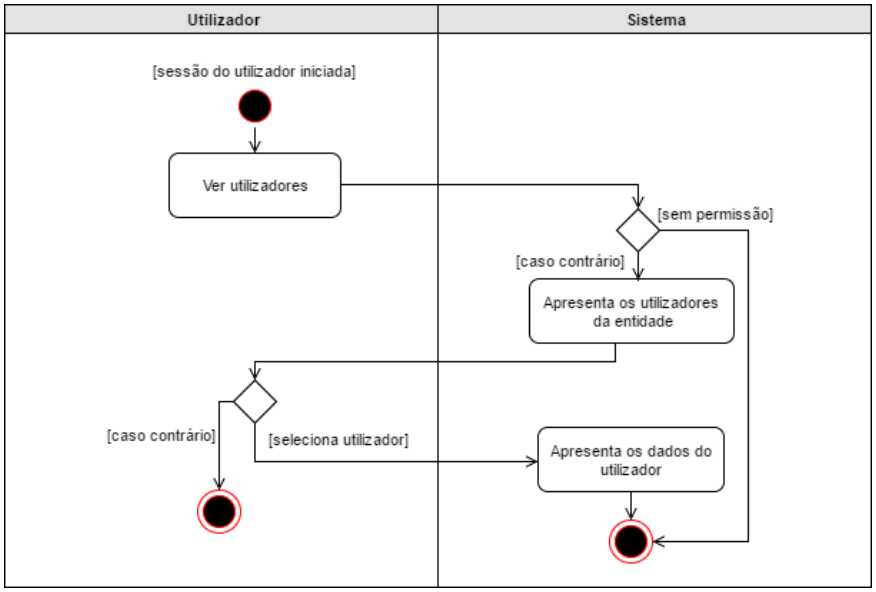


Figura 85 - Diagrama de atividade para ver utilizadores

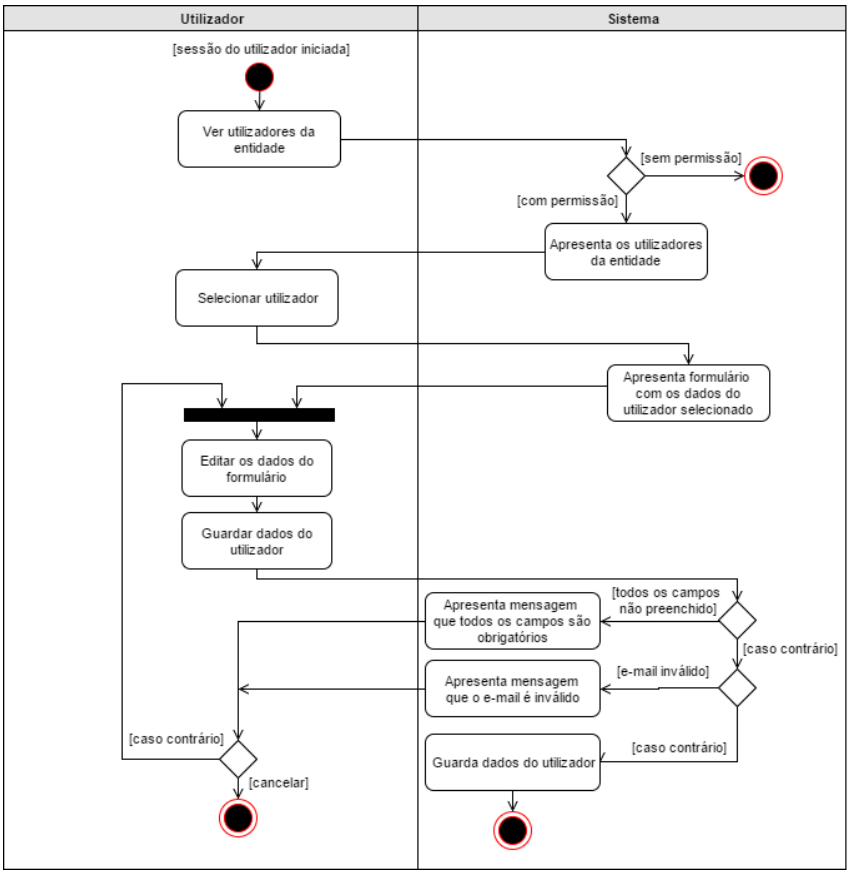


Figura 86 - Diagrama de atividade para editar utilizadores da entidade

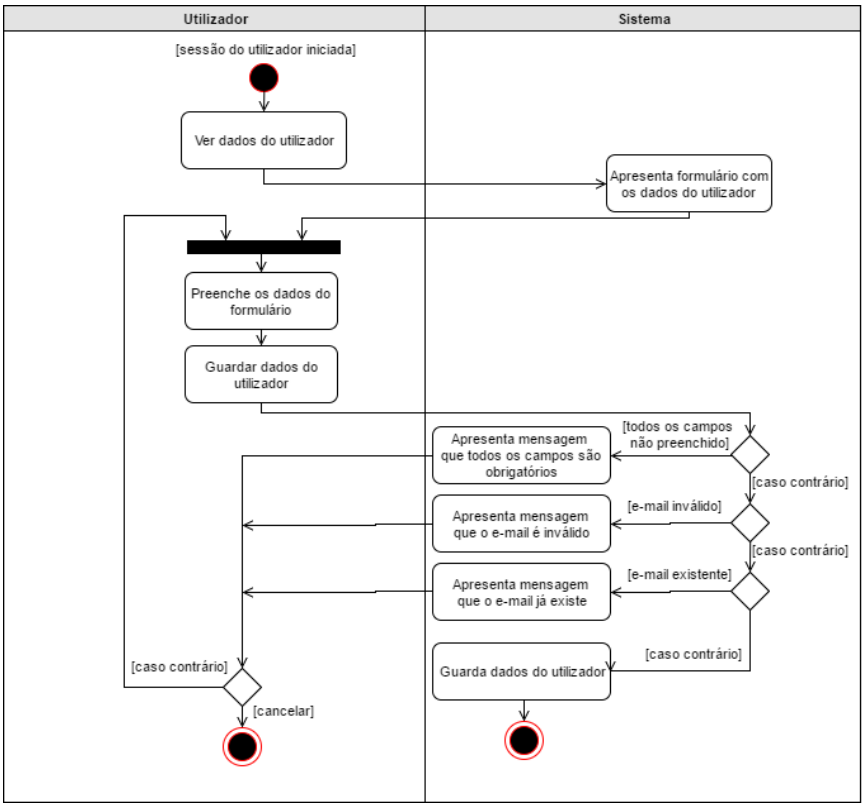


Figura 87 - Diagrama de atividade para alterar dados do utilizador

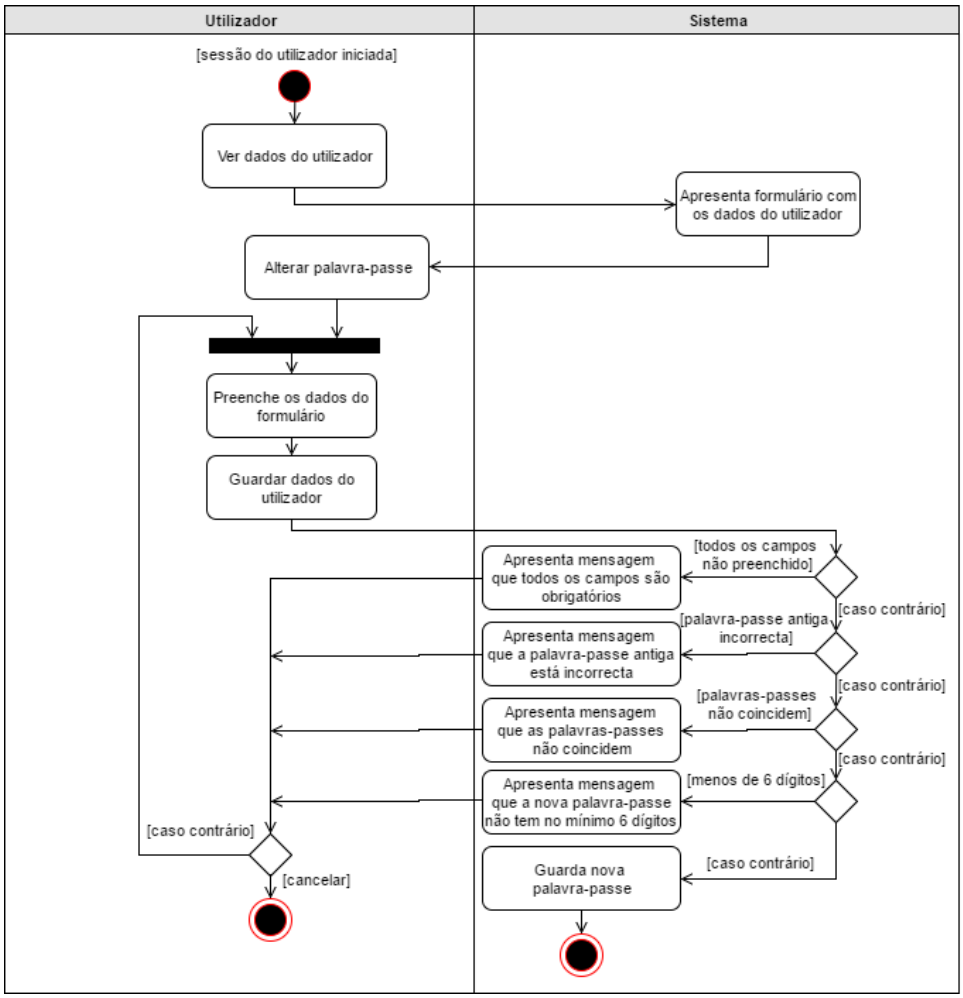


Figura 88 - Diagrama de atividade para alterar palavra-passe do utilizador

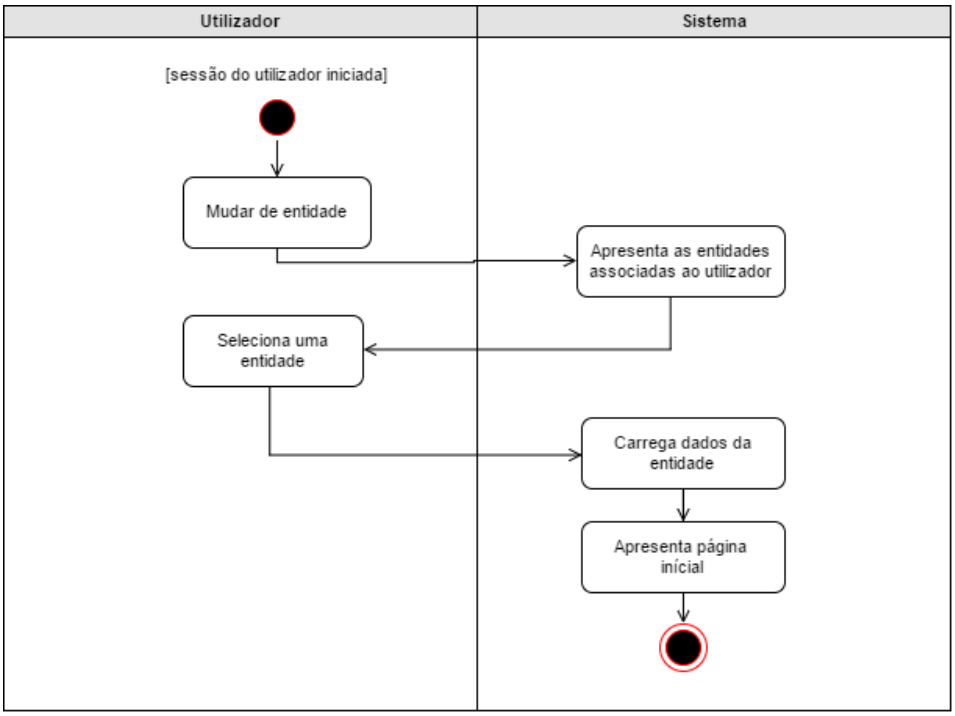


Figura 89 - Diagrama de atividade para selecionar entidade

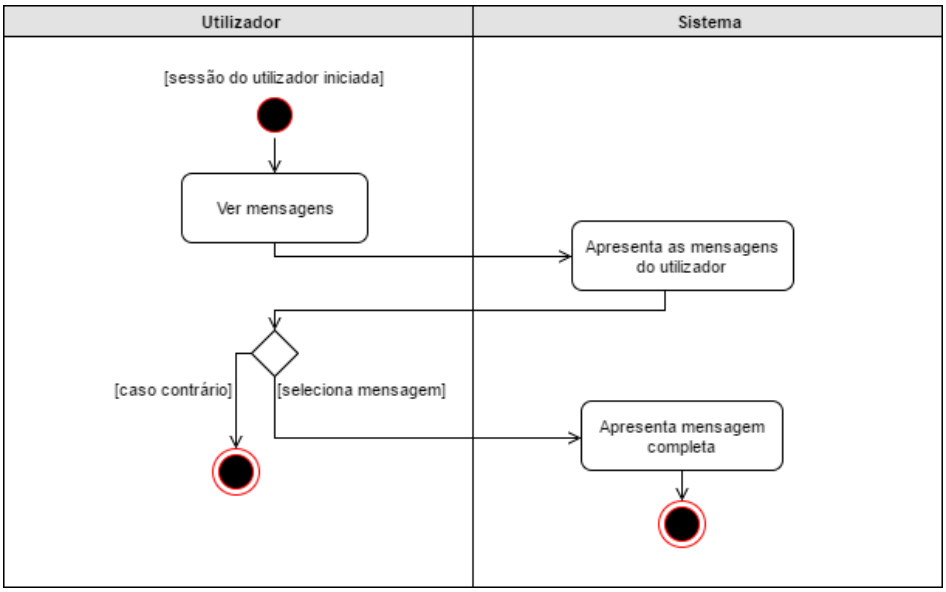


Figura 90 - Diagrama de atividade para ver mensagens do utilizador

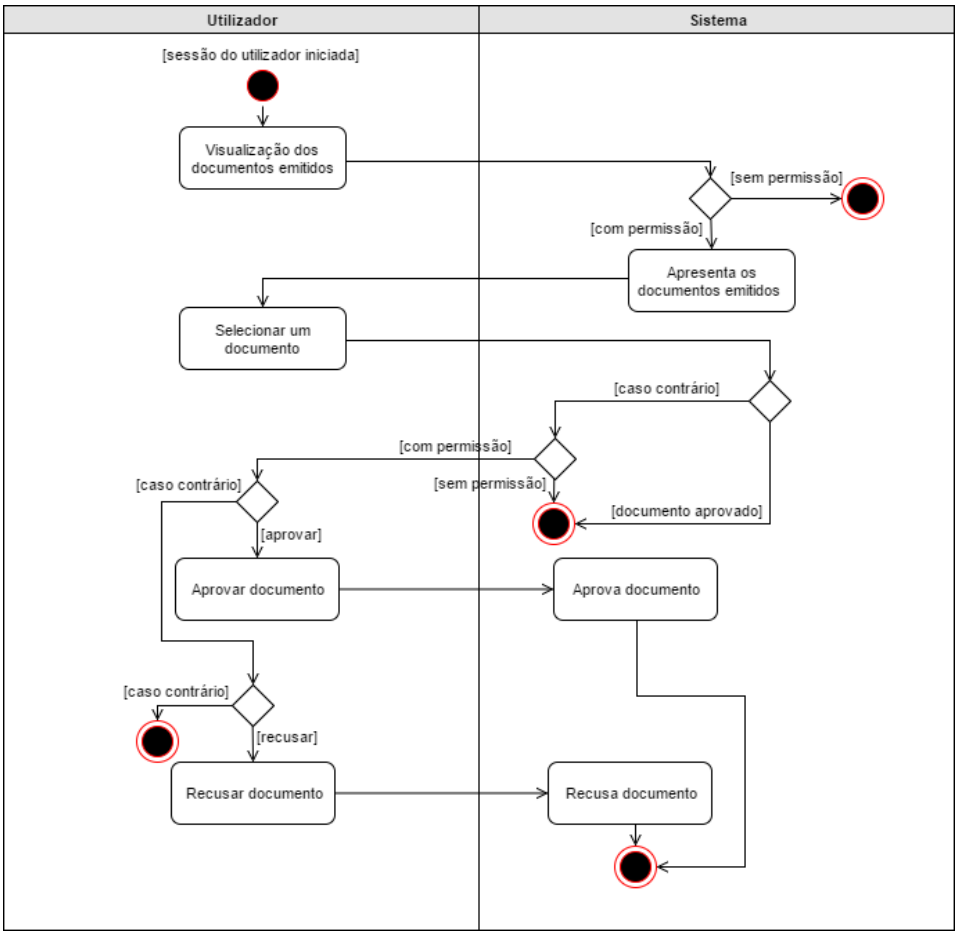


Figura 91 - Diagrama de atividade para aprovar ou recusar documentos recebidos

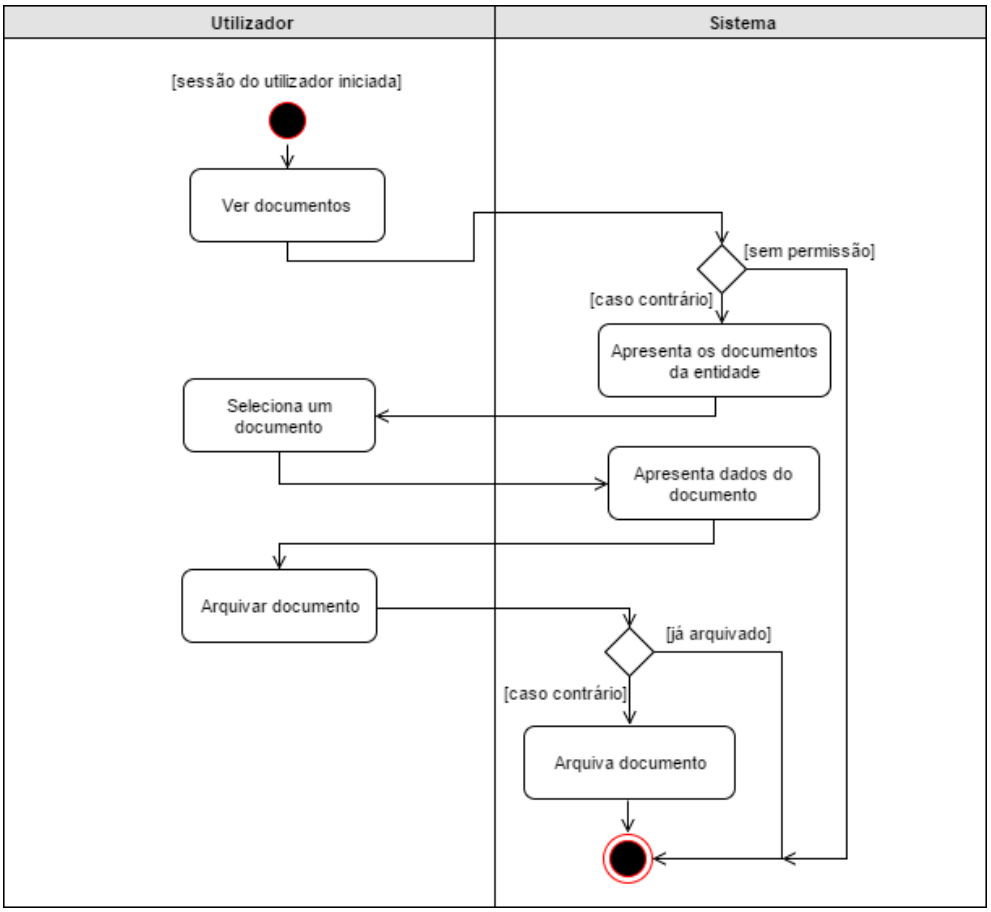


Figura 92 - Diagrama de atividade para arquivar documentos

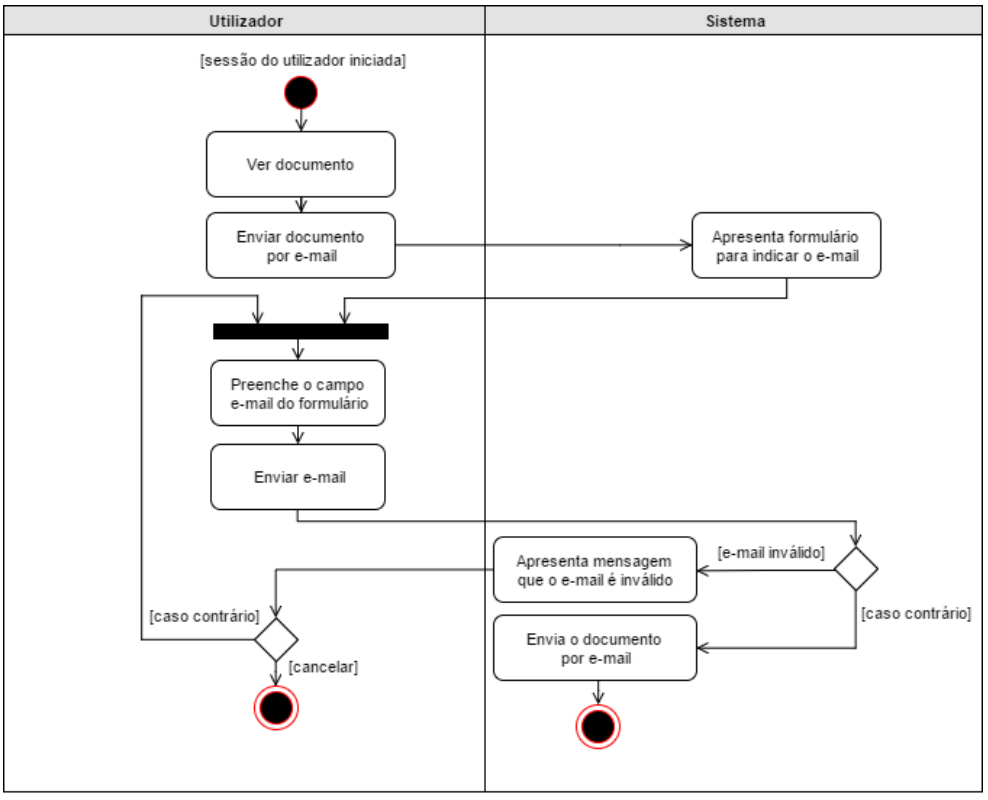


Figura 93 - Diagrama de atividade para enviar documentos por e-mail

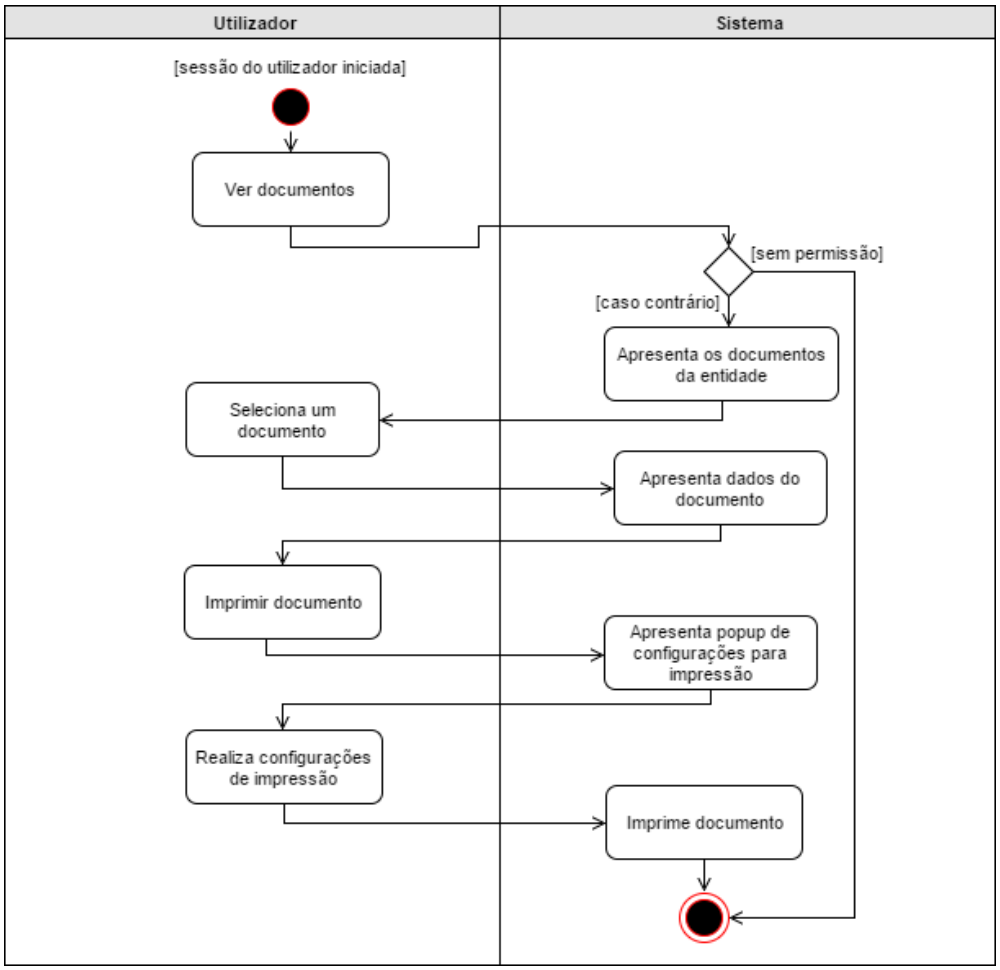


Figura 94 - Diagrama de atividade para imprimir documentos

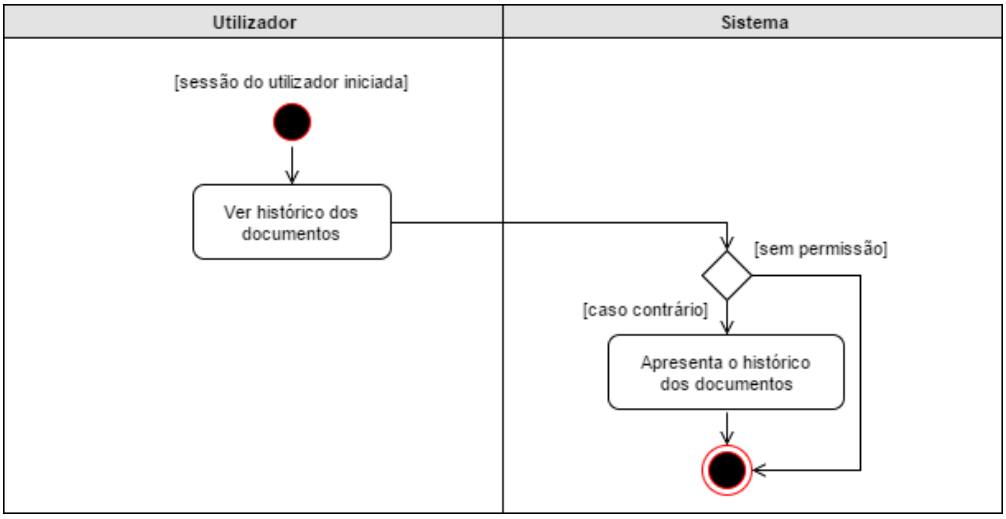


Figura 95 - Diagrama de atividade para ver histórico dos documentos

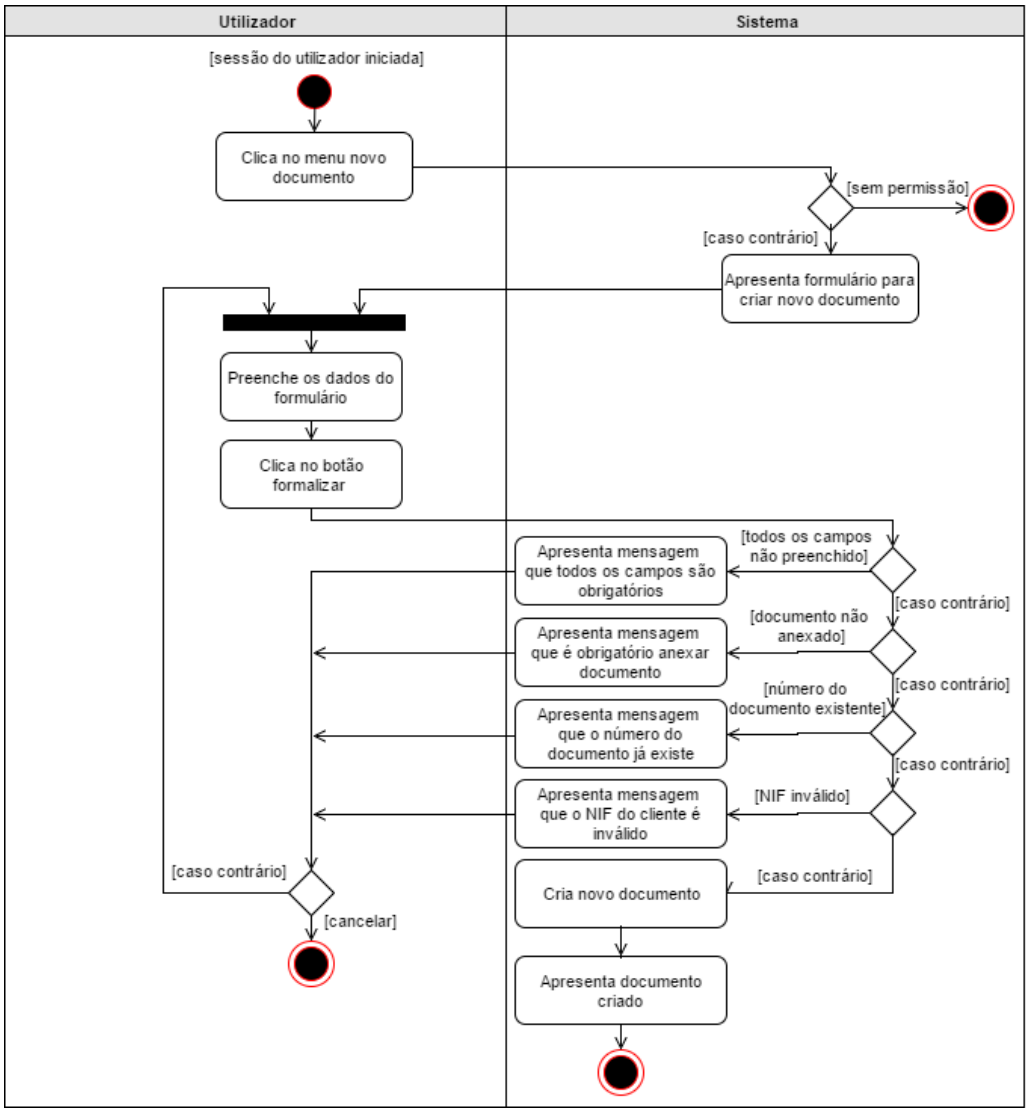


Figura 96 - Diagrama de atividade para criar novo documento

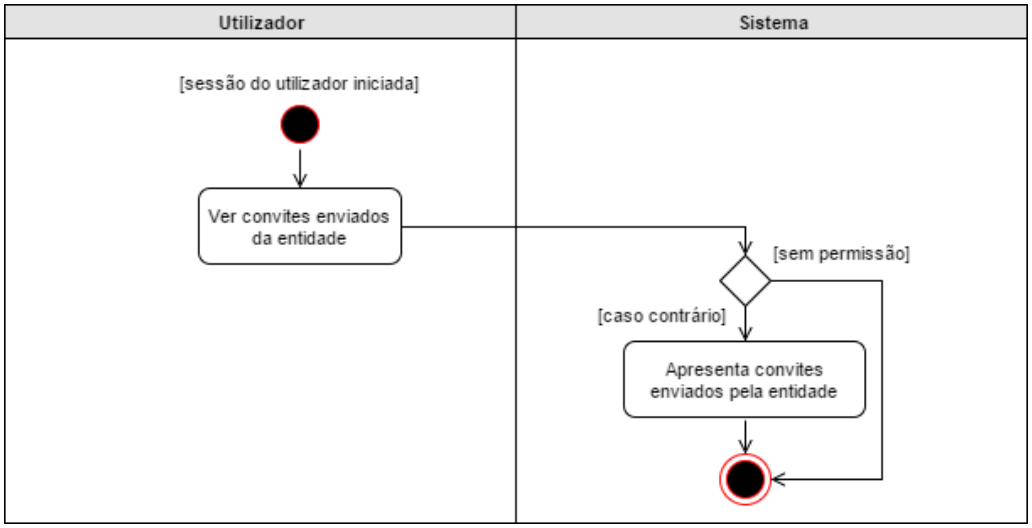


Figura 97 - Diagrama de atividade para ver convites enviados

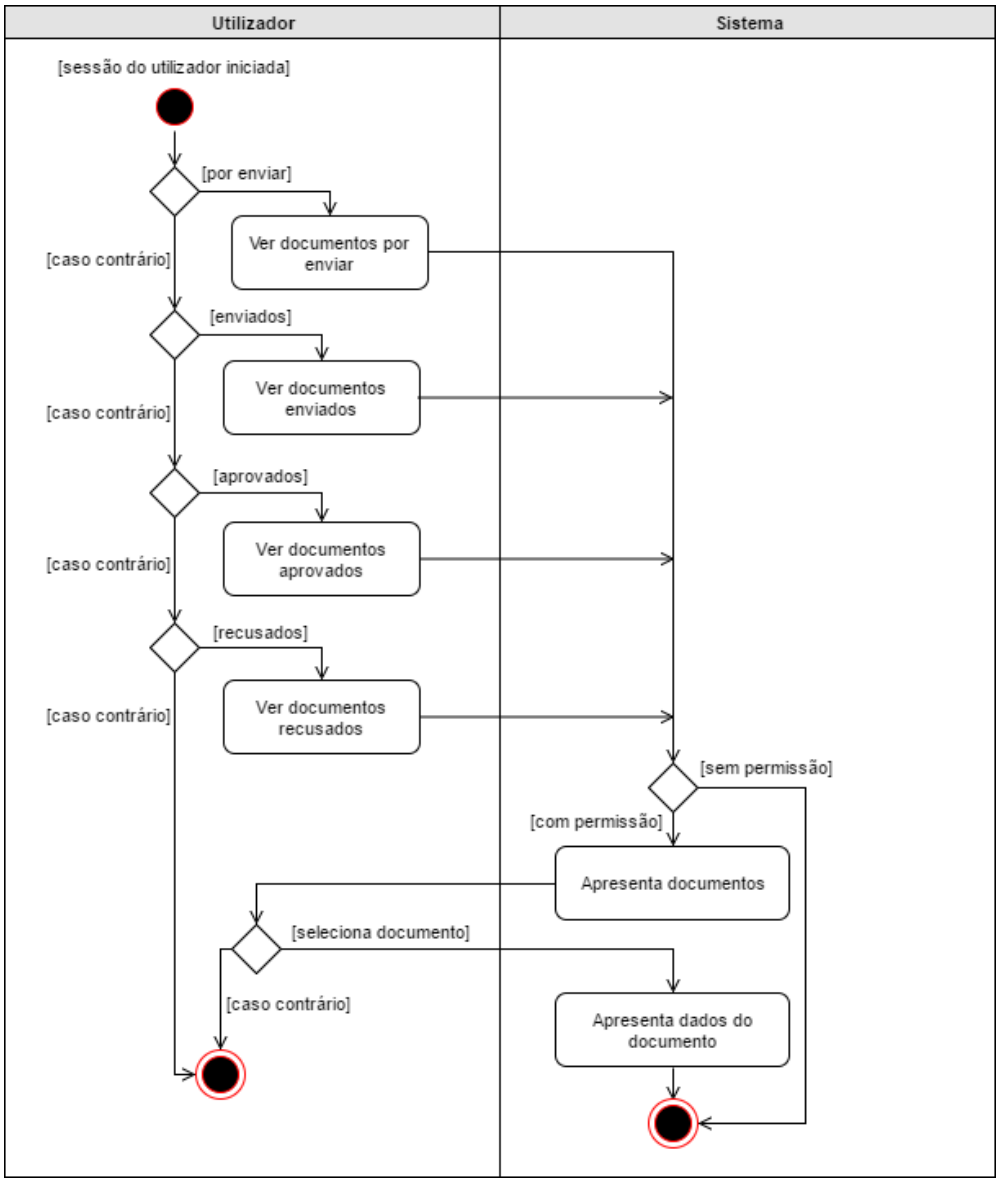


Figura 98 - Diagrama de atividade para ver documentos emitidos

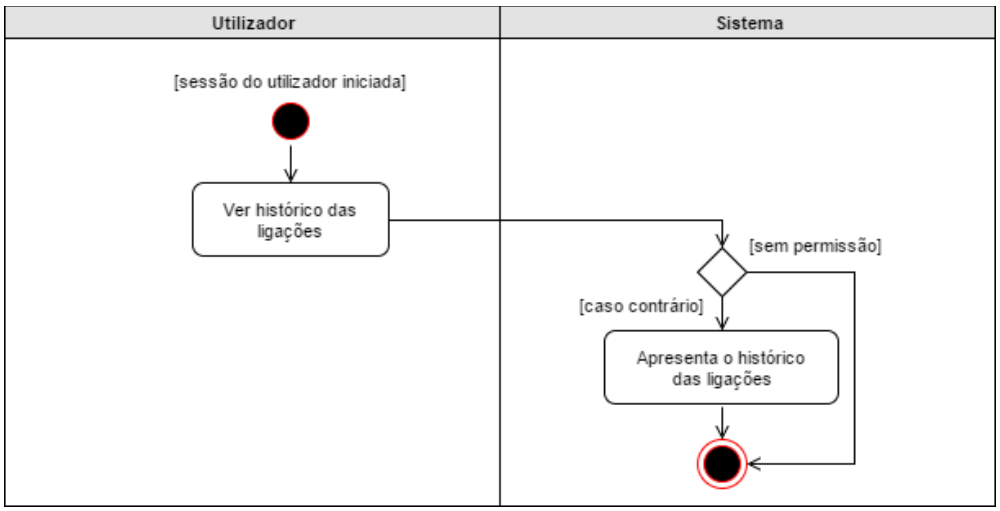


Figura 99 - Diagrama de atividade para ver histórico das ligações

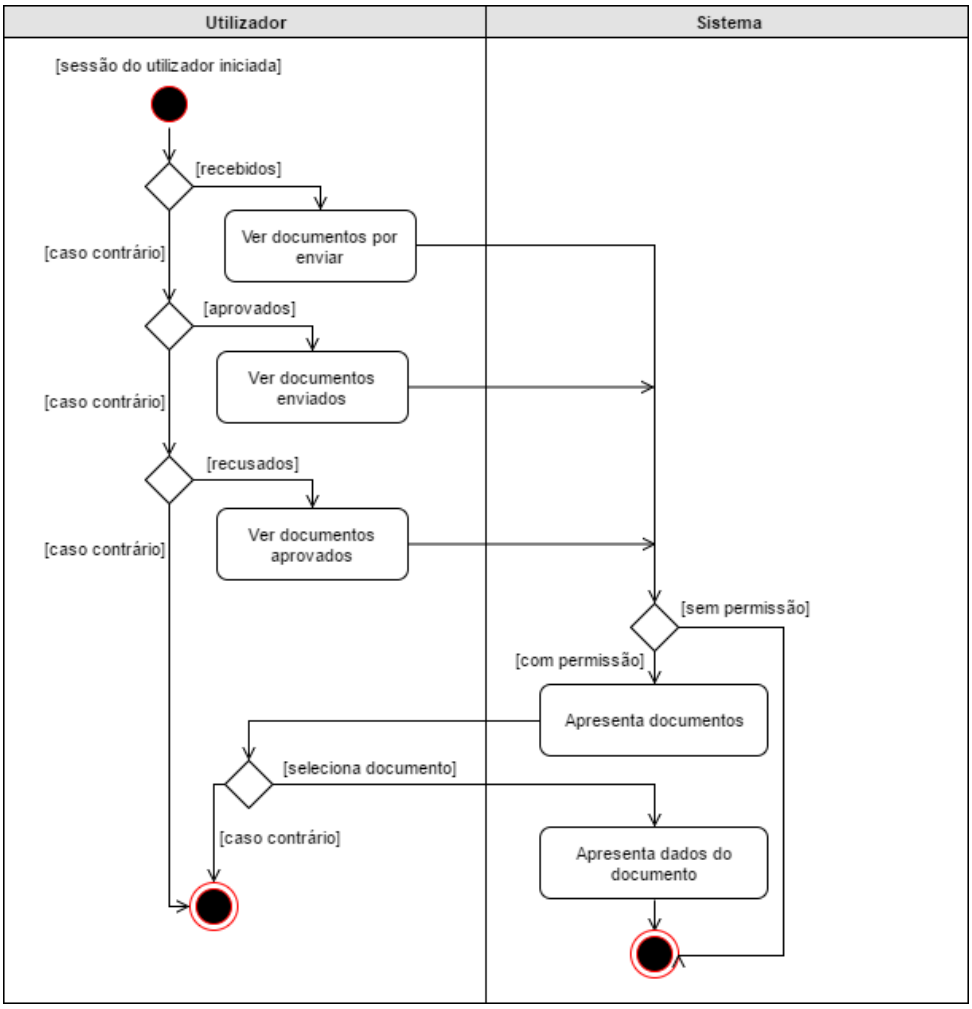


Figura 100 - Diagrama de atividade para ver documentos recebidos

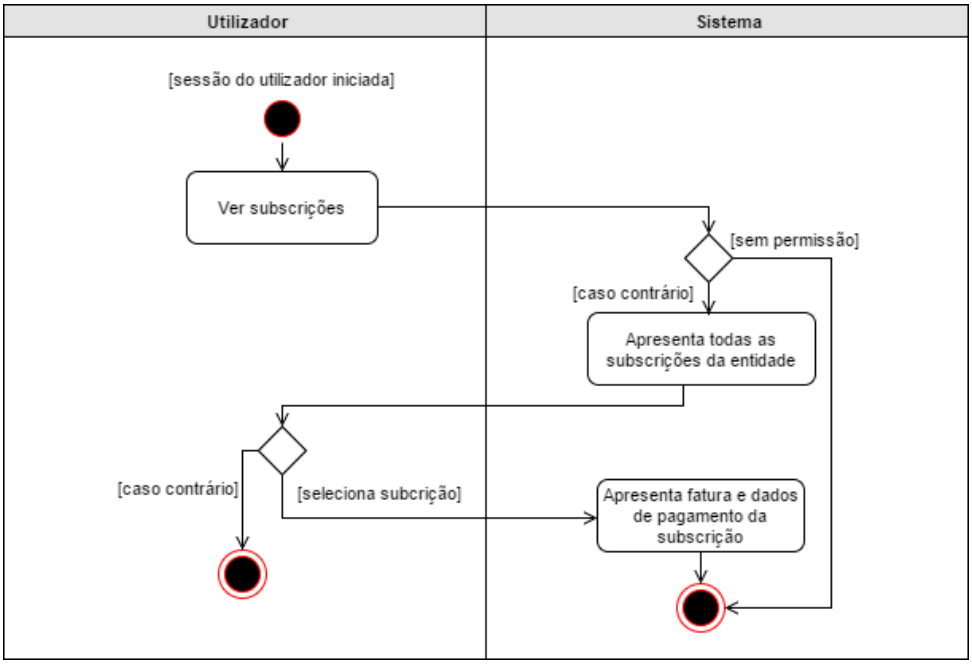


Figura 101 - Diagrama de atividade para ver subscrições

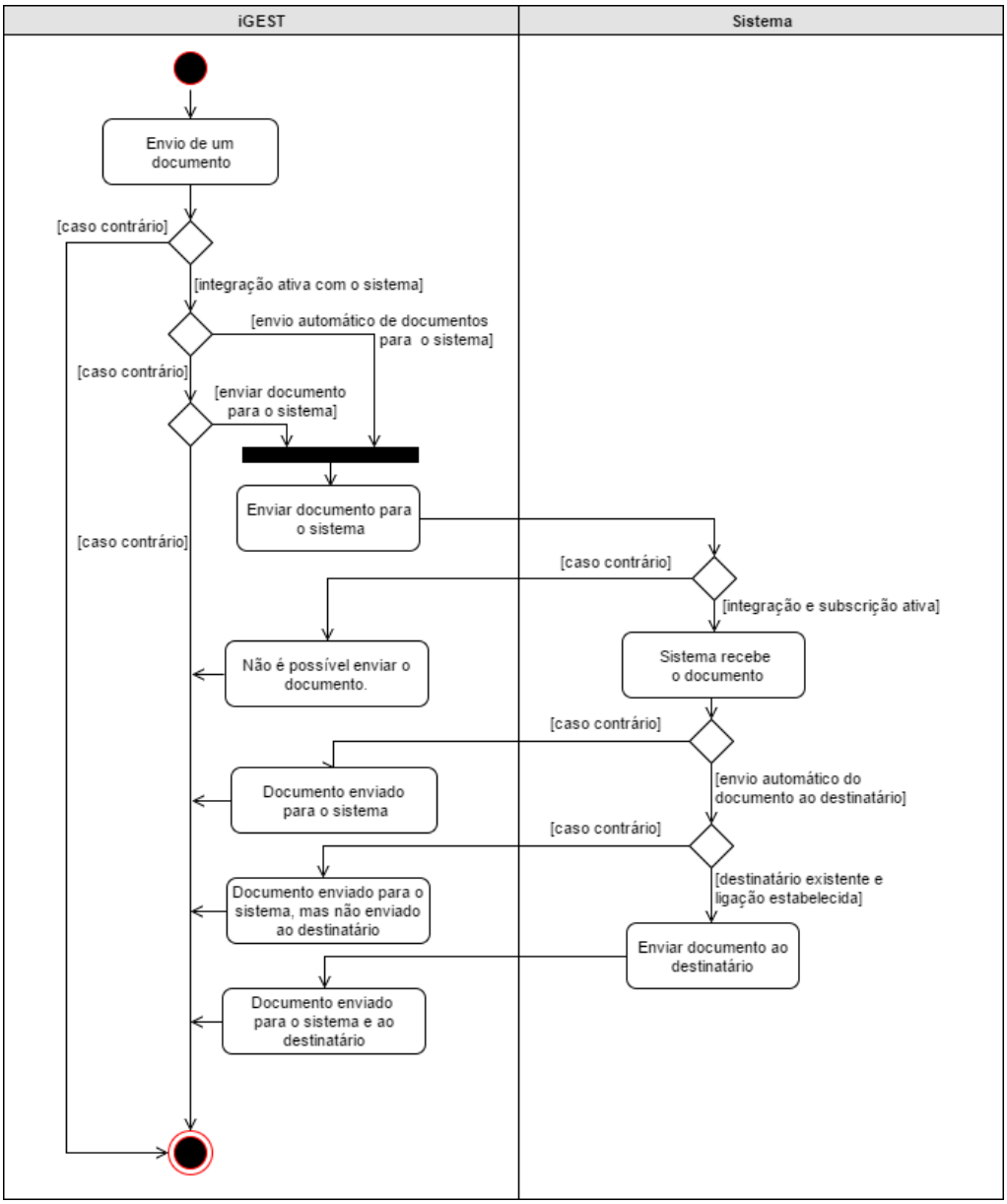


Figura 102 - Diagrama de atividade para enviar documento do iGEST para o sistema

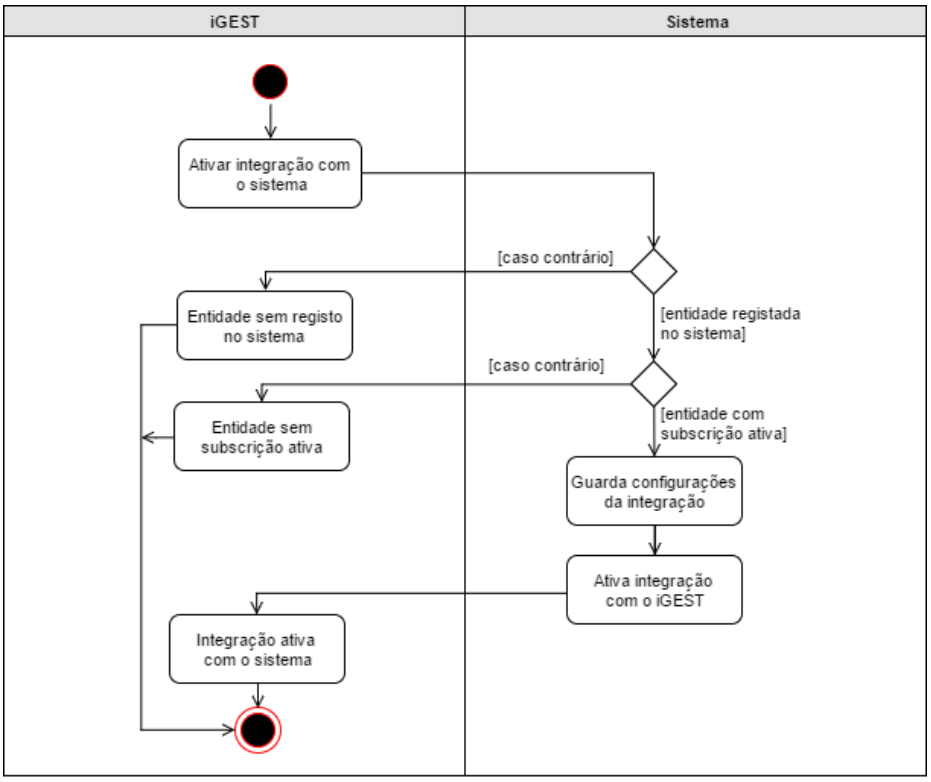


Figura 103 - Diagrama de atividade para integração do iGEST com o sistema

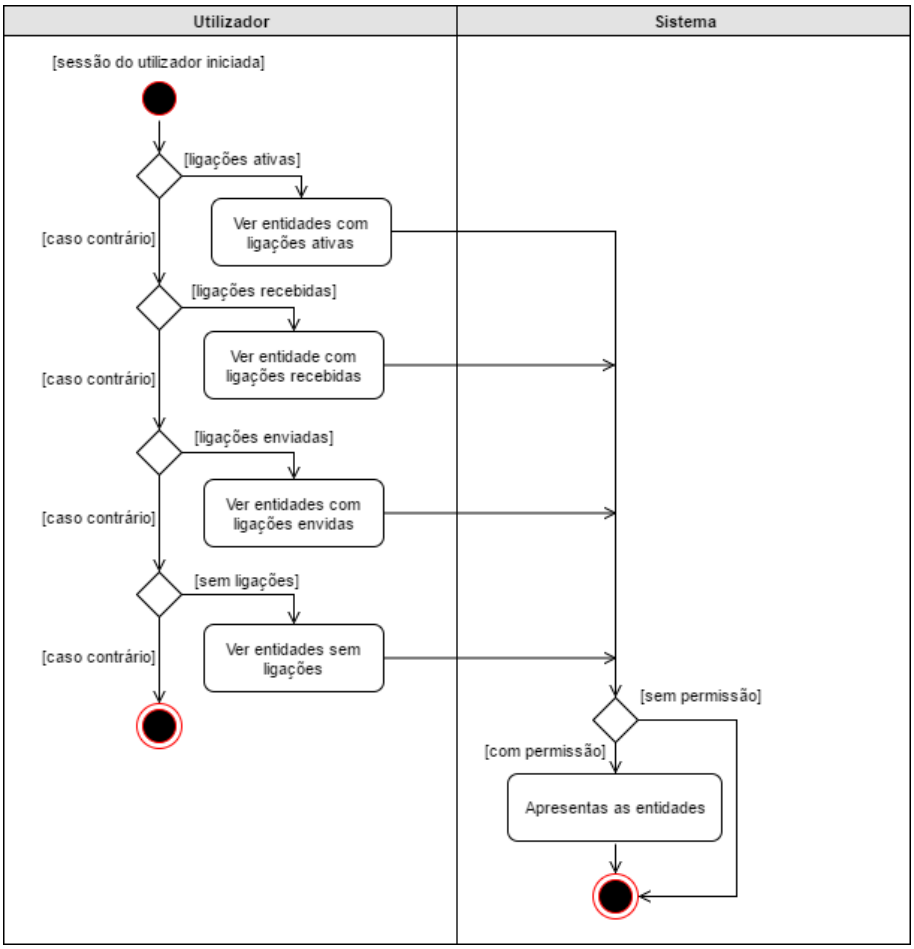


Figura 104 - Diagrama de atividade para ver ligações

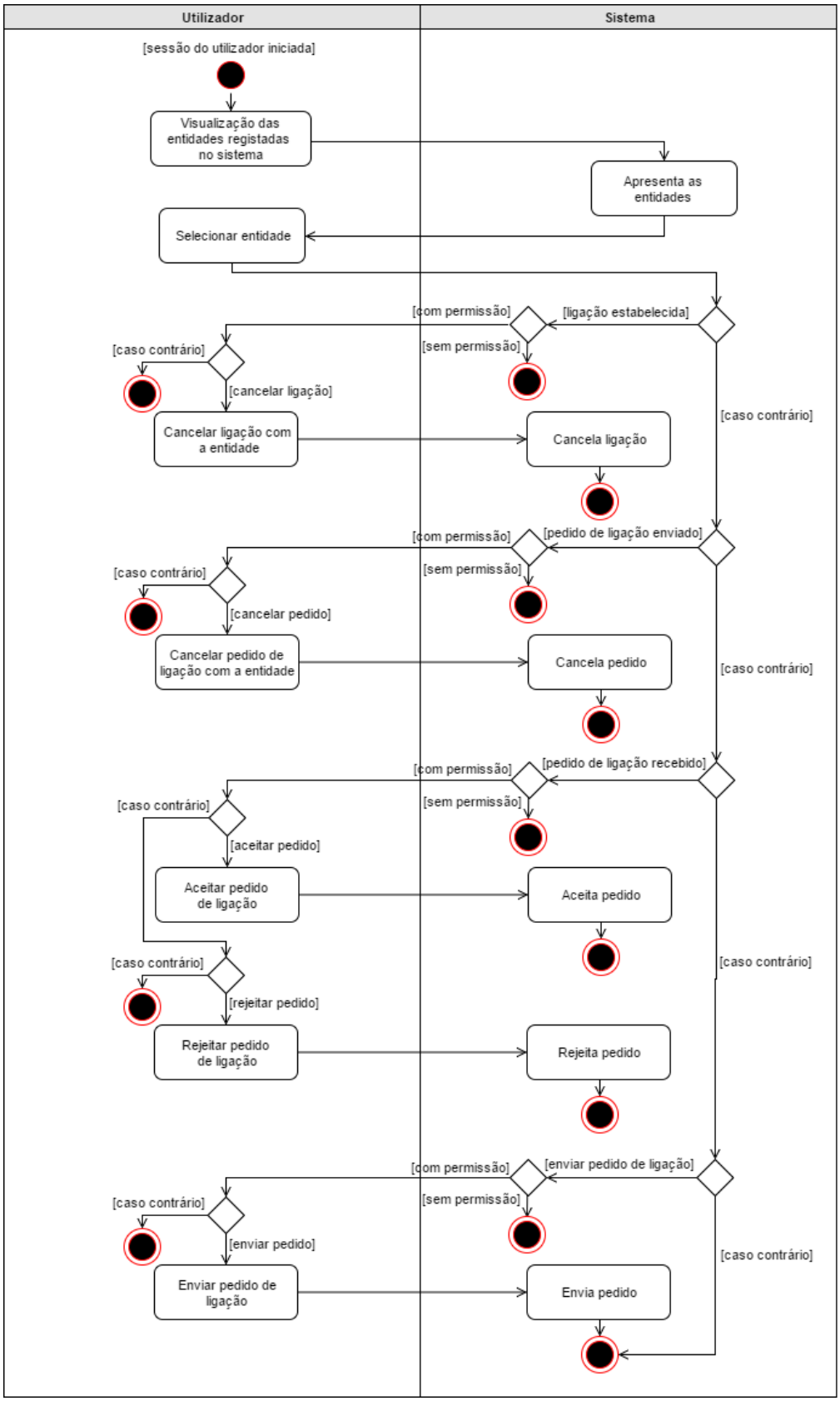


Figura 105 - Diagrama de atividade para criar, alterar, aceitar ou rejeitar ligações

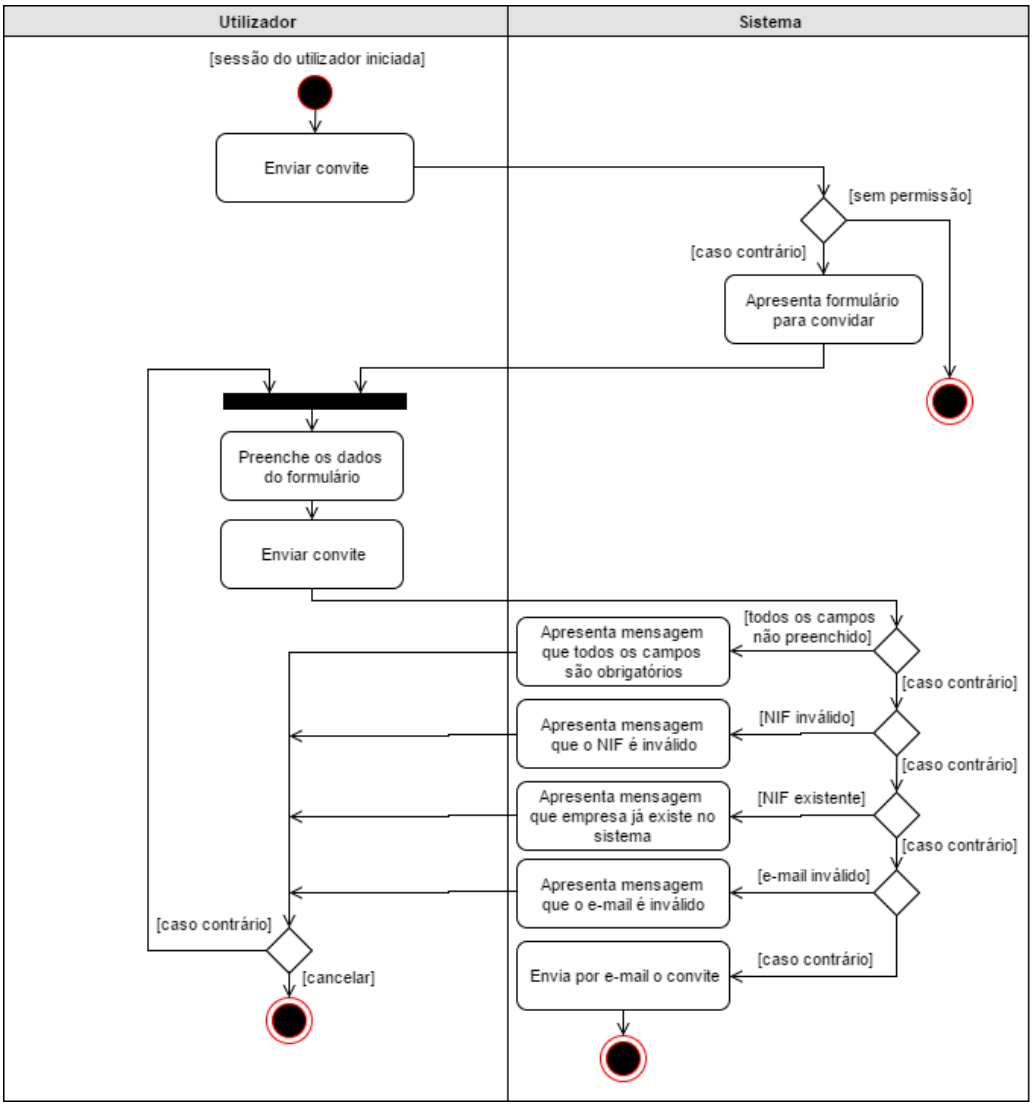


Figura 106 - Diagrama de atividade para enviar convites

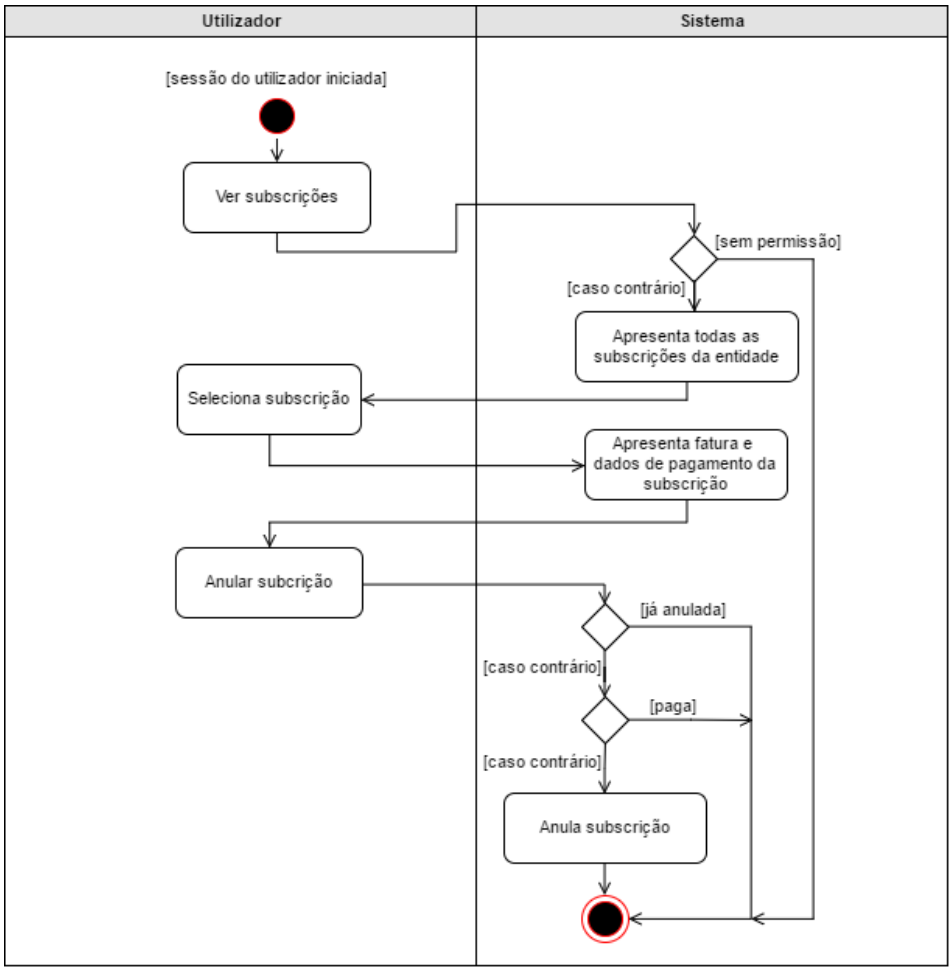


Figura 107 - Diagrama de atividade para anular subscrição

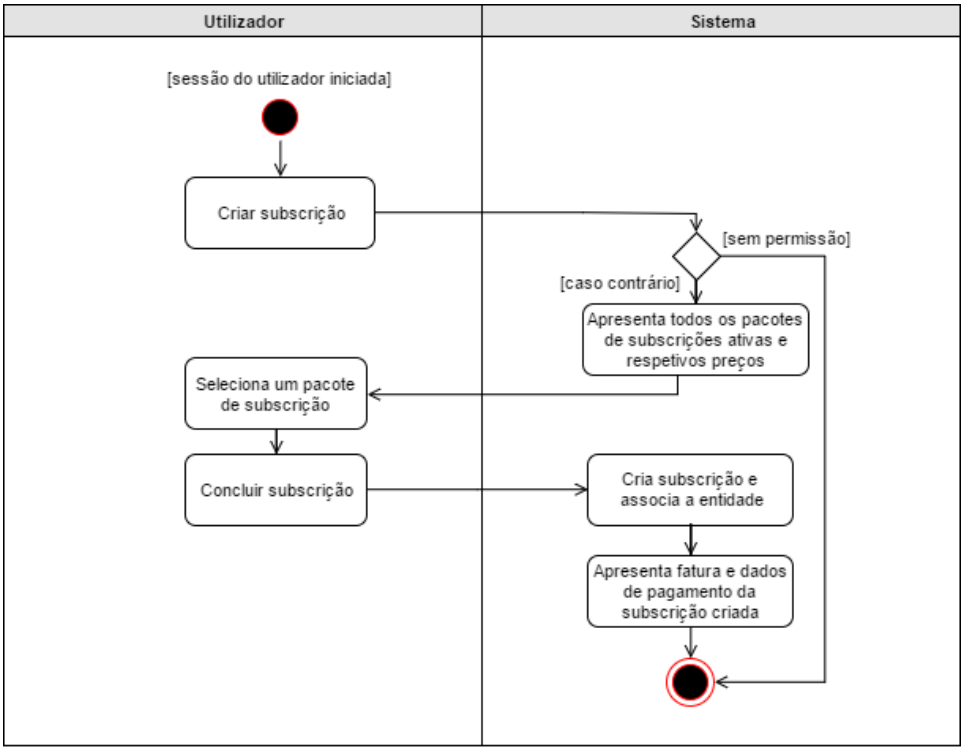


Figura 108 - Diagrama de atividade para criar subscrição

10.3 Anexo C – Base de Dados

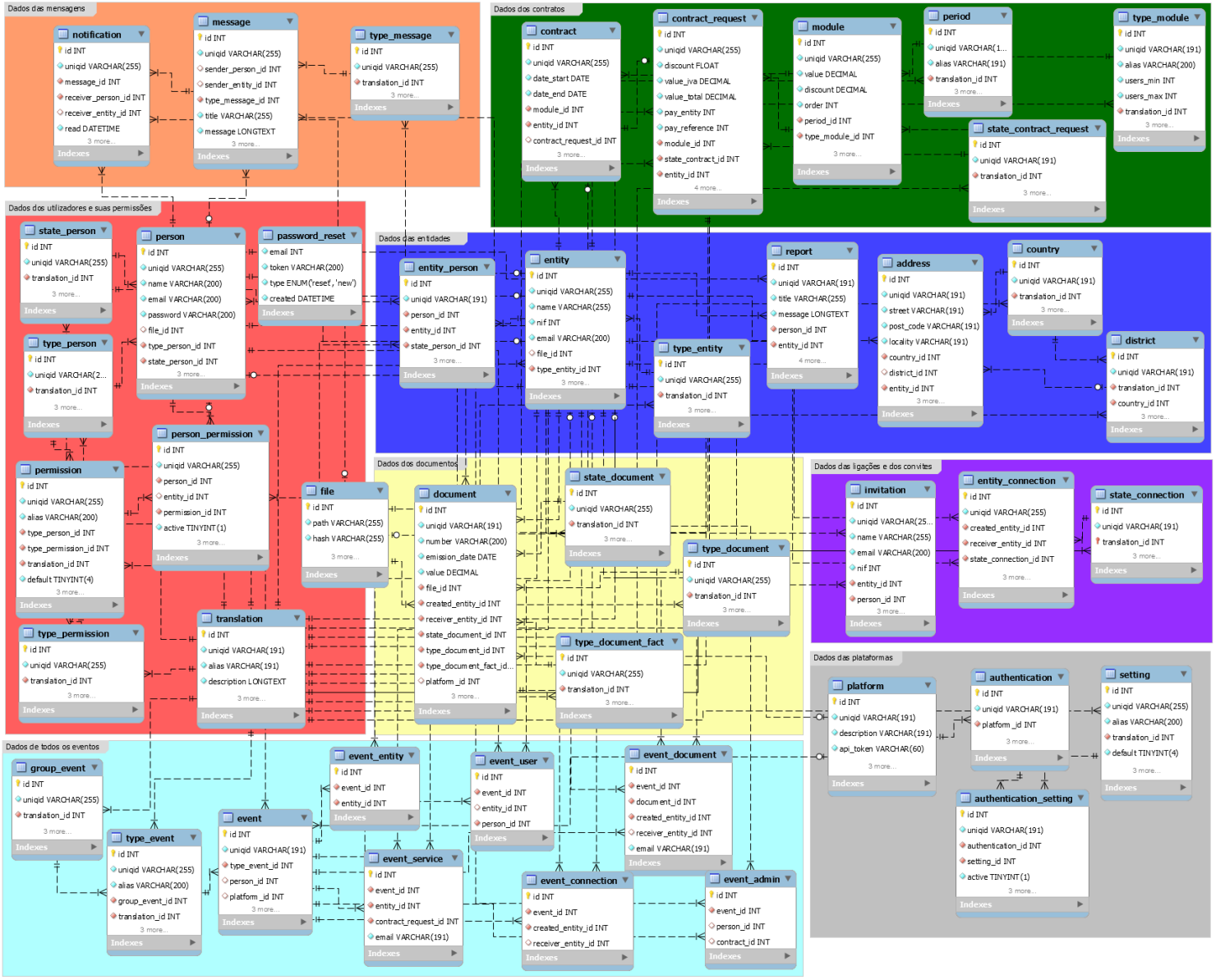


Figura 109 - Base de dados completa do sistema

10.4 Anexo D – Cenários

Cenário 1

O utilizador André da entidade Importação e Exportação Lda, emite uma fatura no sistema de faturação iGEST e o cliente Lupa Camiões Lda pretende receber a fatura eletronicamente. Para tal o utilizador André, acede ao novo sistema e pesquisa o documento emitido com o cliente Lupa Camiões Lda e visualiza o documento. Depois verifica se os dados estão todos corretos e envia o documento ao cliente. Por fim depois de enviar o documento, o André termina sessão.

Tarefas:

- Iniciar sessão no sistema.
- Pesquisar pelo documento emitido número 1 que ainda não foi enviado.
- Visualizar o documento número 1.
- Enviar documento ao cliente.
- Terminar sessão.

Cenário 2

O utilizador André com sessão iniciada, recebe uma notificação e verifica que a entidade Importação e Exportação Lda recebeu um documento. O utilizador André pesquisa pelo documento através do seu número e visualiza o documento. Depois verifica que o documento é engano e recusa o documento. O utilizador André ainda altera a sua palavra-passe, para manter a sua conta em segurança.

Tarefas:

- Iniciar sessão no sistema.
- Visualizar notificações recebidas.
- Pesquisar pelo documento recebido número 1 que ainda não foi aceite.
- Visualizar o documento número 1.
- Recusar documento.
- Alterar palavra-passe do utilizador.

Cenário 3

Ao navegar no sistema o utilizador André verifica que tem um documento emitido por enviar ao cliente. Como tal, visualiza o documento e verifica que o cliente do documento não se encontra registado no sistema e envia um convite. Devido a essa situação, o utilizador lembra-se da nova parceria com a entidade Empresa das Águas e antecipa-se e envia também o convite a essa entidade, para que no futuro seja possível partilhar documentos eletrónicos.

Tarefas:

- Iniciar sessão no sistema.
- Visualizar documentos emitidos por enviar
- Pesquisar pelo documento emitido número 3 que ainda não foi enviado.
- Visualizar o documento número 3.
- Enviar convite ao cliente do documento número 3.
- Enviar convite a entidade Empresa das Águas.

Cenário 4

O utilizador André com sessão iniciada, recebe uma notificação e verifica que recebeu um pedido de ligação da entidade Escritodecor - Mobiliário, Lda para a troca de documentos eletrónicos. Ele acede aos pedidos de ligação e aceita o pedido de ligação da entidade Escritodecor - Mobiliário, Lda. Aproveitando que está nas ligações, o André verifica as ligações das sua entidade e cancela as ligações com as entidades que nunca partilhou documentos.

Tarefas:

- Iniciar sessão no sistema.
- Visualizar notificações recebidas.
- Visualizar pedidos de ligação recebidos.
- Pesquisar o pedido de ligação da entidade Escritodecor - Mobiliário, Lda.
- Aceitar o pedido de ligação da entidade Escritodecor - Mobiliário, Lda.
- Pesquisar os pedidos de ligação ativos.
- Cancelar a ligação da entidade Mediação de Seguros, Lda.

Cenário 5

A entidade André & Filhos Lda contratou dois novos empregado Maria e Roberto e pretende adicionar esses dois novos empregados no sistema. Como os novos empregados ainda não são de confiança, o patrão da entidade decidiu apenas atribuir permissão de visualizar os documentos emitidos e recebidos. Por isso o utilizador André acede ao sistema, e nas configurações da entidade cria dois utilizadores com os dados dos novos empregados e apenas atribui permissão de visualizar os documentos emitidos e recebidos.

Tarefas:

- Iniciar sessão no sistema.
- Visualizar utilizadores da entidade.
- Criar novo utilizador e selecionar as permissões de visualização de documentos emitidos e recebidos.
- Voltar a criar novo utilizador e selecionar as permissões de visualização de documentos emitidos e recebidos.

Cenário 6

A entidade Venda de Carros Lda não utiliza o sistema de faturação iGEST mas pretende criar um novo documento no novo sistema e de seguida enviar ao cliente. Deste modo, o utilizador André inicia sessão no sistema e cria um novo documento indicando as informações do documento e do cliente. Depois de criar o documento, o utilizador pretende enviar o documento ao cliente mas ainda existe ligação entre as duas entidades. Por isso, o utilizador apenas envia um pedido de ligação a entidade do documento e também imprimir o documento.

Tarefas:

- Iniciar sessão no sistema.
- Criar novo documento.
- Enviar pedido de ligação.
- Imprimir documento.

Cenário 7

O utilizador Célio é patrão da empresa Mota & Carros Lda e pretende visualizar quais dos seus utilizadores tem aceitado e/ou recusado documentos recebidos para seu controlo. Para isso o utilizador Célio acede ao sistema e visualiza o histórico de documentos recebidos efetuando várias pesquisas. Aproveitando dessa situação, o utilizador lembra-se que o logótipo da entidade sofreu algumas alterações e por isso acede as configurações da entidade e altera o logótipo da entidade.

Tarefas:

- Iniciar sessão no sistema.
- Visualizar histórico de documentos recebidos.
- Pesquisar histórico de documento emitidos.
- Ver configurações da entidade.
- Alterar logótipo da entidade.

Cenário 8

O administrador do sistema João foi contactado pela entidade com o NIF 503123543 que tentou pesquisar pelos documentos recebidos no estado aceite da entidade Mota & Carros Lda e não obteve qualquer resultado, quando deveria existir pelo menos um documento. Logo o João inicia sessão no sistema e pesquisa pela entidade com o NIF 503123543 e acede a essa entidade. Depois de aceder a entidade, o João faz o mesmo que o utilizador da entidade e verifica que existe mesmo um problema com o sistema. Por fim reporta o problema ao departamento de desenvolvimento.

Tarefas:

- Iniciar sessão no sistema como administrador.
- Visualizar as entidades.
- Pesquisar pela entidade com o NIF 503123543.
- Ver dados da entidade com o NIF 503123543.
- Entrar na entidade como um utilizador normal.
- Ver documentos recebidos que foram aceites.
- Pesquisar os documentos da entidade Mota & Carros Lda.

Cenário 9

O administrador do sistema João tem uma nova colega de trabalho e pretende criar uma nova conta de administrador para a sua nova colega. Os novos administradores inicialmente devem ter apenas permissões de gerir as entidades e os utilizadores. Para isso o João acede ao sistema e cria um novo administrador com as permissões indicadas anteriormente. O João aproveita a situação e desativa o administrador José André Silva.

Tarefas:

- Iniciar sessão no sistema como administrador.
- Criar novo administrador e selecionar as permissões de gestão das entidades e dos utilizadores.
- Visualizar os administradores.
- Pesquisar pelo administrador José André Silva e desativar a sua conta.

Cenário 10

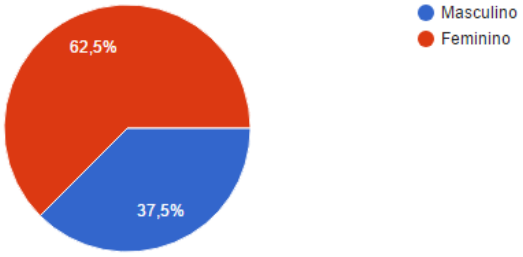
O administrador do sistema João pretende saber quais as entidades que terminaram o contrato e não realizaram a sua renovação, pois precisa contacta-las para saber o motivo da não renovação do contrato. Desse modo, o João inicia sessão no sistema e pesquisa as entidades que terminaram o contrato e não renovaram. Depois de contactar as entidades o João decide oferecer um contrato gratuito de 30 dias as entidades que não renovaram contrato.

Tarefas:

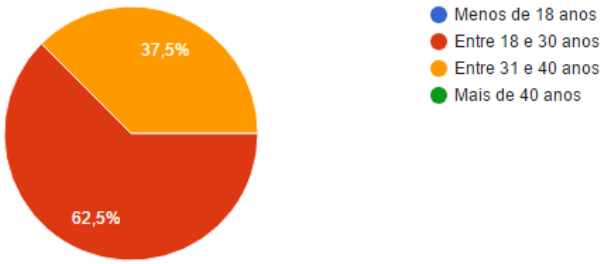
- Iniciar sessão no sistema como administrador.
- Visualizar contratos que terminaram e não foram renovados.
- Pesquisar a 1 entidade.
- Atribuir contratos gratuitos a 1 entidade.
- Pesquisar a 2 entidade.
- Atribuir contratos gratuitos a 2 entidade.

10.5 Anexo E – Questionário e Resultados

Qual o seu género ? (8 respostas)

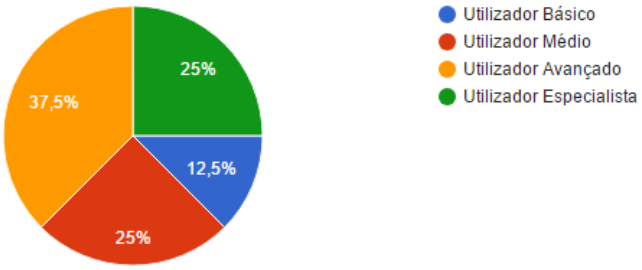


Qual a sua idade? (8 respostas)

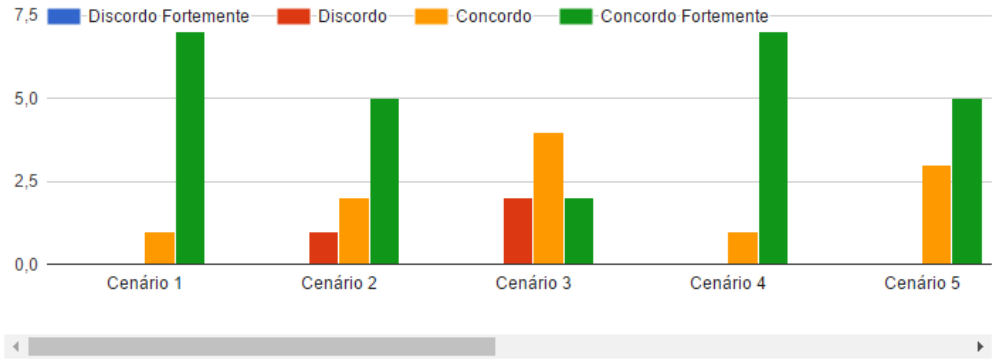


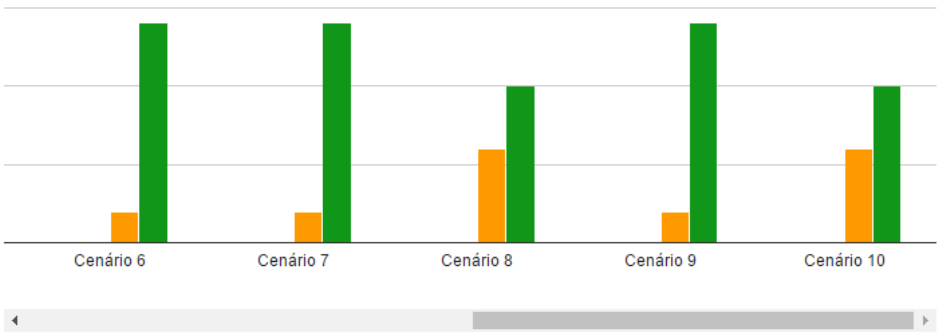
Em geral, como classifica o seus conhecimentos informáticos a nível de utilizador ?

(8 respostas)

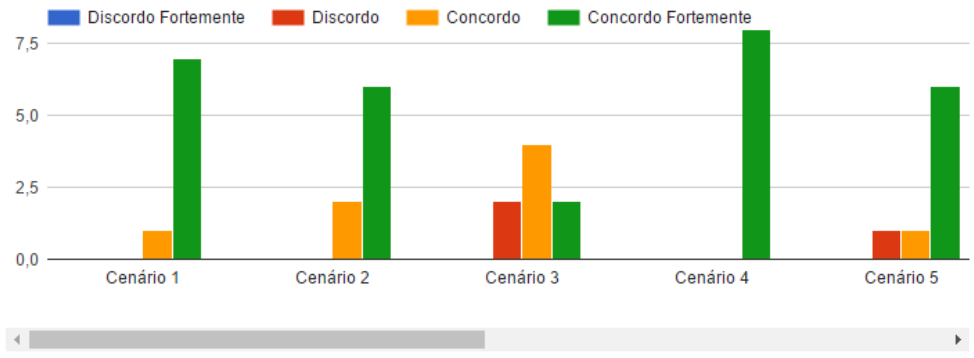


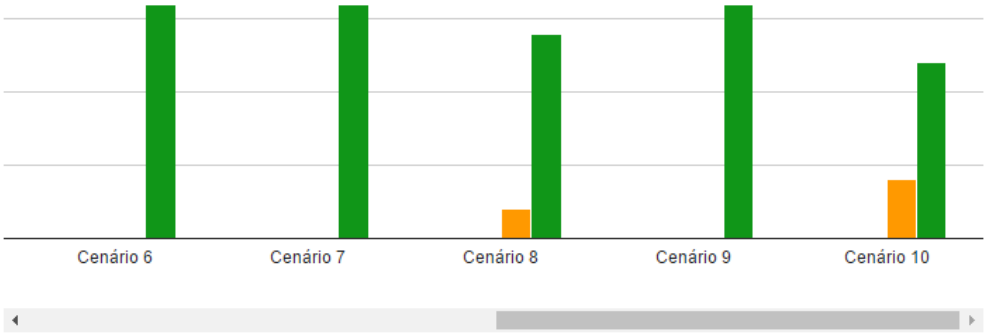
Em geral, estou satisfeito com a facilidade de realização dos cenários.



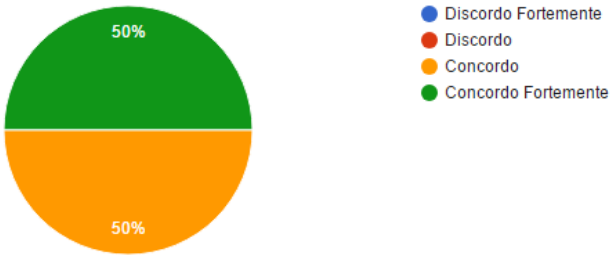


Em geral, estou satisfeito com o tempo de realização dos cenários.

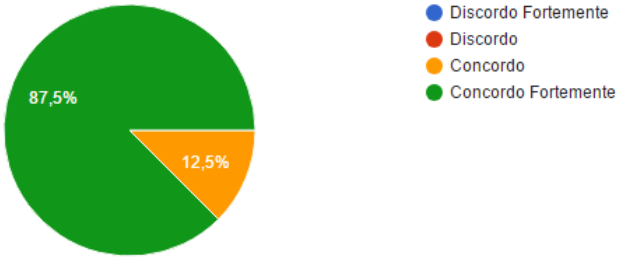




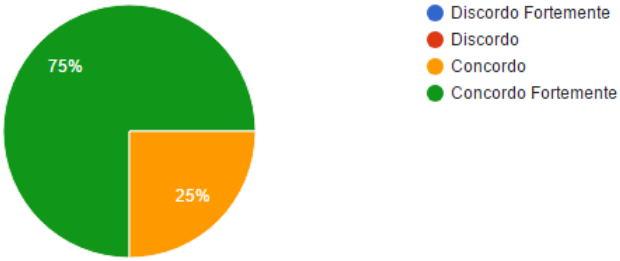
O sistema é simples de utilizar. (8 respostas)



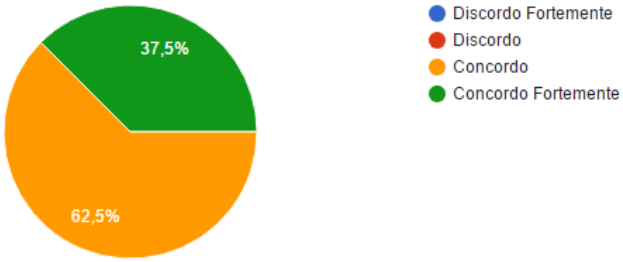
É fácil aprender a utilizar o sistema. (8 respostas)



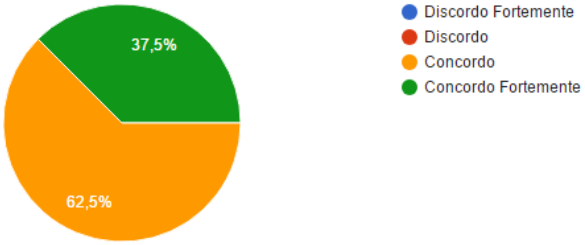
Senti-me confortável utilizar o sistema. (8 respostas)



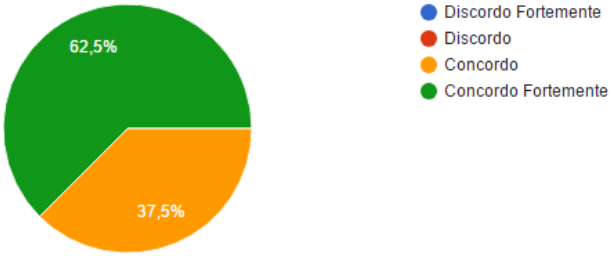
É fácil encontrar as informações que eu preciso. (8 respostas)



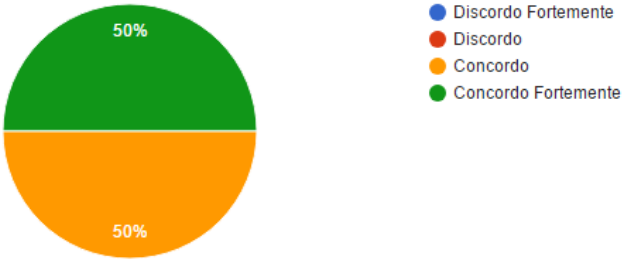
A organização da informação no ecrã do sistema é clara. (8 respostas)



A interface do sistema é agradável. (8 respostas)



Em geral, estou satisfeito com o sistema. (8 respostas)



Observações ou melhorais de usabilidade sobre o sistema. (2 respostas)

nada a referir
Em geral, está muito bom.

10.6 Anexo F – Tempos e Erros dos utilizadores

Na tabela 15 encontra-se os tempos que cada utilizador teve na realização das tarefas associadas aos cenários.

Tabela 18 - Tempos dos utilizadores

Cenário	Pessoa 1	Pessoa 2	Pessoa 3	Pessoa 4	Pessoa 5	Pessoa 6	Pessoa 7	Pessoa 8
1	00:45:78s	00:35:98s	00:32:77s	00:55:55s	00:54:02s	00:27:50s	00:59:58s	01:07:03s
2	00:52:02s	00:47:83s	00:28:95s	00:31:83s	00:49:53s	00:54:46s	00:49:17s	01:01:23s
3	01:17:77s	03:06:03s	00:29:93s	00:51:35s	01:20:13s	01:14:37s	01:10:11s	01:11:68s
4	00:30:95s	00:52:62s	00:51:88s	00:34:84s	00:32:14s	00:41:81s	00:58:26s	00:53:52s
5	00:45:97s	00:15:32s	00:33:47s	00:53:49s	00:51:24s	00:46:52s	00:41:22s	00:42:02s
6	00:35:27s	00:19:96s	00:22:40s	00:17:57s	00:13:92s	00:17:67s	00:38:72s	00:29:13s
7	00:23:73s	00:22:03s	00:45:84s	00:28:14s	00:21:52s	00:35:49s	00:28:68s	00:40:17s
8	00:49:88s	00:59:09s	00:54:95s	00:39:70s	00:39:70s	00:56:41s	00:48:37s	00:58:76s
9	00:36:19s	00:39:89s	00:35:71s	00:44:27s	00:44:27s	00:34:35s	00:53:43s	00:52:32s
10	00:52:88s	01:17:96s	01:12:07s	00:59:67s	00:59:67s	00:49:32s	01:01:51s	01:18:88s

E na tabela 16 encontra-se o número de erros de cada utilizador ao realizar as tarefas associadas aos cenários.

Tabela 19 - Número de erros dos utilizadores

Cenário	Pessoa 1	Pessoa 2	Pessoa 3	Pessoa 4	Pessoa 5	Pessoa 6	Pessoa 7	Pessoa 8
1	0	1	0	0	0	0	0	0
2	1	1	0	0	0	3	0	1
3	2	2	0	0	2	2	0	0
4	0	0	1	0	0	0	0	0
5	2	1	1	1	1	0	0	0
6	0	0	0	0	0	0	0	0
7	1	0	1	0	0	1	1	1
8	0	0	1	0	1	1	0	0
9	0	0	0	0	0	0	0	0
10	1	1	2	1	1	1	0	1

10.7 Anexo G – Protótipos

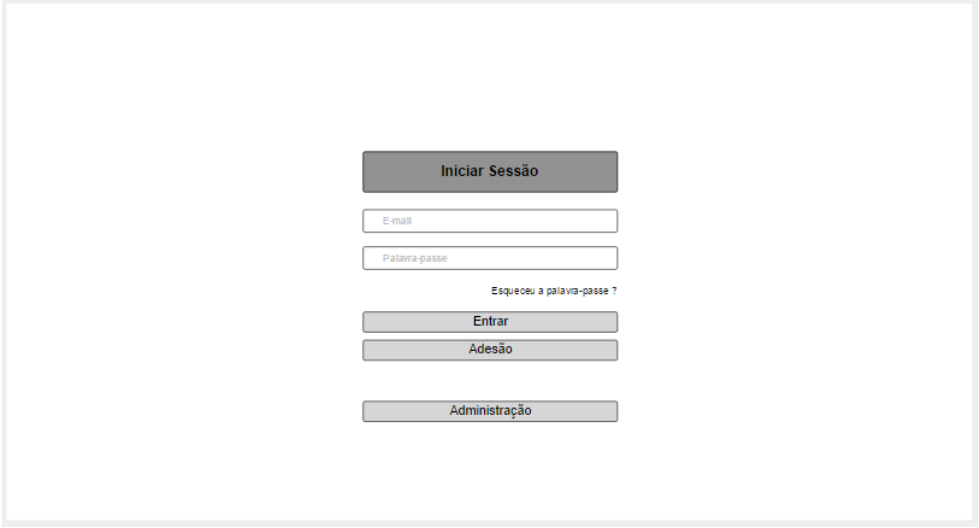


Figura 110 - Página para iniciar sessão

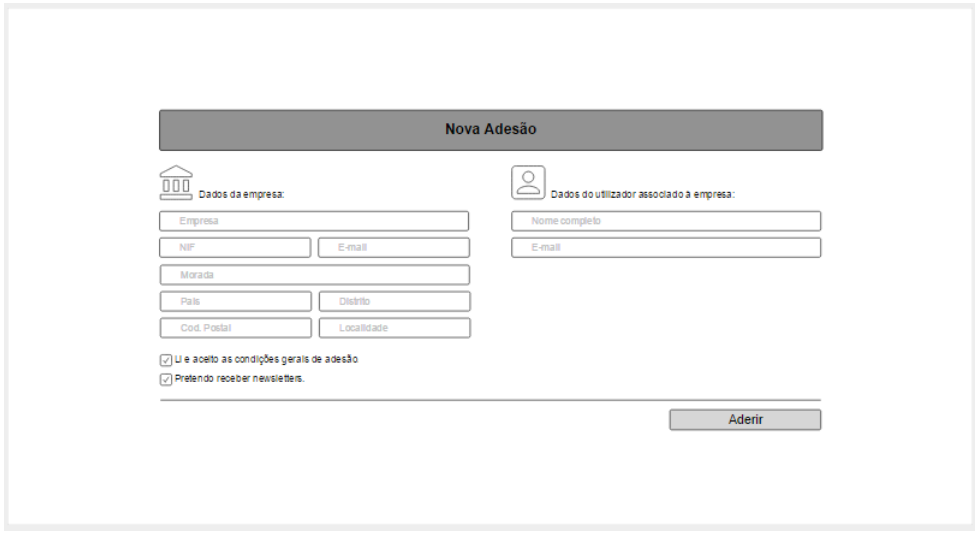


Figura 111 - Página para nova adesão

Figura 112 - Página para recuperar palavra-passe

Figura 113 - Página para inserir nova palavra-passe

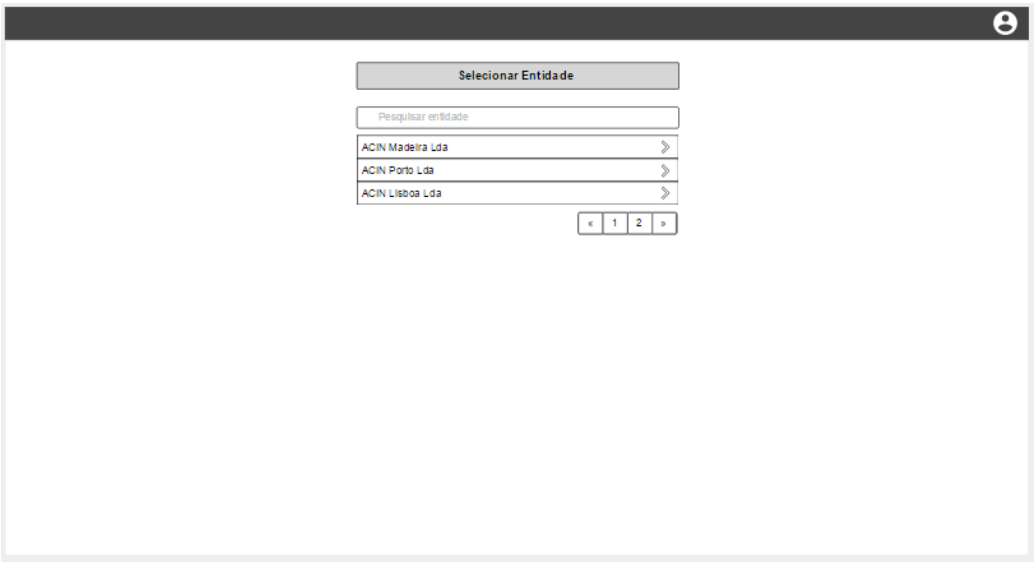


Figura 114 - Página para selecionar entidade

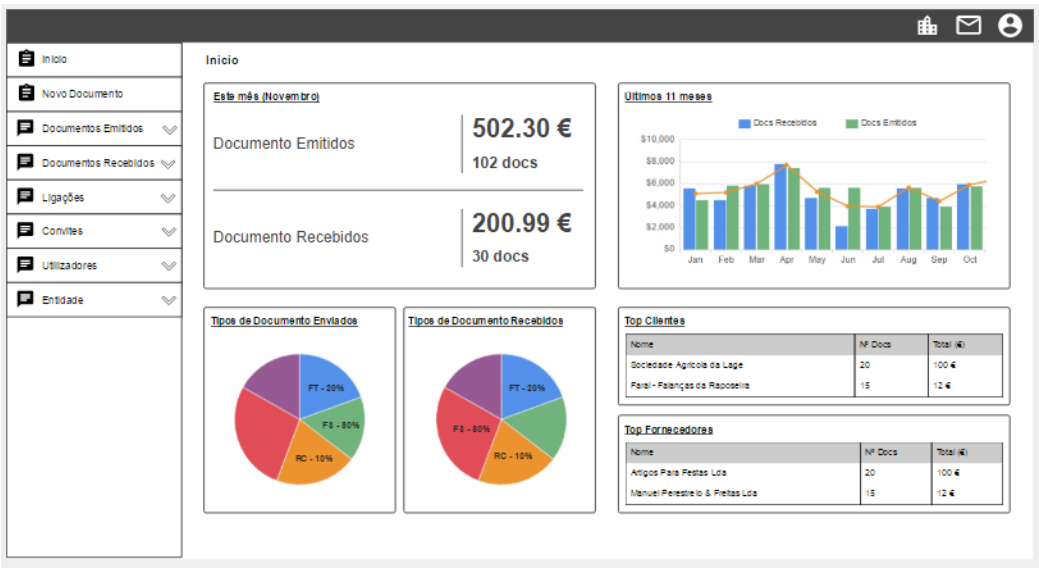


Figura 115 - Página início

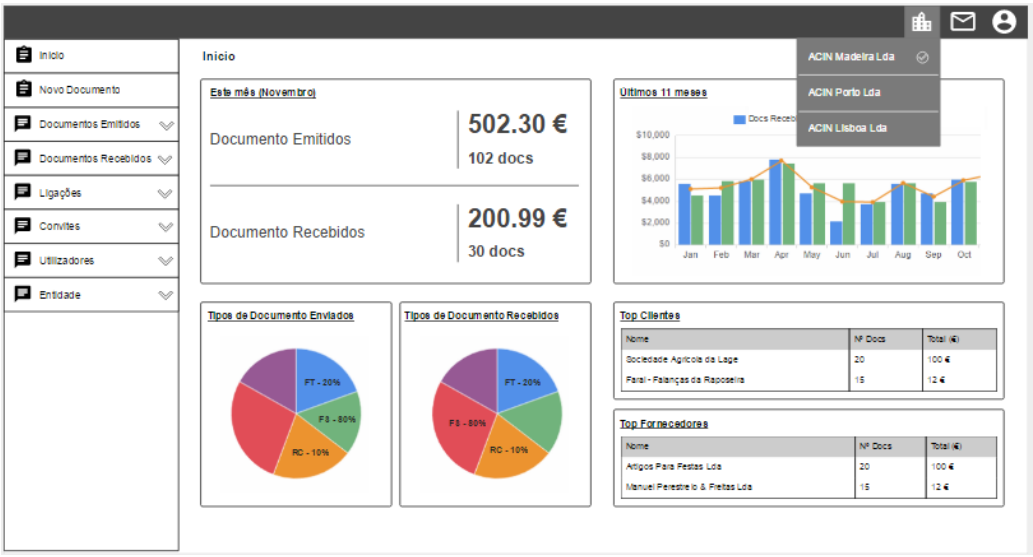


Figura 116 - Página início com as entidades

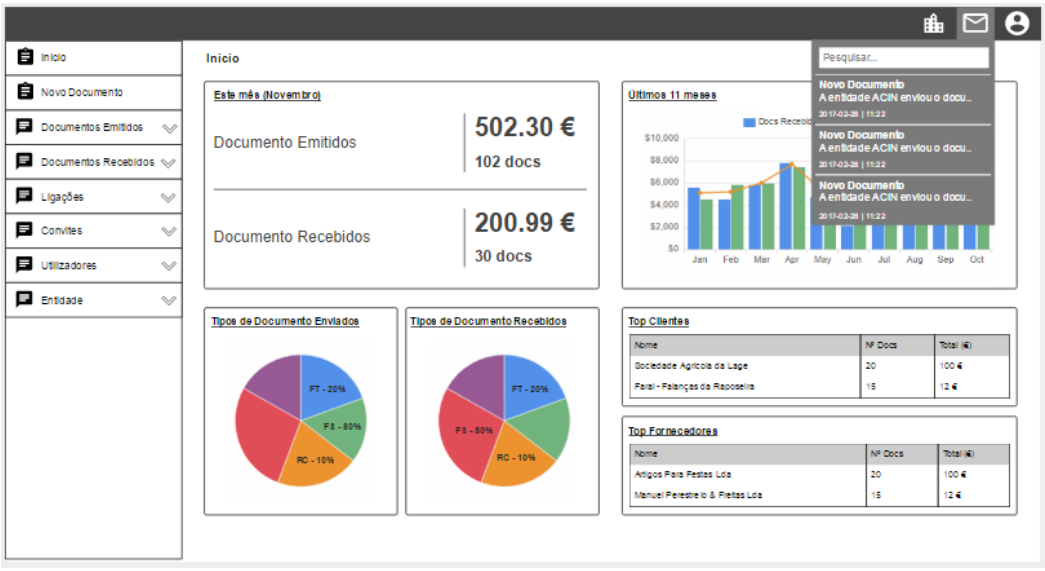


Figura 117 - Página início com as notificações

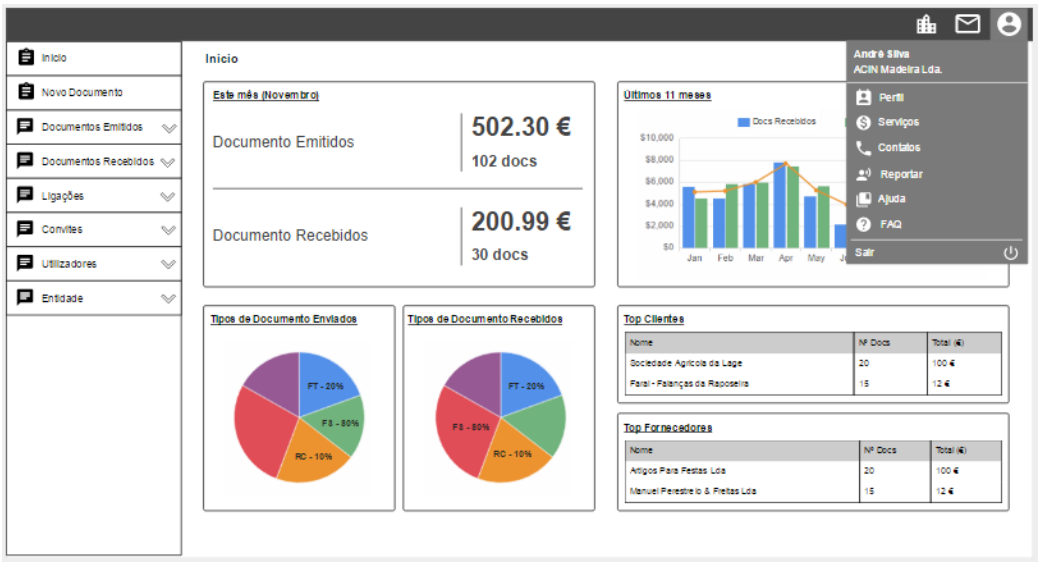


Figura 118 - Página início com opções de perfil do utilizador

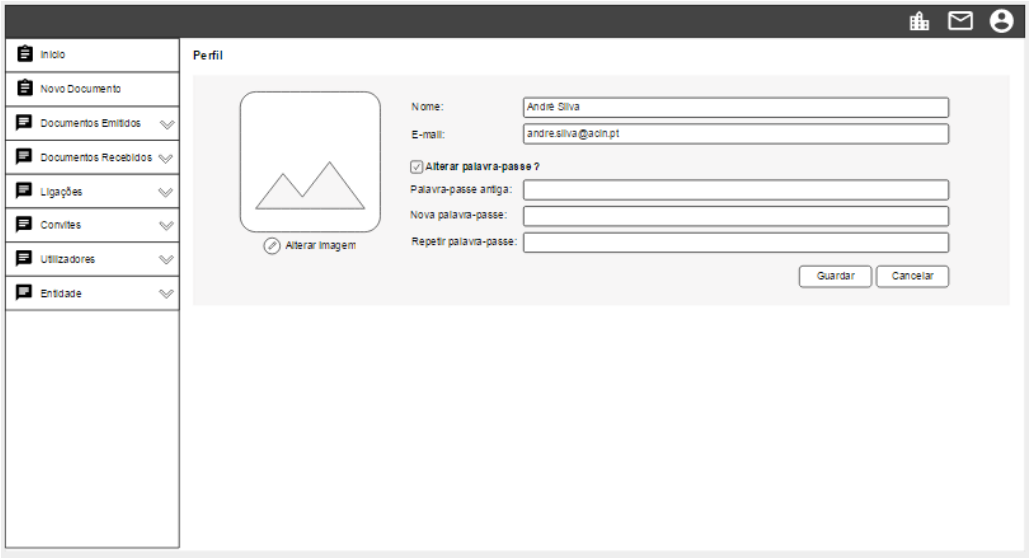


Figura 119 - Página perfil

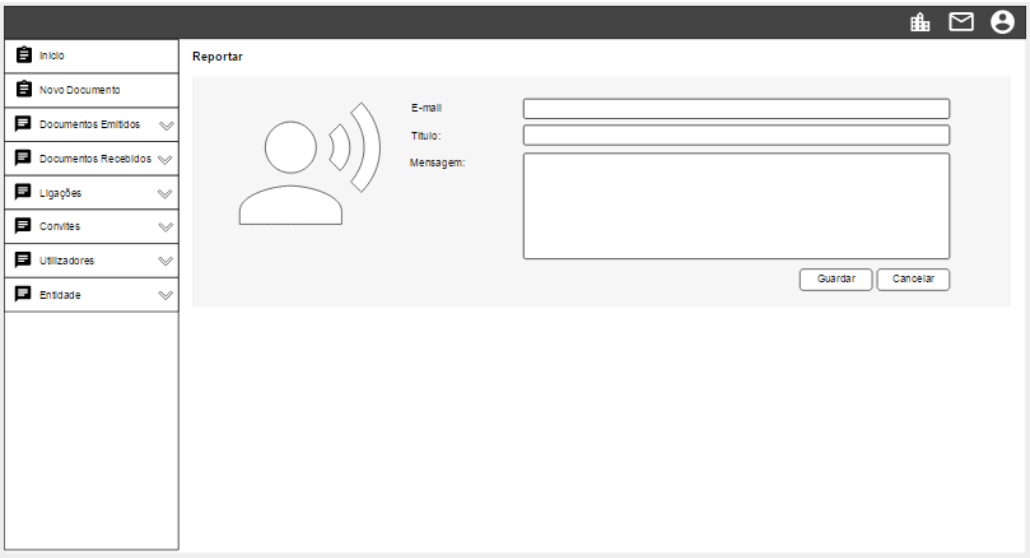


Figura 120 - Página reportar

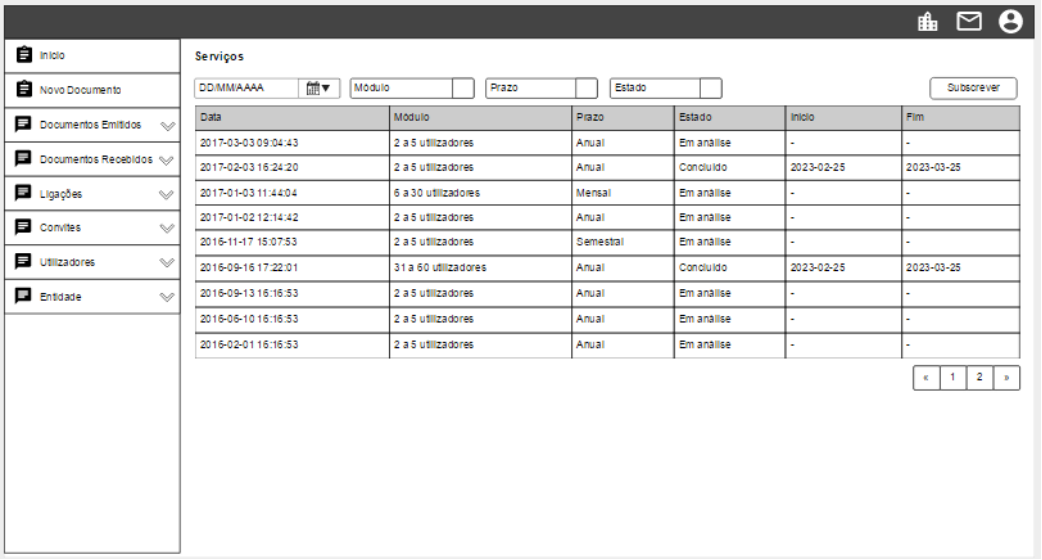


Figura 121 - Página serviços

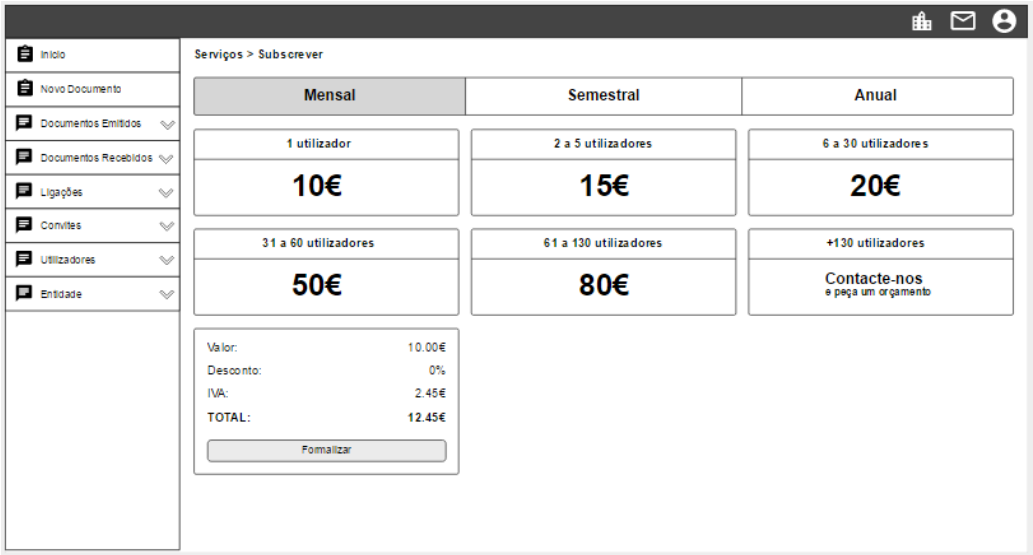


Figura 122 - Página para subscrever

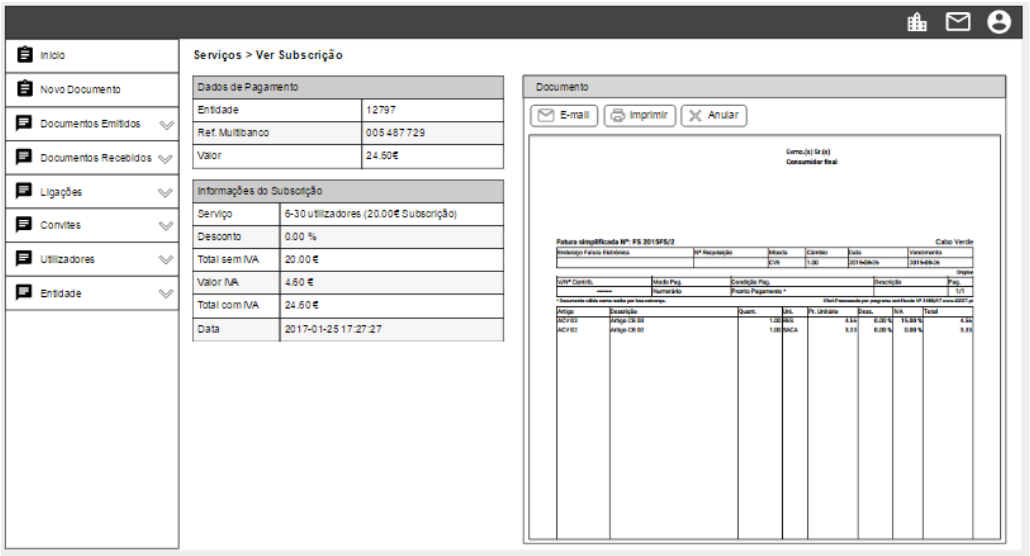


Figura 123 - Página para ver subscrição

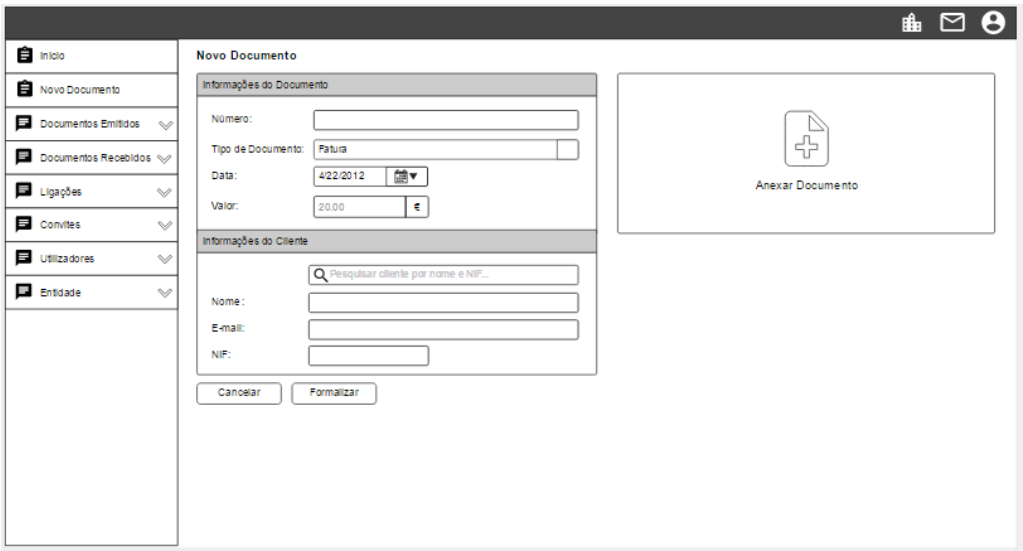


Figura 124 - Página para criar novo documento

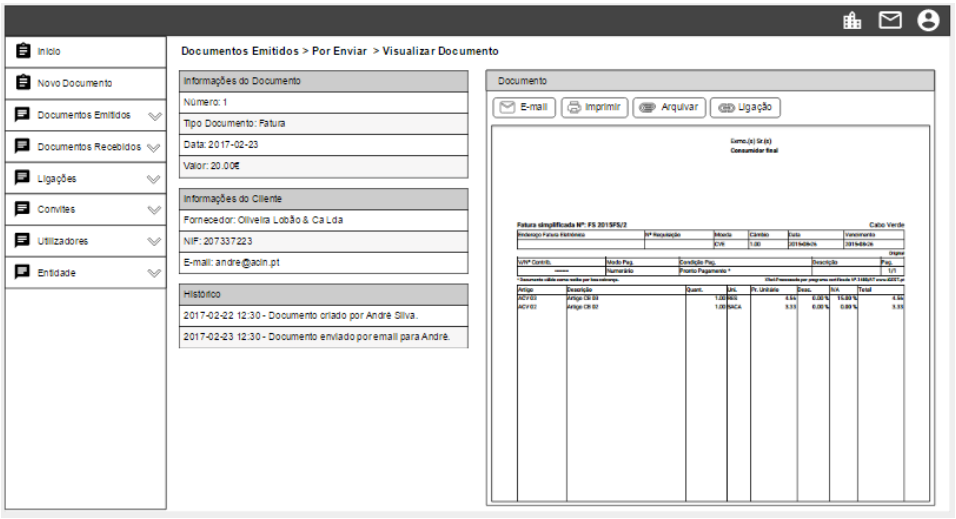


Figura 125 - Página para visualizar documento criado

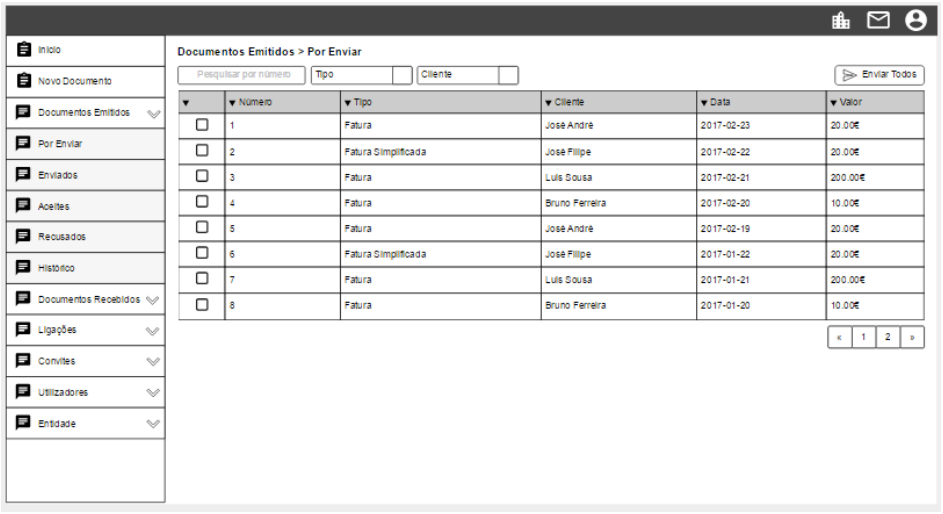


Figura 126 - Página para visualizar documentos emitidos por enviar

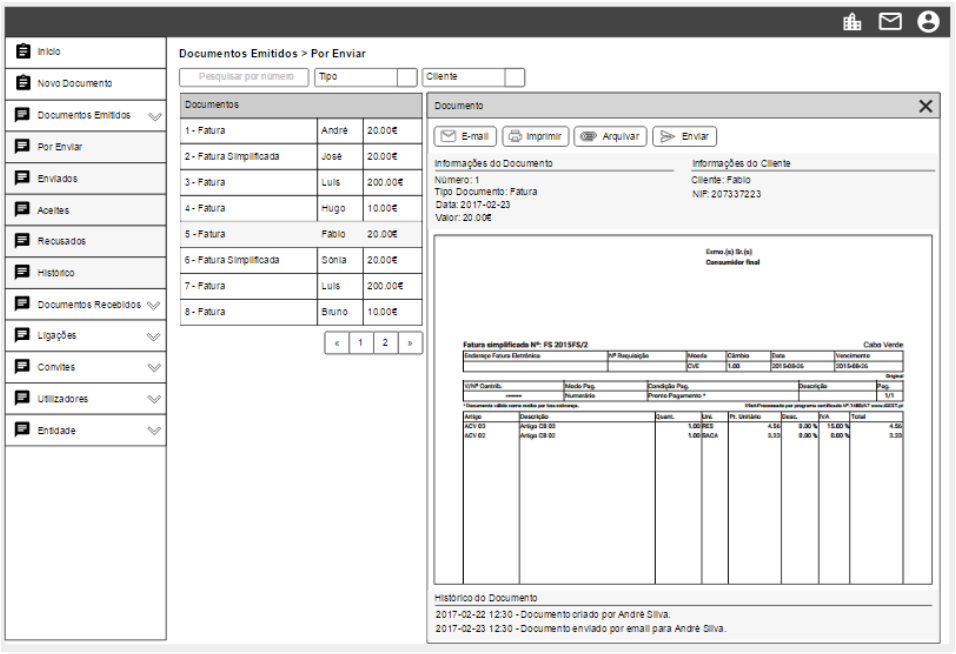


Figura 127 - Página para visualizar documentos emitidos

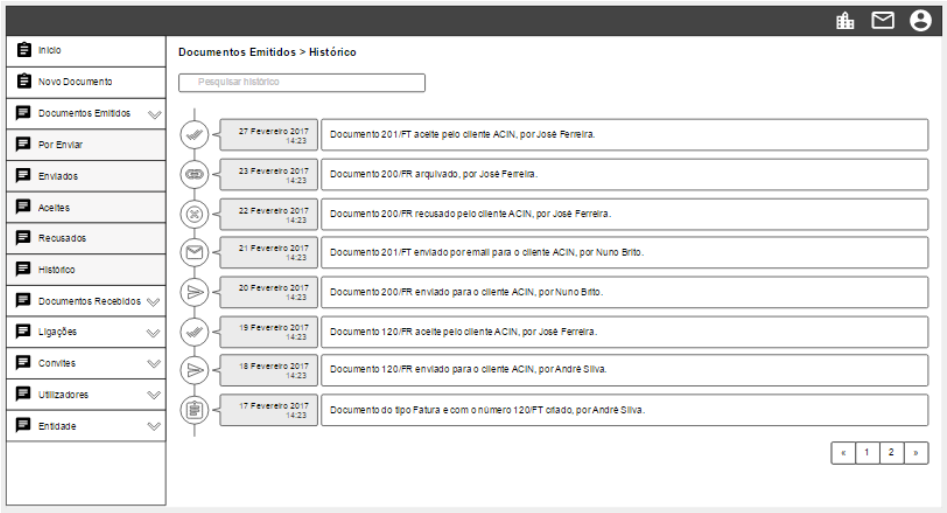


Figura 128 - Página para ver o histórico dos documentos emitidos

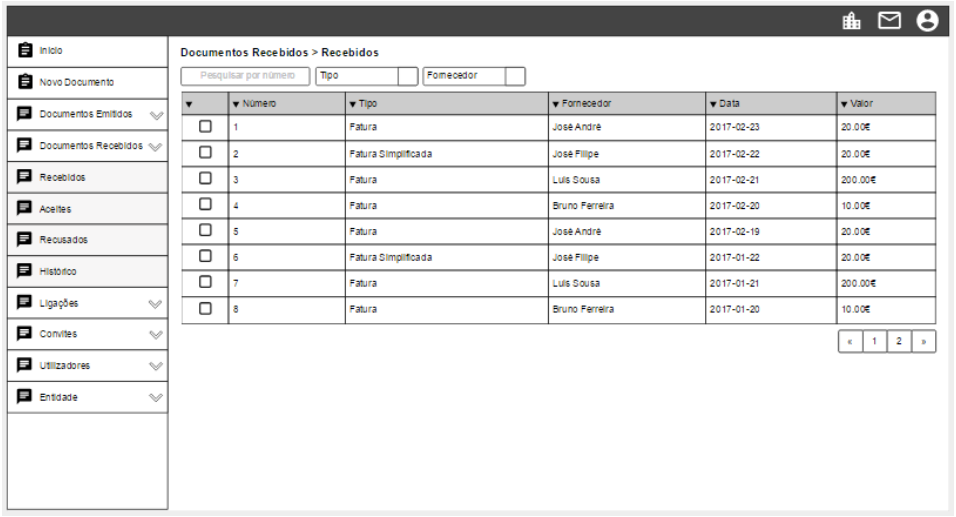


Figura 129 - Página para ver documentos recebidos

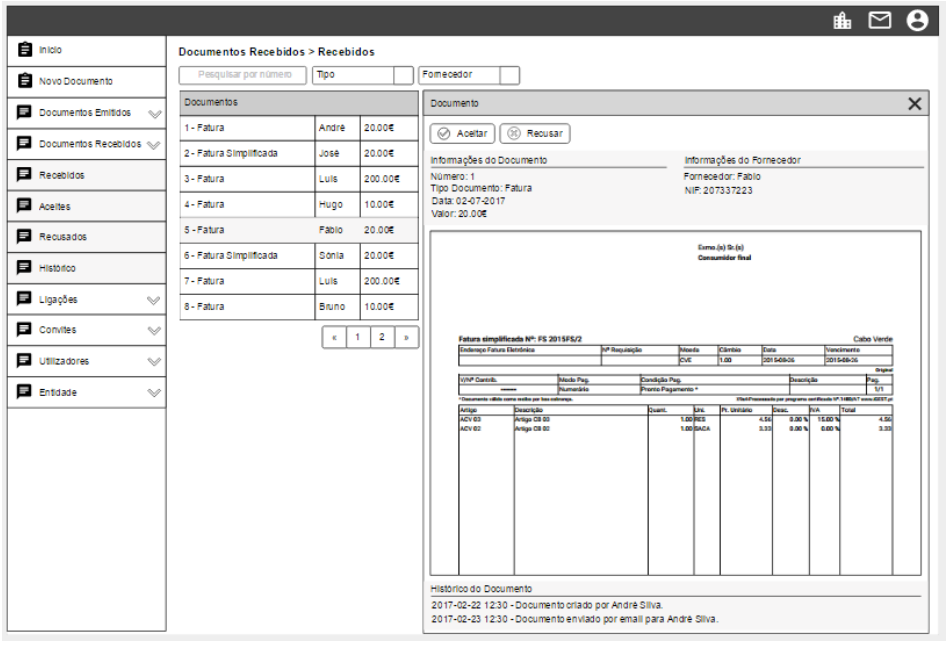


Figura 130 - Página para ver documentos recebidos

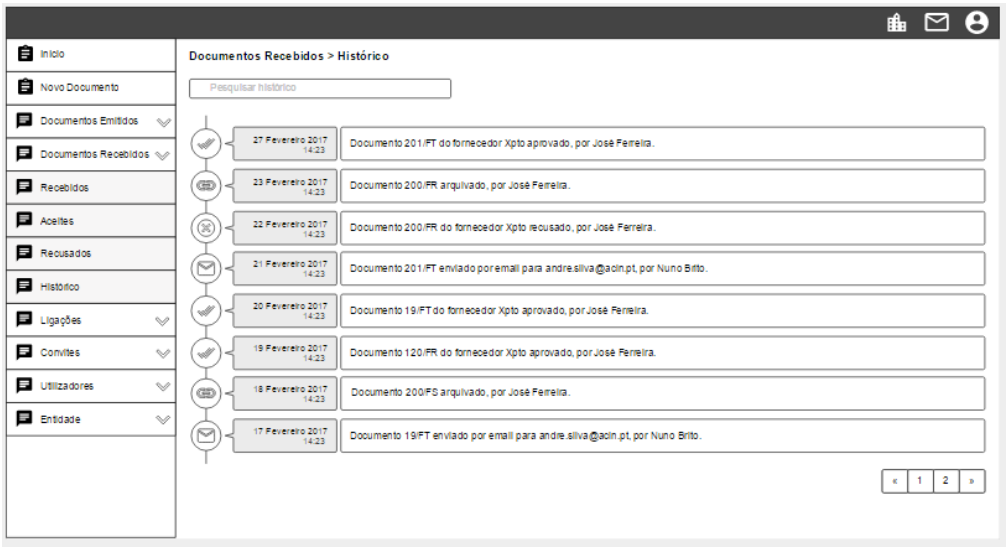


Figura 131 - Página para ver o histórico de documentos recebidos

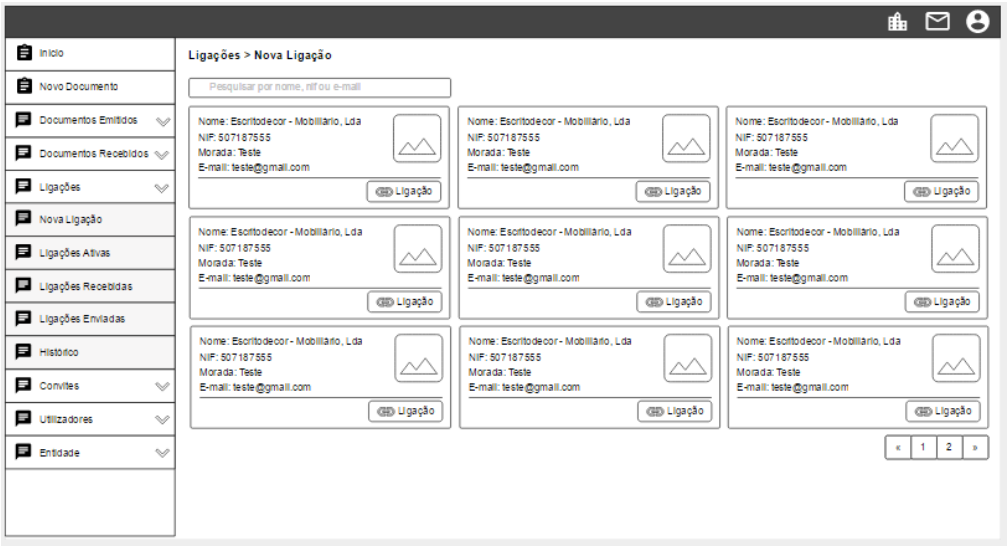


Figura 132 - Página para criar nova ligação

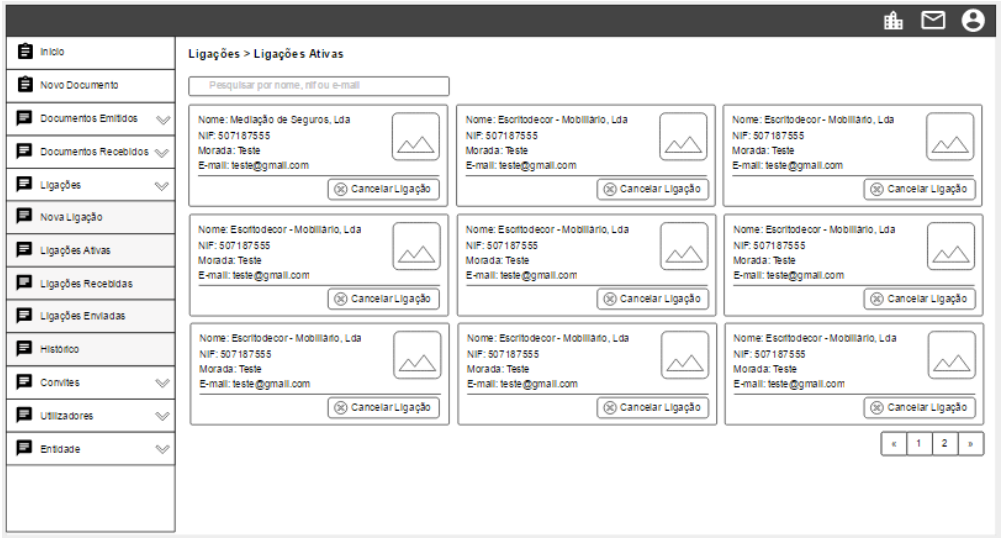


Figura 133 - Página com as ligações ativas

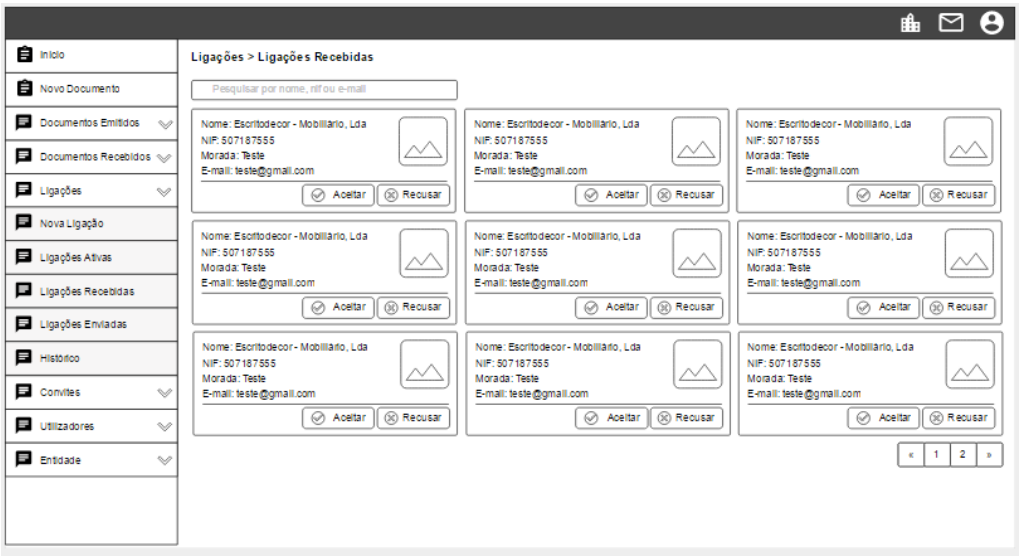


Figura 134 - Página com as ligações recebidas

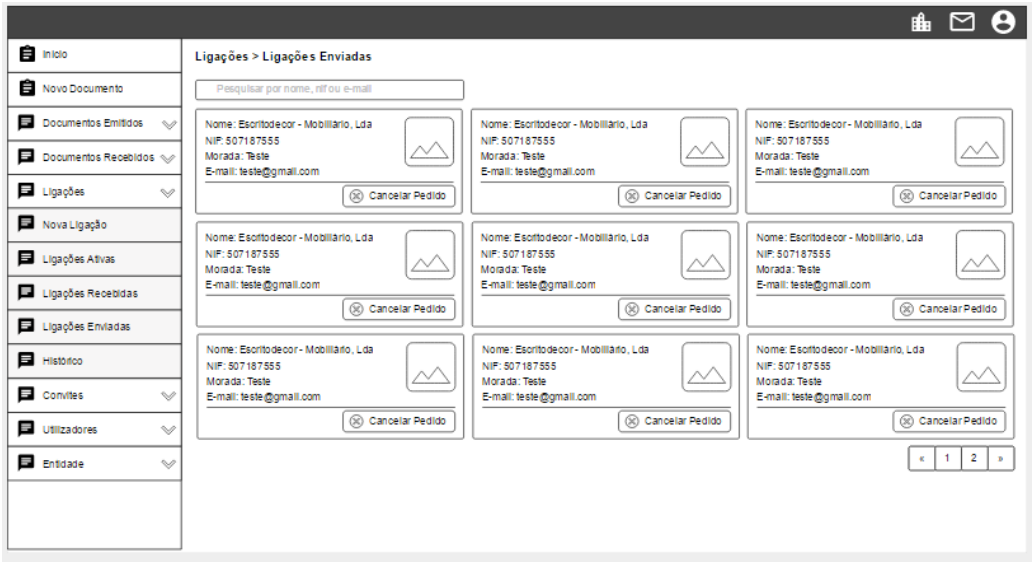


Figura 135 - Página com as ligações enviadas

Figura 136 - Página histórico das ligações

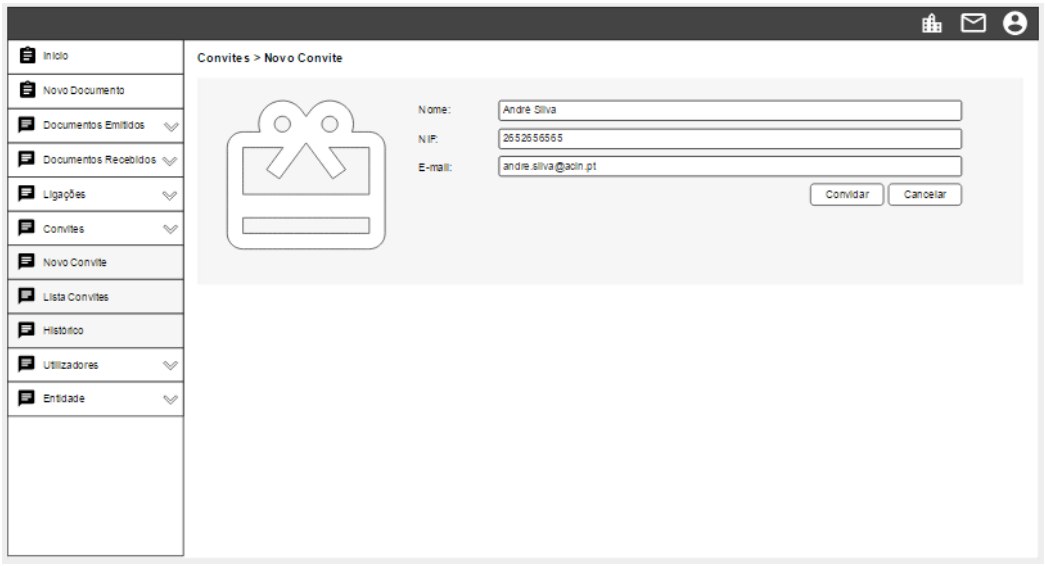


Figura 137 - Página para enviar novo convite

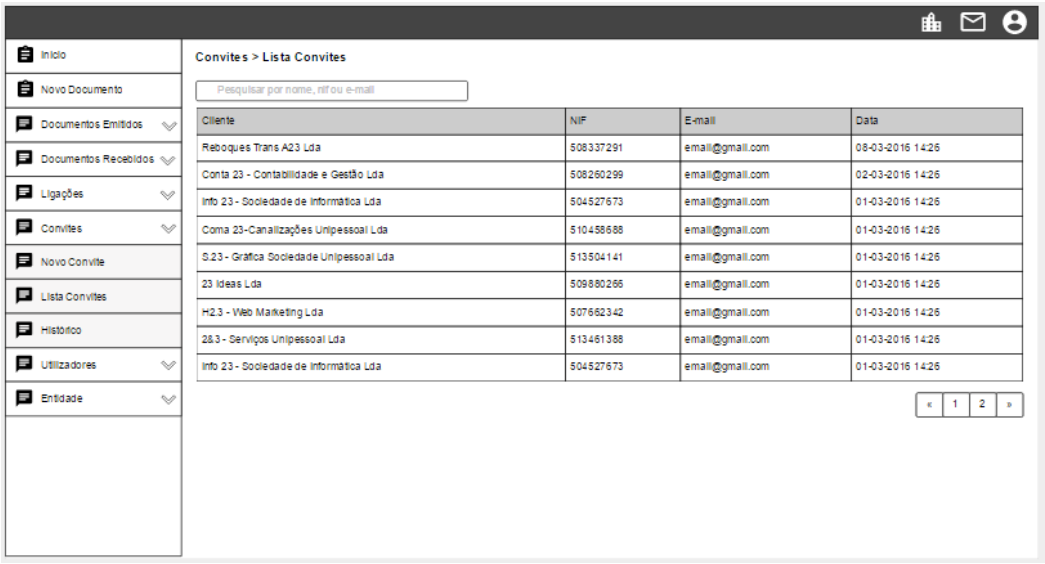


Figura 138 - Página com a lista de convites

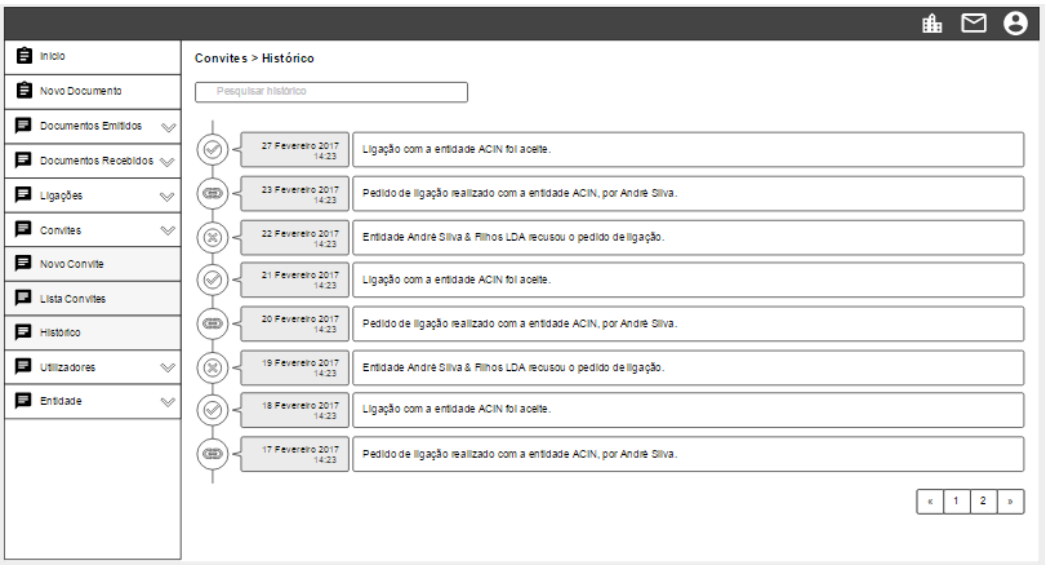


Figura 139 - Página com o histórico dos convites

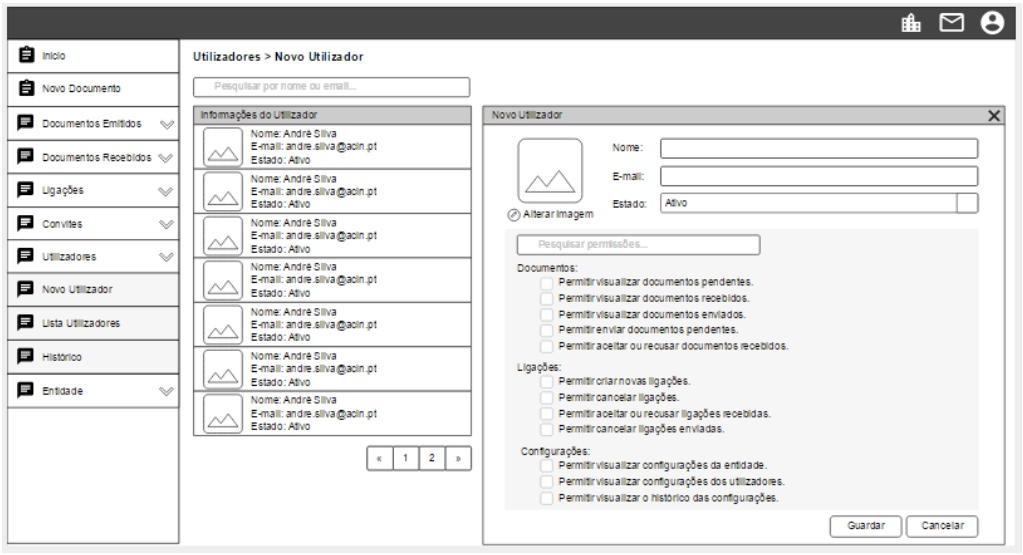


Figura 140 - Página para criar novo utilizador

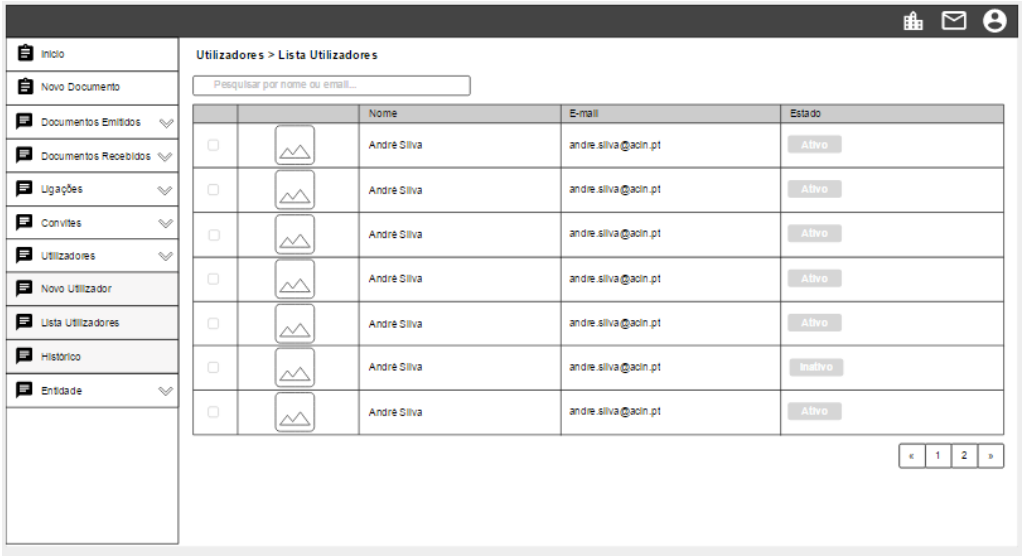


Figura 141 - Página com os utilizadores da entidade

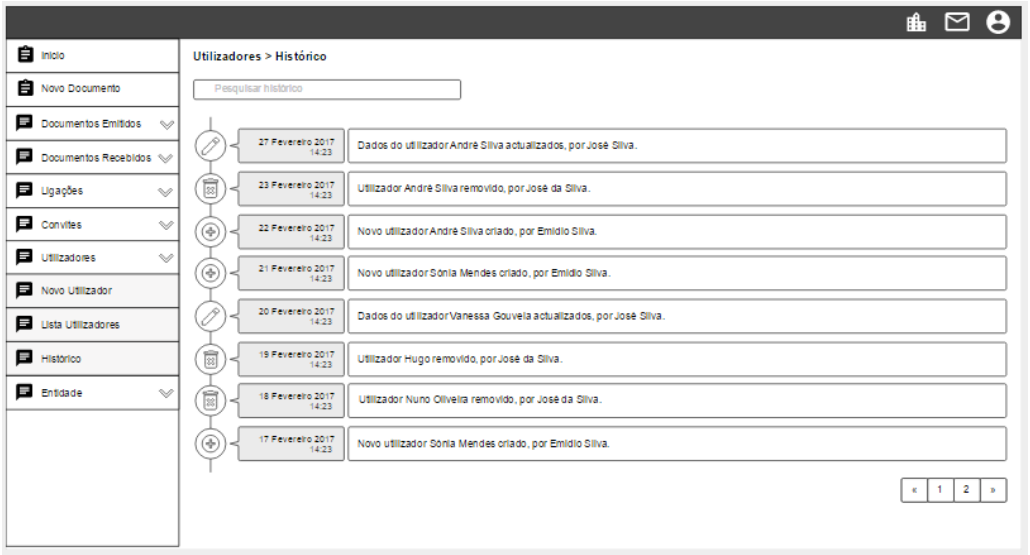


Figura 142 - Página com o histórico dos utilizadores

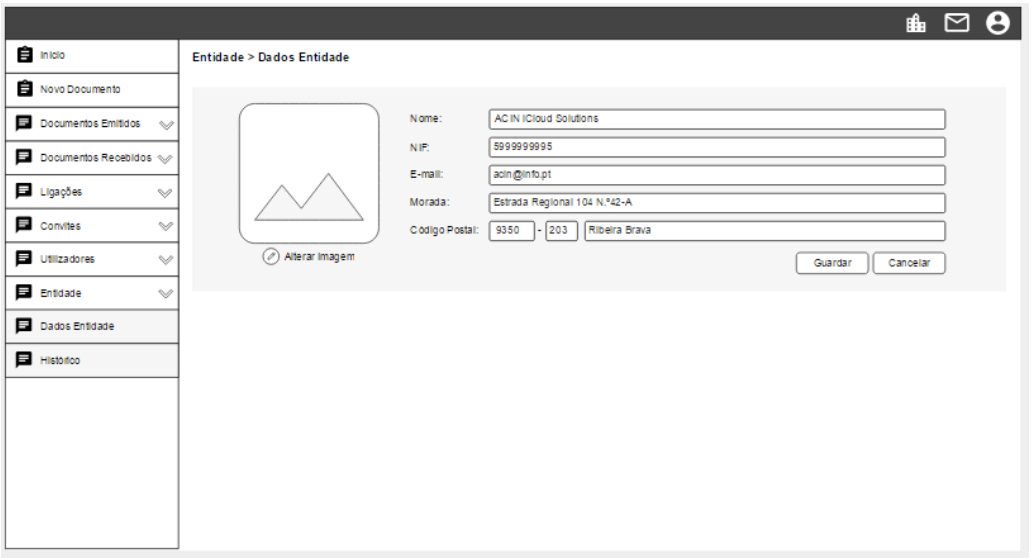


Figura 143 - Página com os dados da entidade

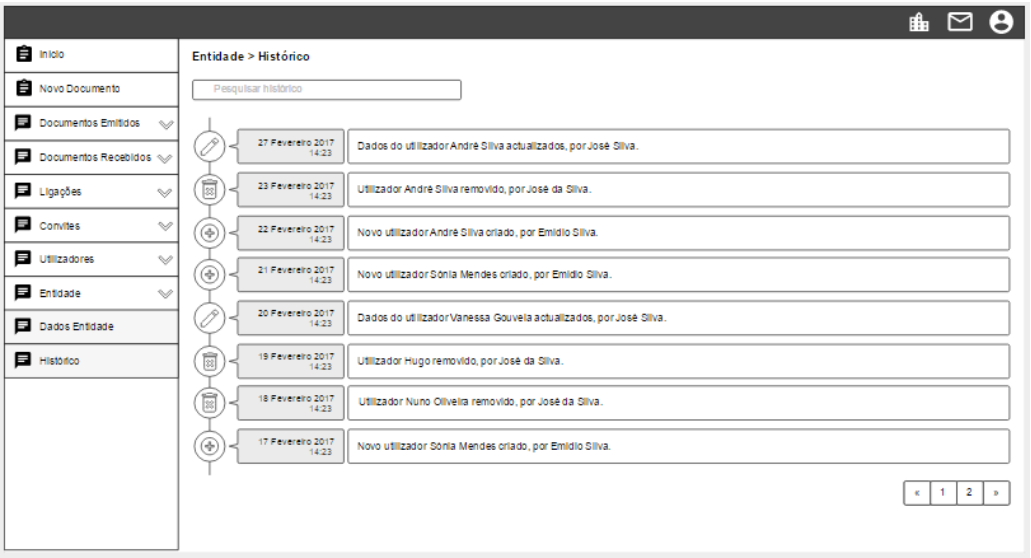


Figura 144 - Página com o histórico da entidade

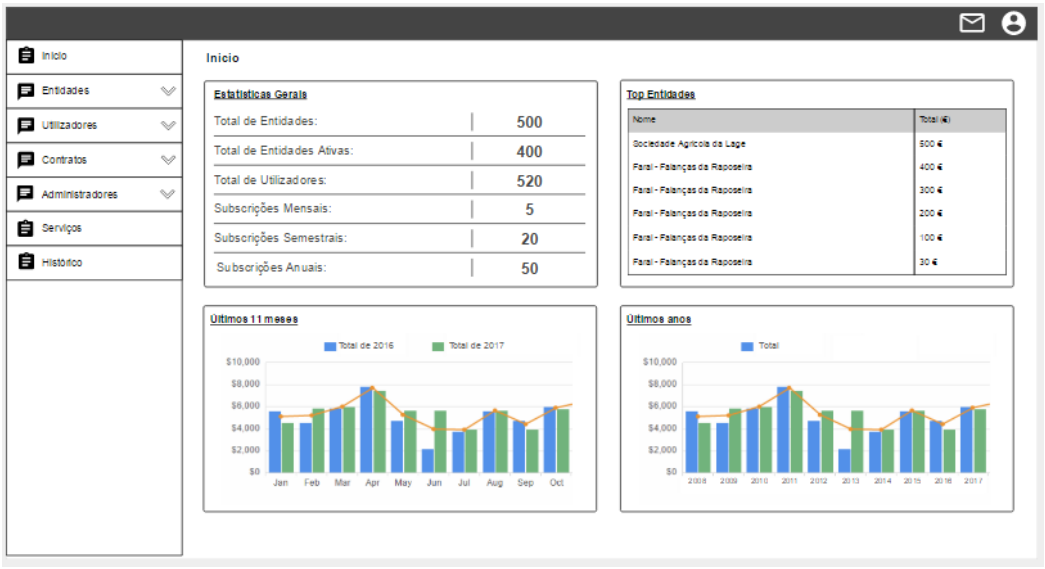


Figura 145 - Página início da administração

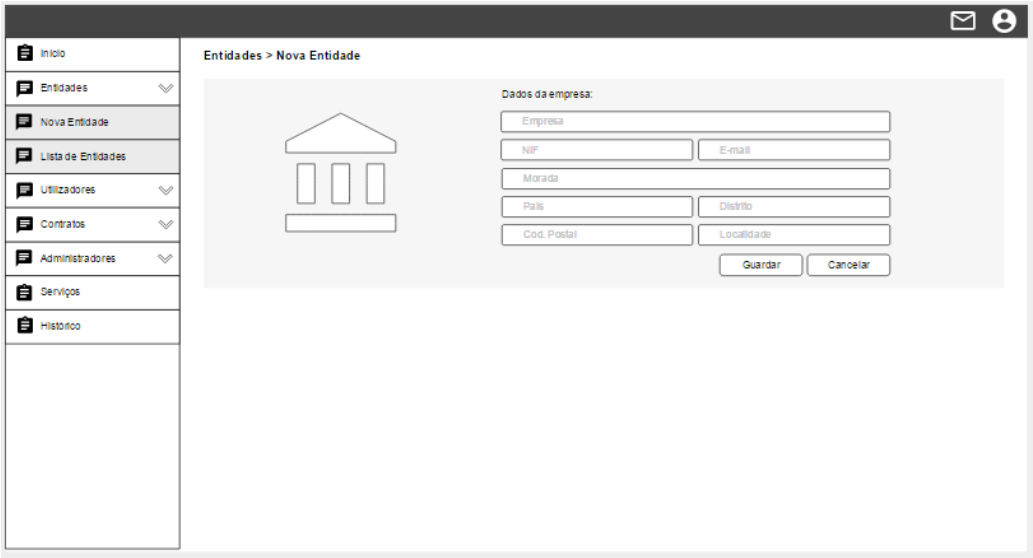


Figura 146 - Página para criar nova entidade na administração

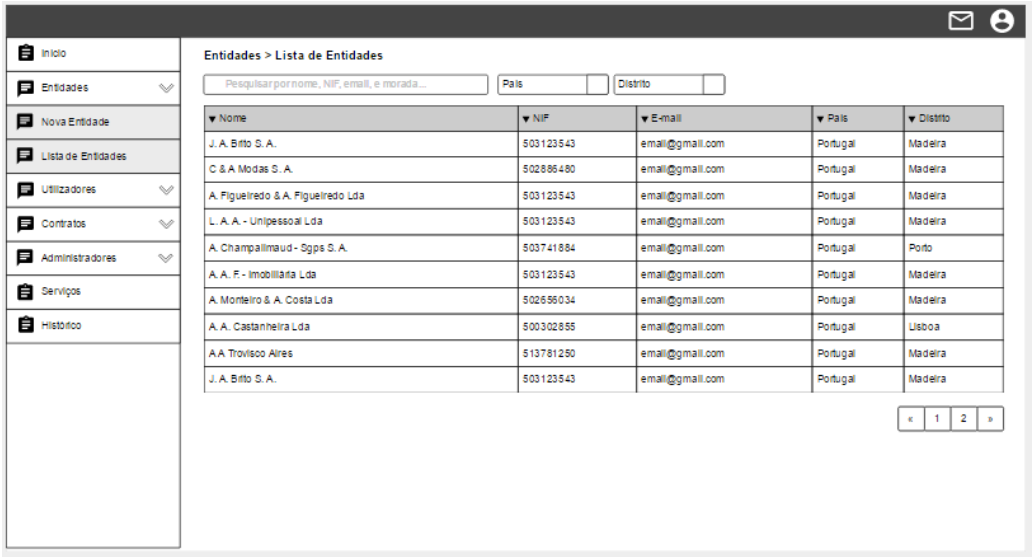


Figura 147 - Página com a lista das entidades na administração

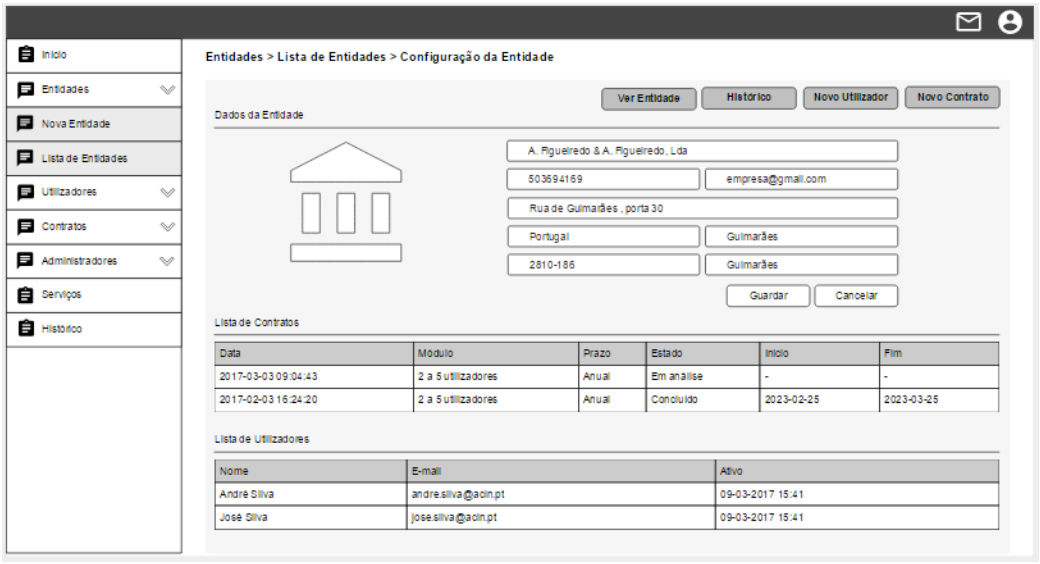


Figura 148 - Página com as configurações da entidade na administração

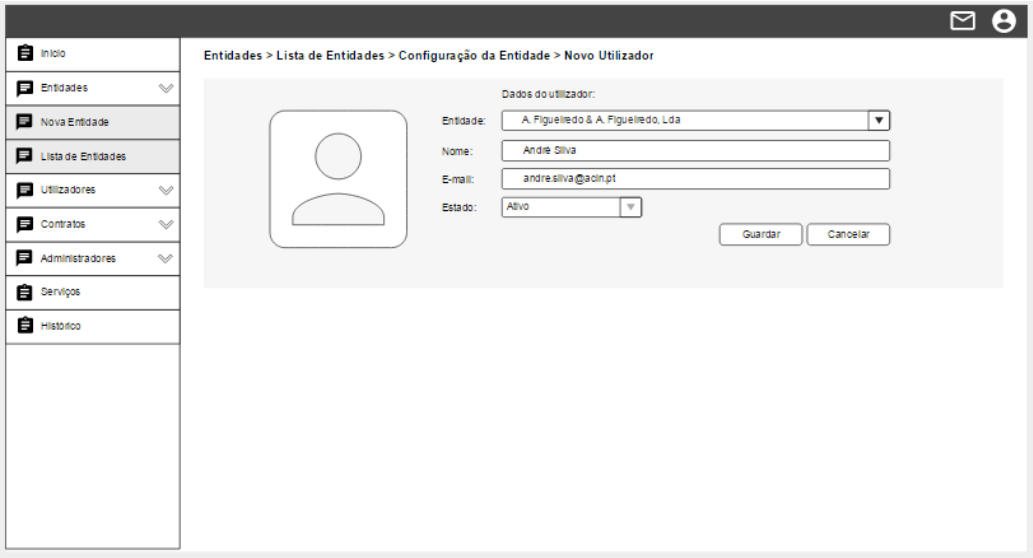


Figura 149 - Página para criar novo utilizador numa entidade na administração

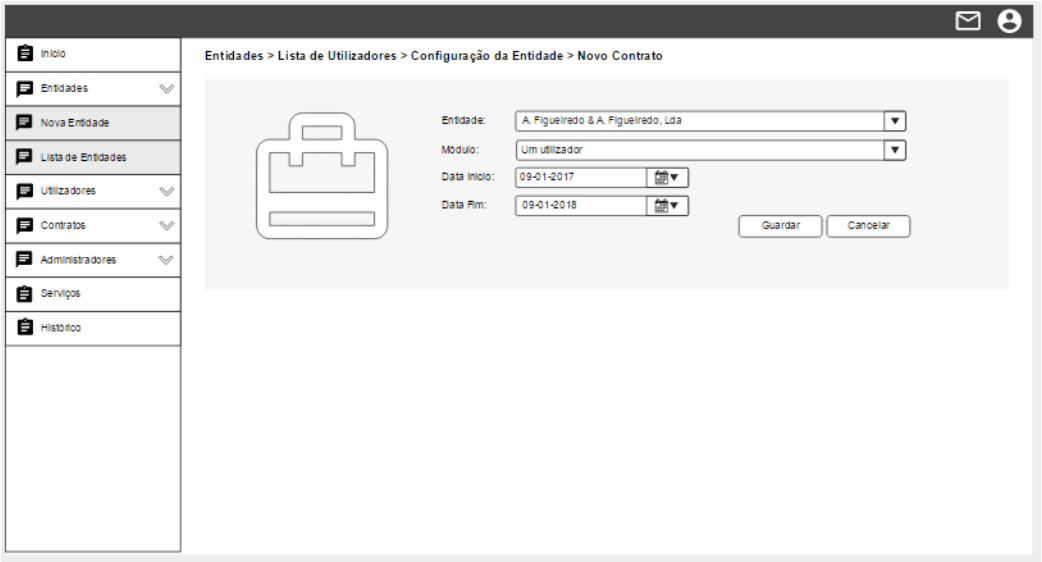


Figura 150 - Página para criar novo contrato numa entidade na administração

Figura 151 - Página para ver histórico da entidade na administração

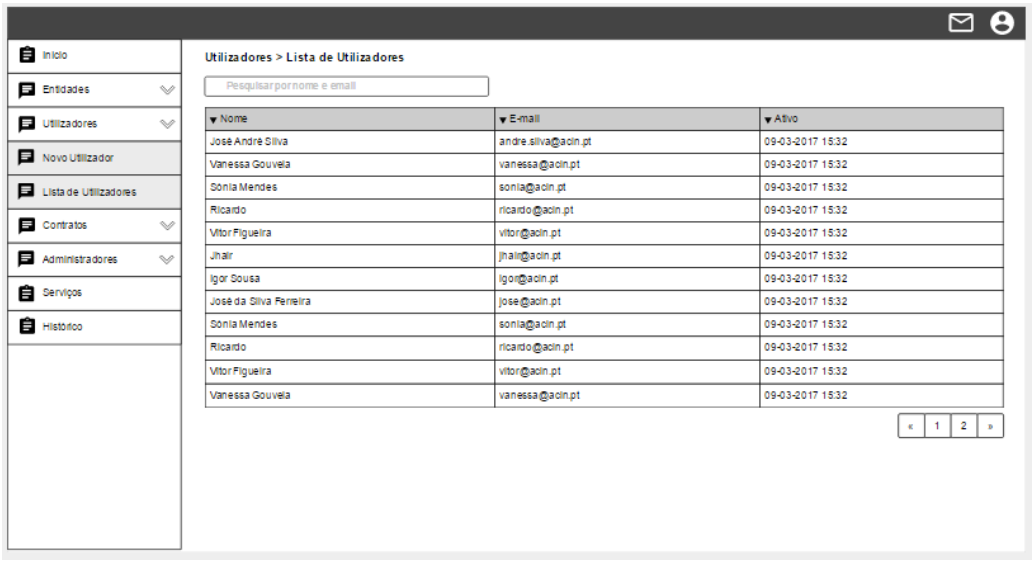


Figura 152 - Página com a lista de utilizadores na administração

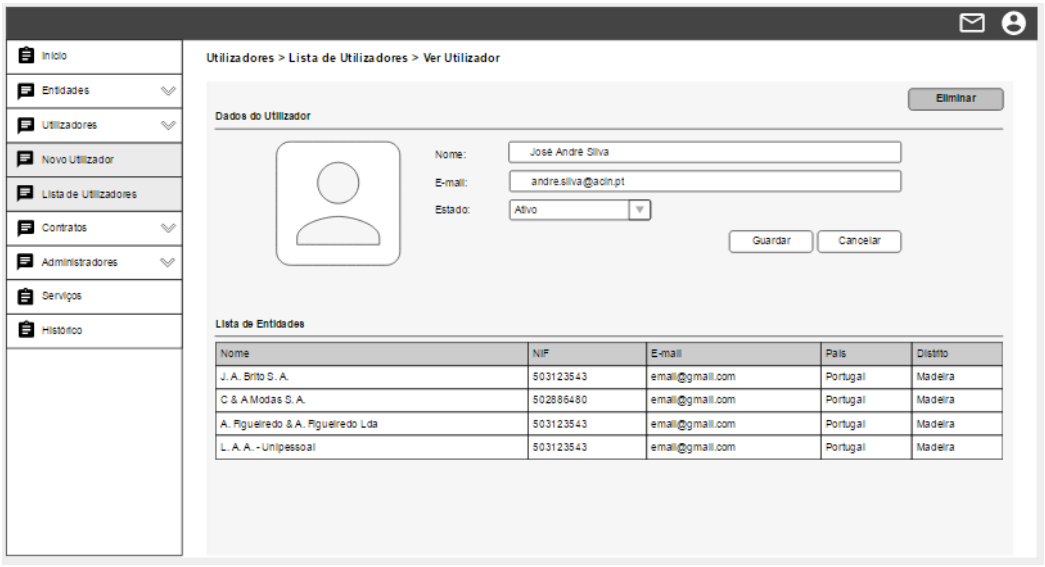


Figura 153 -Página para ver utilizador na administração

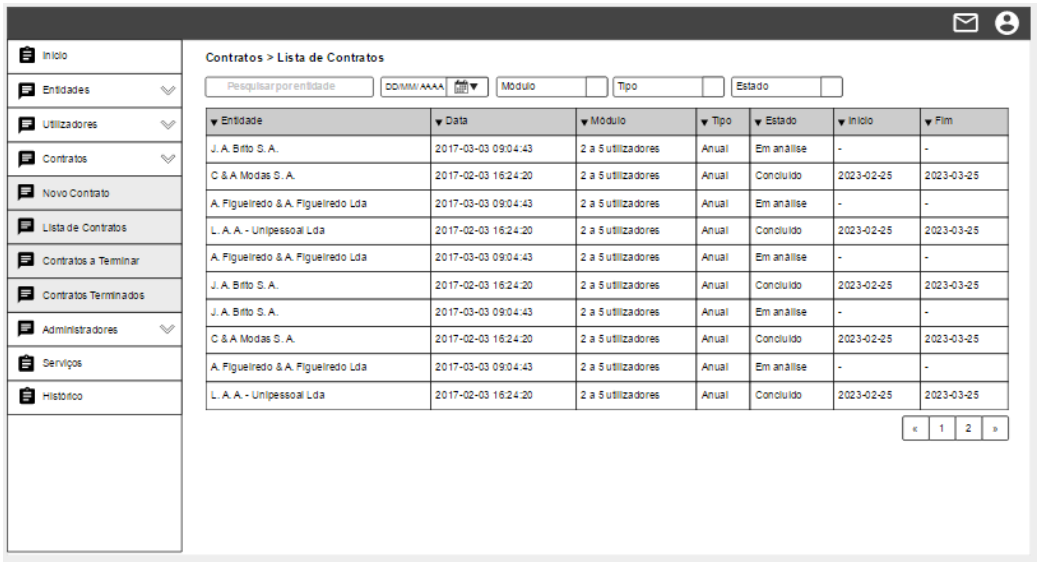


Figura 154 - Página com a lista de contratos na administração

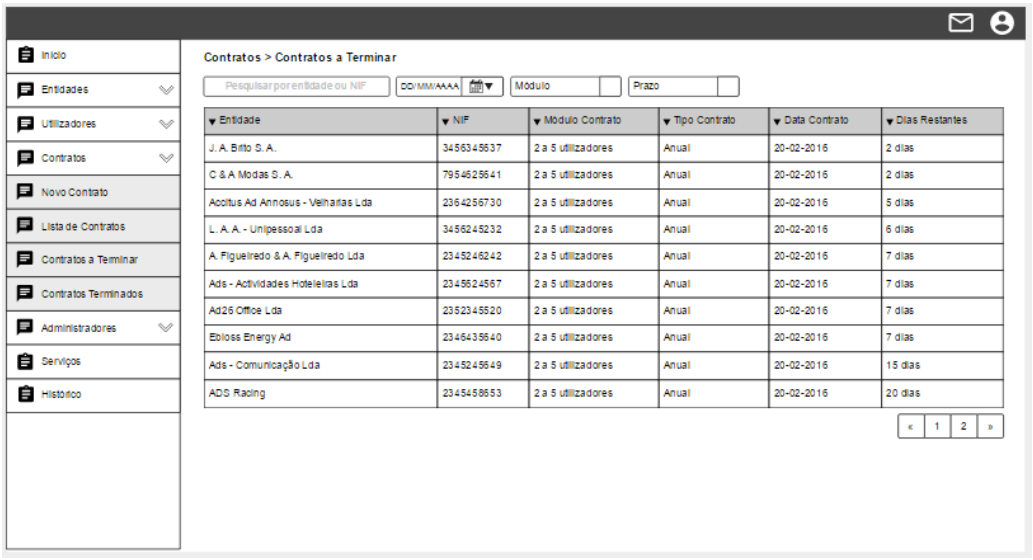


Figura 155 - Página para ver contratos a terminar na administração

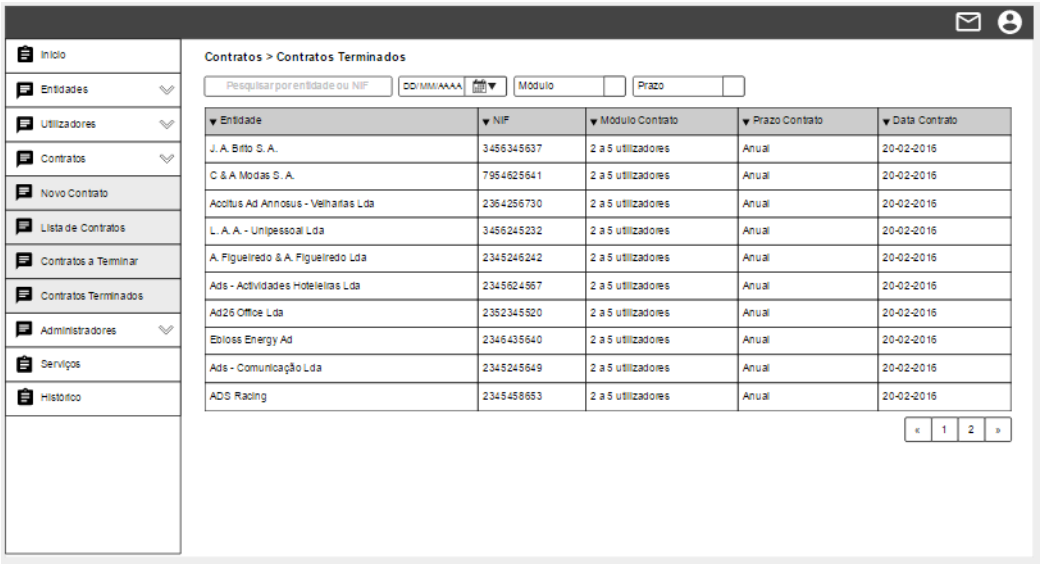


Figura 156 - Página para ver contratos terminados na administração

Figura 157 - Página para criar novo administrador na administração

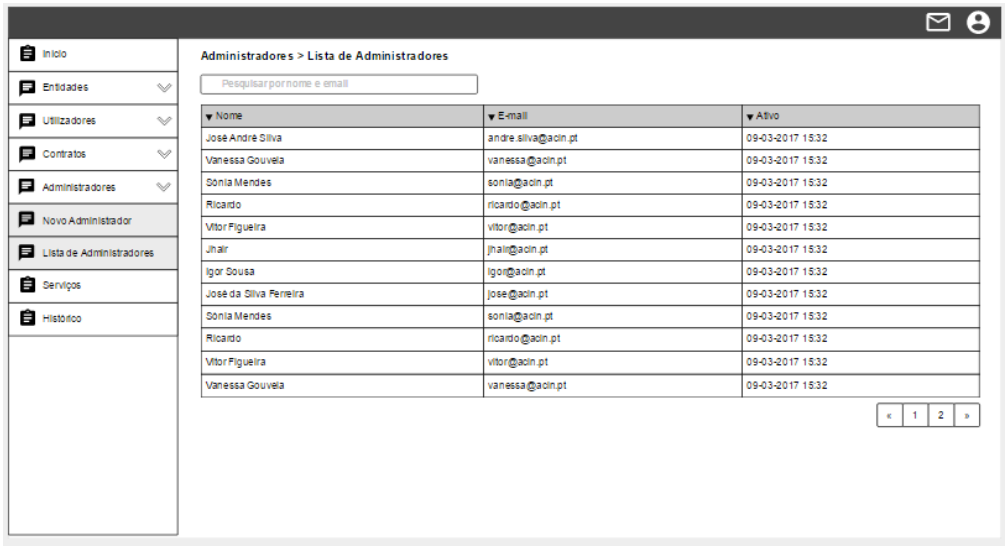


Figura 158 - Página com os administradores na administração

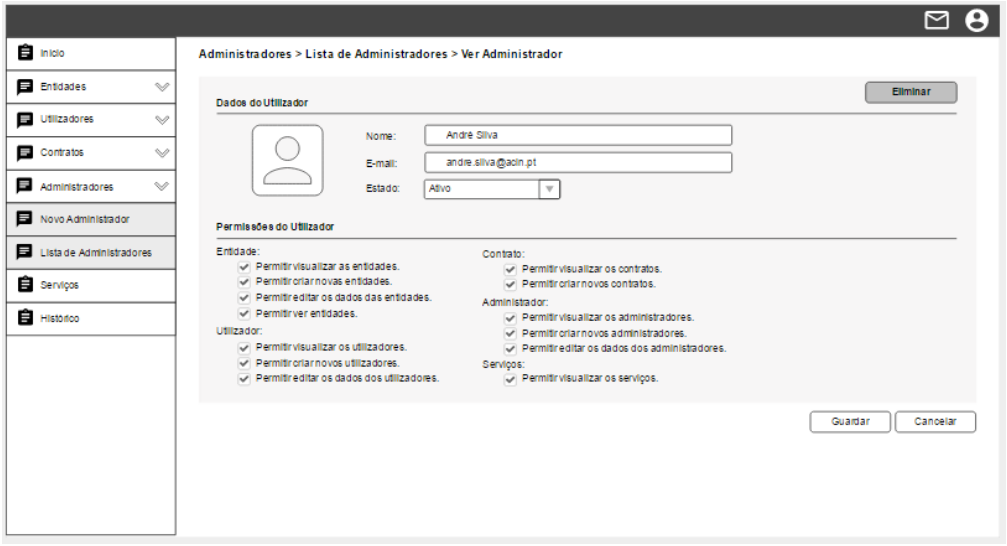


Figura 159 - Página para ver administrador na administração

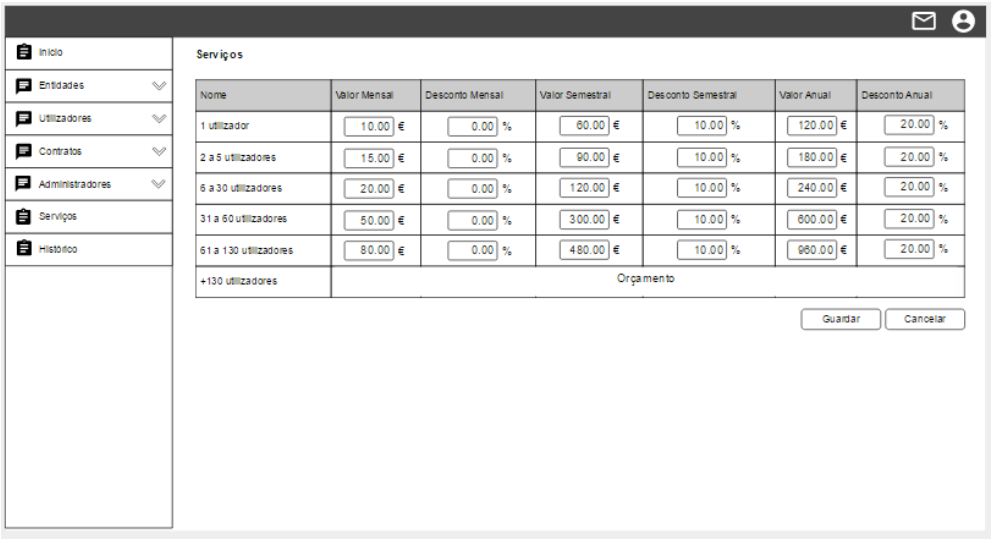


Figura 160 - Página para ver os serviços na administração

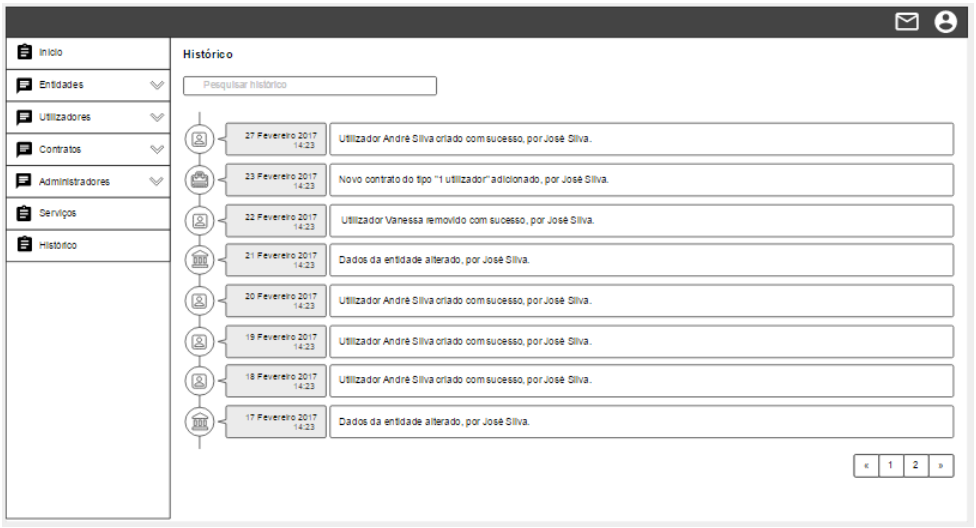


Figura 161 - Página com o histórico da administração

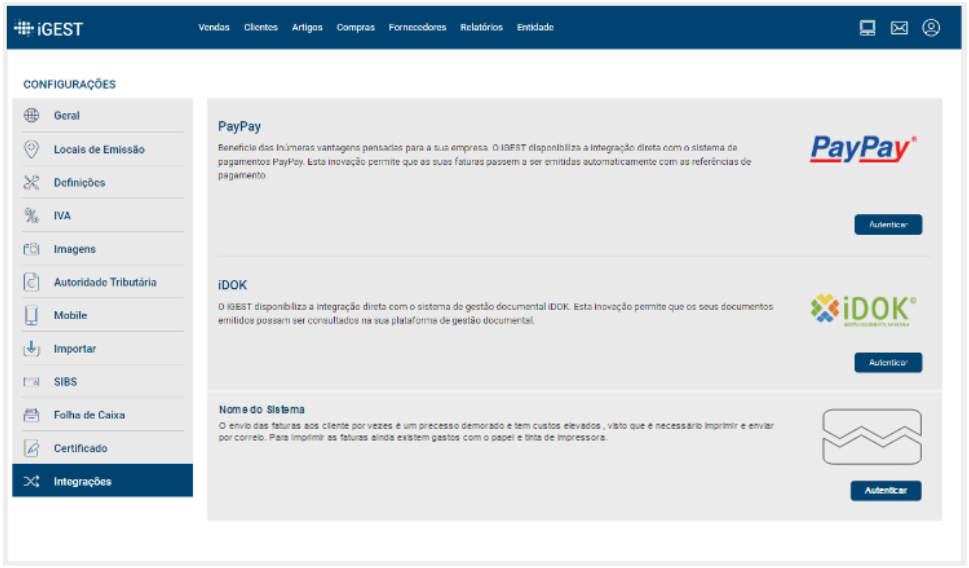


Figura 162 - Página de integrações do iGEST

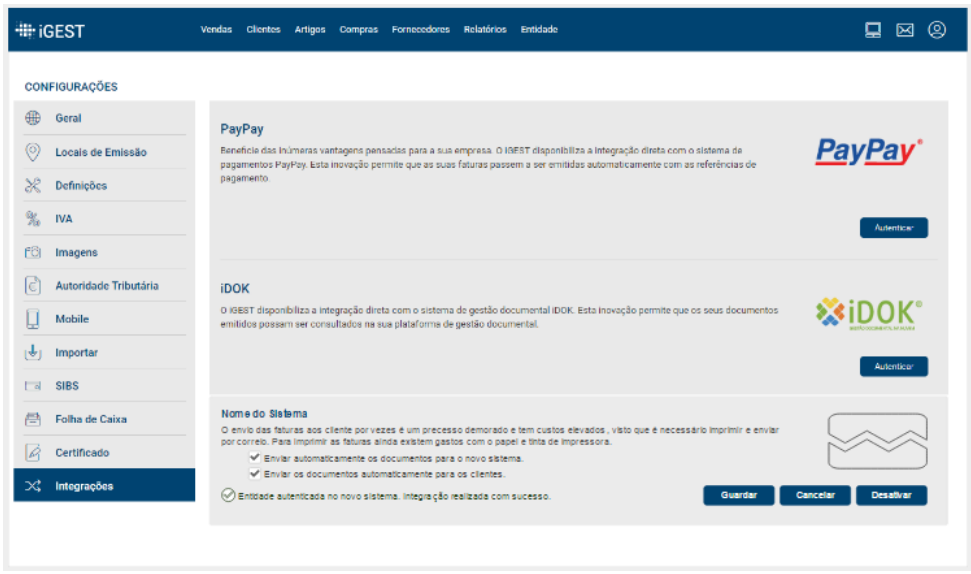


Figura 163 - Página de integração do iGEST com integração com o novo sistema ativa