



INSTITUTO POLITÉCNICO DE LEIRIA

ADMINISTRATION WEST

RELATÓRIO DE PROJETO EM SISTEMAS DE
INFORMAÇÃO

RODOLFO BARREIRA N° 2180714
CIDÁLIA PINTO N°2180709

PROGRAMAÇÃO EM SISTEMAS DE
INFORMAÇÃO

14 FEVEREIRO 2021

Relatório de Projeto em Sistemas de Informação para cumprimento dos requisitos necessários
à realização da prova de apresentação de projeto do Curso Técnico Superior Profissional
(TeSP) de **Programação de Sistemas de Informação** realizado sob a orientação de Marco
Vicente

DECLARAÇÃO

Declaro que este Relatório se encontra em condições de ser apreciada (o) pelo júri a designar.

O estudante Cidália Pinto,

Leiria, de de

Relatório de Projeto em Sistemas de Informação para cumprimento dos requisitos necessários à realização da prova de apresentação de projeto do Curso Técnico Superior Profissional (TeSP) de **Programação de Sistemas de Informação** realizado sob a orientação de Marco Vicente

DECLARAÇÃO

Declaro que este Relatório se encontra em condições de ser apreciada (o) pelo júri a designar.

O estudante Rodolfo Barreira,

Leiria, de de

Resumo

RELATÓRIO DE PROJETO – Administration West

Cidália Pinto, Rodolfo Barreira

Administration West consiste num projeto desenvolvido especificamente para micro, pequenas empresas presentes no concelho de Torres Vedras. Este projeto tem um *website* desenvolvido utilizando uma *framework* de PHP designada CodeIgniter, acesso a uma *API Restful* utilizando a mesma *framework* e uma aplicação *Android*, desenvolvida em *JAVA*.

O objetivo deste projeto é promover o comércio *Online* de modo a facilitar a venda de produtos na atual situação económica debilitada pela atual pandemia.

No *website*, o cliente pode ver os produtos, as empresas parceiras, as categorias, o perfil, o histórico de compras e o carrinho e efetuar as suas compras. A empresa pode criar os produtos, os funcionários e vendas com os produtos da mesma. Os funcionários poderão criar produtos, vendas para a empresa em que estão. Para além disso o administrador pode criar empresas e atribuir funcionários à empresa.

A aplicação *Android* recebe os dados através da *API Restful*, esta é direcionada à utilização apenas para os clientes, ou seja, para poderem efetuar os seus pedidos, verem os produtos, histórico de compra e outros detalhes.

PALAVRAS-CHAVE: CodeIgniter, Android, Restful, API

Índice

Resumo	iv
1. Dicionário de significados	2
2. Introdução	3
3. Metodologia.....	4
4. Arquitetura do Sistema	5
5. Gestão do Projeto	6
6. Análise	9
7. Desenho	11
8. Implementação.....	28
9. Testes.....	30
10. Conclusão e trabalho futuro	33
11. Webgrafia	34
12. Guia de instalação.....	35
13. Anexos.....	36

Índice de ilustrações

1 - Arquitetura do sistema.....	5
2 - Tabela de atividades dos utilizadores	6
3 - Gráfico das atividades.....	7
4 -Página inicial do website.....	11
5 - Página de produtos.....	12
6 - Detalhes do produto	13
7- Carrinho do utilizador.....	14
8 - Histórico de compras.....	15
9 - Atividade de produtos da aplicação	16
10 - Detalhes dos produto	17
11 - Carrinho na aplicação	18
12 - Caso de uso	19
13 - Estrutura das tabelas da base de dados	22
14 - Mockup inicial	23
15 - Estrutura de funcionamento explicada	24
16 - Mockup inicial do site.....	25
17 - Estrutura do frontoffice explicada.....	26
18 - Dashboard do backoffice.....	27
19 - Resposta ao teste de Mudança de password	30
20 - Resposta ao teste de mudança falhada de password	30
21 - Resposta ao teste sucedido de mudança de password	31
22 - Teste de renovação da password	31
23 - Teste add product to cart.....	31
24 - Teste de criação de venda	32
25 - Teste de mudança de estado.....	32

1. Dicionário de significados

Neste dicionário estão várias palavras e siglas que ao longo do projeto serão referidas, de modo assim a conseguir explicar os seus significados.

- **PHP** – Linguagem de programação web de processamento de texto;
- **Android** – Plataforma de aplicações móveis;
- **Java** – Linguagem de programação orientada a objetos;
- **Framework** - Conjunto de ferramentas que ajudam a agilizar o desenvolvimento e reutilização de código.
- **API** – É uma ferramenta de programação que permite a conexão e troca de informações entre aplicações distintas;
- **Rest** – Conjunto de métodos e regras de conexão entre o cliente e um serviço web.
- **Frontoffice** – É o local da aplicação ao qual o cliente comum tem acesso;
- **Backoffice** – É o local de administração de uma aplicação;
- **Mockup** – É um modelo do qual a aplicação se irá basear;
- **CRUD** – Método de adicionar, ler, atualizar e apagar dados;
- **Bootstrap** – Framework web utilizada para o design de aplicações;
- **Javascript** – É uma linguagem de script que ajuda a criar interações no website sem necessitar de atualizar a página;
- **SQL ou MYSQL** – Linguagem de utilizada para a consulta de dados estruturados.

2. Introdução

O nosso projeto foi desenvolvido especificamente para empresas locais de modo a promover o comércio *online*. A ideia do projeto é criar uma empresa que permita a outras empresas venderem os seus produtos na nossa plataforma, gerir as vendas através da área administrativa e mostrar ao cliente com a finalidade de promover a venda online de produtos

Para o desenvolvimento deste projeto utilizámos a *framework* de *PHP Codeigniter* para o desenvolvimento *web* e para a *API*, e para o desenvolvimento da aplicação utilizámos *Android* com a linguagem *Java*.

Neste relatório iremos falar das várias etapas de desenvolvimento das quais, dividimos em 8 tópicos, metodologia, planeamento do projeto, arquitetura do sistema, gestão do projeto, desenhos do projeto, implementação, testes realizados e conclusão.

No tópico da metodologia vamos explicar os métodos que adotámos no desenvolvimento do projeto. Iremos também apresentar a maneira como planeámos o projeto no planeamento e a arquitetura do sistema utilizada no nosso sistema no tópico respetivo.

Na gestão do projeto vamos apresentar o plano do projeto, o gráfico *Gantt* e as funções de cada membro do grupo.

Nos desenhos mostraremos os protótipos, casos de uso, modelo de dados e *mockups* iniciais e finais. Iremos também falar sobre o desenvolvimento do projeto e como o implementámos.

Nos testes iremos mostrar os testes realizados à aplicação, no nosso caso para o website.

3. Metodologia

A metodologia adotada na gestão e desenvolvimento do projeto é uma metodologia ágil, o *Scrum*.

O nosso grupo realizou vários *sprints* com duração de 1 semana. No final de cada semana reunimo-nos através do *Discord* para avaliar o ponto da situação do projeto, em que víamos o quanto progrediu e também o que precisaríamos de fazer para prosseguir para a próxima fase. Para além disso, também realizávamos reuniões diárias de curta duração sempre que possível para poder ver o progresso diário.

Nesta metodologia precisamos de ter um *Product Owner*, um *Scrum Master*, uma equipa de desenvolvimento (*Scrum team*). O *Product Owner* é o responsável pela gestão do *Product backlog* e gestão de reuniões. O *Scrum Master* é responsável por manter a equipa a seguir a metodologia e ter os passos corretos para o bom funcionamento. A *Scrum Team* é responsável por desenvolver o projeto. O *product backlog* é o registo das funcionalidades desejadas para o produto.

Nos recursos humanos escolhemos que o *Product Owner* fosse o Rodolfo Barreira, o *Scrum Master* fosse a Cidália Pinto e a equipa de desenvolvimento são o Rodolfo Barreira e a Cidália Pinto.

O *product backlog* do projeto foi o seguinte:

1. Ver os produtos e detalhes dos produtos
2. Ver empresas
3. Ver categorias
4. Fazer o *login* e registo nos vários locais necessários
5. Carrinho de produtos do cliente
6. Ver as faturas das compras
7. Gestão dos vários elementos no *Backoffice*
8. Correção de problemas gerais

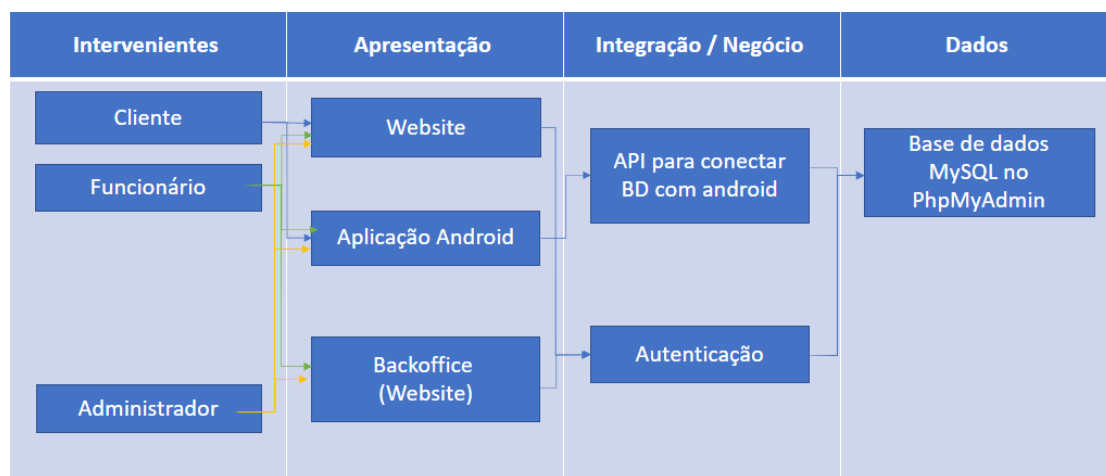
4. Arquitetura do Sistema

A arquitetura do sistema representa a maneira como os vários pontos interagem entre si.

Os nossos intervenientes principais são os clientes, os funcionários e o administrador. O cliente como utilizador tem menos permissões que os restantes, ou seja, acesso ao website (*frontoffice*) e à aplicação *Android*. O funcionário e o administrador têm acesso a mais funcionalidades, tal como utilização do *backoffice* para poderem efetuar a gestão dos seus serviços. A diferença do acesso do funcionário e do administrador é apenas que o funcionário pode utilizar as funcionalidades que dizem respeito à sua empresa, enquanto o administrador tem acesso a todas, tal como a gestão das empresas e criação de utilizadores.

O *website* (*frontoffice*) e o *backoffice* fazem a integração à base de dados através da autenticação enquanto o *android* faz através da *API*.

Como podemos ver na imagem abaixo o nosso projeto tem 3 intervenientes, 3 formas de apresentação, 2 maneiras de integração e 1 base de dados.



1 - Arquitetura do sistema

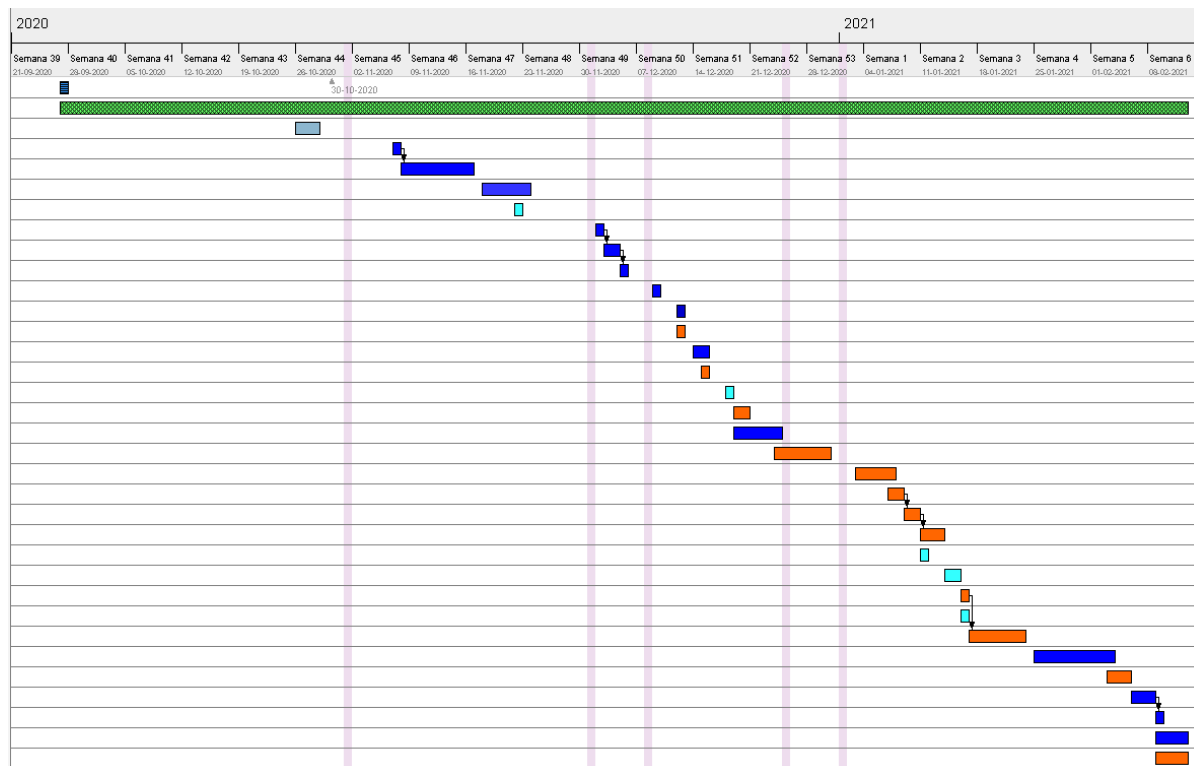
5. Gestão do Projeto

O diagrama de *Gantt* é um gráfico que é utilizado para fazer a gestão temporal de atividades em projetos, no nosso caso este desenvolvimento foi realizado através da aplicação *Gantt Project*.

Neste gráfico estão representadas as atividades realizadas no projeto e será anexado a este documento para melhor visualização. O nome do ficheiro é “*Administration_west GANTT*” e está representado na seguinte imagem, através de uma tabela e de um gráfico.

Nome	Data de início	Data de fim
• Ideia inicial do projeto	27-09-2020	27-09-2020
• Desenvolvimento da base de dados	27-09-2020	12-02-2021
• Desenhos iniciais do projeto	26-10-2020	28-10-2020
• Início do desenvolvimento do website	07-11-2020	07-11-2020
• Login e registo	08-11-2020	16-11-2020
• Início de página de produtos	18-11-2020	23-11-2020
• Desenvolvimento inicial da API	22-11-2020	22-11-2020
• Perfil do utilizador	02-12-2020	02-12-2020
• Início do backoffice	03-12-2020	04-12-2020
• Categorias, detalhes de produto e historico	05-12-2020	05-12-2020
• Atualização do backoffice	09-12-2020	09-12-2020
• Carrinho inicial website(frontoffice)	12-12-2020	12-12-2020
• Início do desenvolvimento de android	12-12-2020	12-12-2020
• Produtos e outras modificações no backoffice	14-12-2020	15-12-2020
• Login e registo básico no android	15-12-2020	15-12-2020
• Métodos de GET para API	18-12-2020	18-12-2020
• Design de inicial das atividades android	19-12-2020	20-12-2020
• Vendas no backoffice	19-12-2020	24-12-2020
• Android produtos	24-12-2020	30-12-2020
• Correção de erros	03-01-2021	07-01-2021
• Android atualização nos designs	07-01-2021	08-01-2021
• Registo e login android	09-01-2021	10-01-2021
• Atualizações android	11-01-2021	13-01-2021
• Api pagamentos	11-01-2021	11-01-2021
• Guardar carrinho e adicionar produtos ao carrinho API	14-01-2021	15-01-2021
• Base de dados em android para produtos	16-01-2021	16-01-2021
• Criação de moradas na API	16-01-2021	16-01-2021
• Compras no android	17-01-2021	23-01-2021
• Criação de utilizadores no backoffice	25-01-2021	03-02-2021
• Historico de vendas android	03-02-2021	05-02-2021
• Contactos frontend e backend	06-02-2021	08-02-2021
• Testes unitarios	09-02-2021	09-02-2021
• Pesquisa de produtos no frontoffice e bug fixes	09-02-2021	12-02-2021
• Terminar android	09-02-2021	12-02-2021

2 - Tabela de atividades dos utilizadores



***Deliverables* (objetivos/ resultados)**

Dentro do objetivo de criação de uma plataforma de comércio *online* a que nos propusemos a fazer, fizemos as atividades que todas as lojas *online* têm, de modo a dar ao utilizador uma boa experiência e às empresas para poderem gerir os seus produtos e atividades.

Para o objetivo que colocámos para este desenvolvimento, conseguimos atingi-lo, pois, a nossa plataforma funciona para os utilizadores e comerciantes.

Milestones

O ponto mais importante no nosso projeto foi a criação do carrinho na aplicação. Até esse momento o utilizador podia ter dois carrinhos, um na aplicação e outro no *website*. O carrinho do *website* utilizava uma biblioteca do *Codeigniter* chamada *Cart*, esta tem funções que auxiliam na criação de carrinhos de compra dos utilizadores. O carrinho do *android* era um carrinho guardado numa base de dados local. Mas numa das reuniões em que tivemos dificuldades nesse desenvolvimento separado, acabámos por chegar à conclusão que seria também mais adequado repensar essa estratégia, sendo então decidido que iríamos criar uma tabela na base de dados *MYSQL* com o propósito de fazer um carrinho partilhado entre as duas plataformas. A maneira como aplicámos isto foi através de pedidos *Post* à *API* no lado da aplicação para colocar o produto no carrinho, e no *website* isto é feito diretamente na base de dados.

Equipa de projeto

A equipa de projeto foi decidida através dos pontos fortes de cada programador. Acabámos por decidir tentar dar ênfase a cada um dos membros do grupo nas suas melhores áreas e ainda assim permitir que ambos os programadores trabalhassem em tudo.

6. Análise

A temática do projeto *Administration West* consiste no desenvolvimento de uma aplicação *web* e *android* para comércio *online* de produtos de empresas locais.

Os objetivos principais do nosso projeto foram desenvolver uma aplicação para apresentar os produtos e também às empresas locais de promover os seus produtos utilizando o comércio *online*. Para o bom funcionamento disto era necessário dar a possibilidade aos clientes de comprarem os artigos e verem os históricos das suas compras. Para isso necessitávamos que os seguintes requisitos fossem cumpridos.

Requisitos funcionais implementados:

- A empresa é registada no *site* através do *backoffice*, ou seja, criadas pelo administrador;
- A empresa faz *login* na plataforma na zona administrativa;
- A empresa adiciona os produtos que tem na plataforma;
- O cliente vê os artigos que as empresas vendem;
- O cliente faz registo na plataforma;
- O cliente faz *login* na plataforma;
- O cliente não precisa de fazer registo nem *login* para ver os produtos das empresas;
- O cliente vê os produtos que quer comprar;
- O cliente coloca os produtos no carrinho;
- O cliente paga os produtos;
- A empresa específica prepara os seus produtos;
- Após todos os produtos sejam preparados são enviados em conjunto.

Requisitos não funcionais implementados:

- Segurança dos dados do utilizador;
- Usabilidade da aplicação e facilidade de uso;
- Desempenho de acesso aos elementos *web* e da aplicação móvel;
- Capacidade de implementar novas funções no futuro.

Requisitos funcionais não implementados

Os requisitos que inicialmente definimos, mas não foram implementados por constrangimentos de tempo, são o pagamento por referências e o envio de faturas por *email* e também o envio de verificação de registo por *email*.

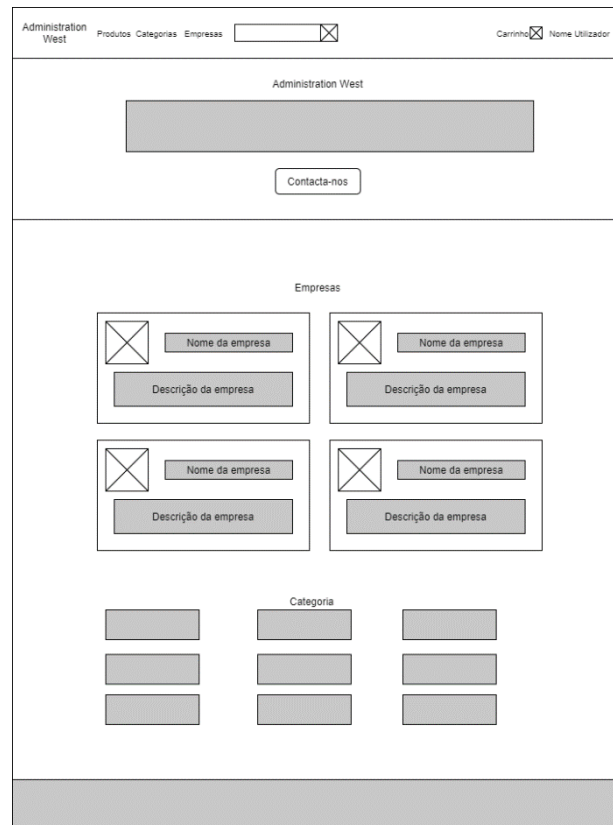
7. Desenho

Protótipos (*wireframes*)

No projeto desenvolvemos vários protótipos de modo a ter uma ideia do que queríamos para o nosso *website*.

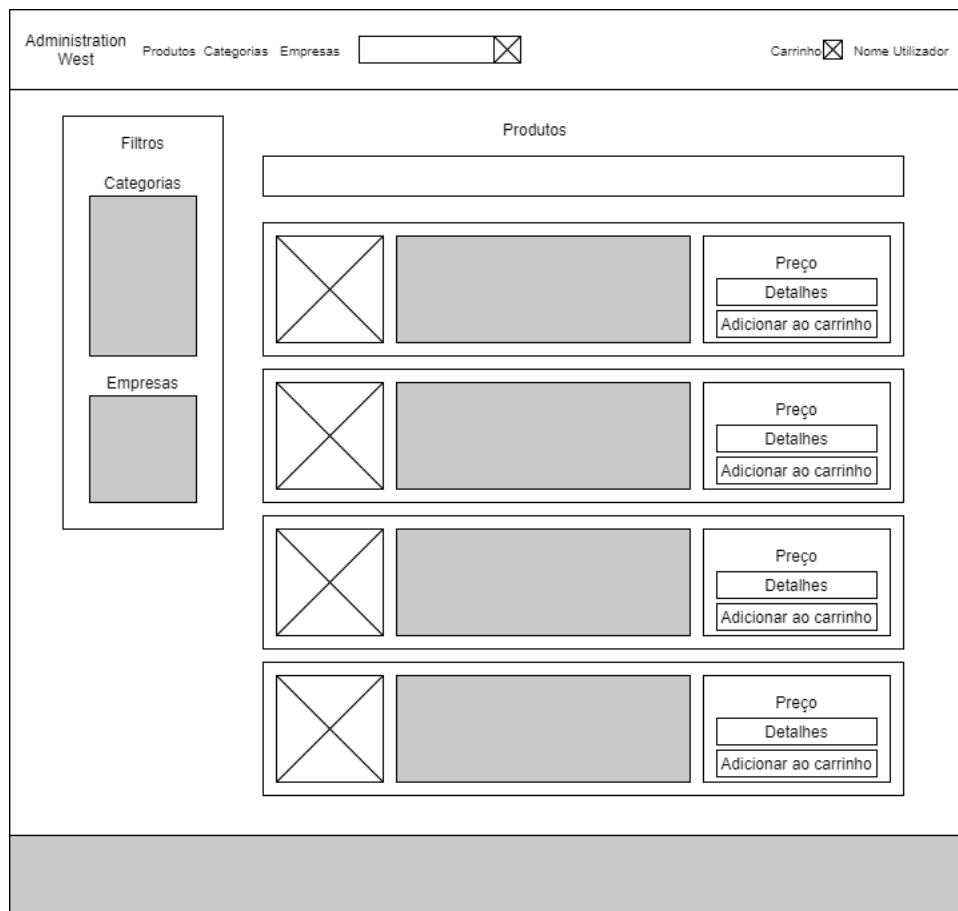
Nos protótipos que iremos demonstrar, temos representados a página inicial, os produtos, os detalhes destes mesmos, o carrinho e o histórico. Na aplicação temos representado o fragmento de produtos, a atividade de detalhes de produtos e o carrinho. De seguida iremos explicar o que cada *wireframe* representa e as suas funcionalidades.

No *website* a primeira página que qualquer utilizador irá observar será o *home*. Por esse motivo existe uma pequena introdução sobre o nosso site, um botão de contactos, uma secção para as empresas e uma secção para as categorias. O botão de contactos direciona o utilizador diretamente para a página de contactos onde o utilizador pode enviar uma mensagem. Caso o utilizador queira observar os produtos de uma categoria ou empresa pode clicar na que escolher e será redirecionado para a página de produtos relacionada. A imagem seguinte representa o que foi mencionado.



4 -Página inicial do website

A página de produtos tem uma secção de filtros, uma barra de pesquisa e pequenas divisões para cada produto. Optámos por ter um filtro de categorias e de empresas. Dentro de cada secção de produto tem uma imagem do produto, um bloco onde contém informações do produto, como o nome, a empresa e a categoria a que pertence, uma pequena descrição e um último bloco onde aparece o preço do produto e dois botões, um de detalhe, que redireciona o utilizador para a página de detalhes do produto, e outro para adicionar o produto ao carrinho. Tal como referido nesta descrição, a imagem seguinte mostra as funcionalidades.



5 - Página de produtos

A página de detalhes é a que contém mais informações sobre o produto. Nesta página pode-se observar que está dividida ao meio, metade tem a imagem e a outra metade tem informações sobre o produto. O *design* da página foi posicionado desta forma para aumentar a imagem e para que o utilizador possa observar as informações do mesmo ao seu lado. Nas informações do produto temos a categoria e a empresa a que pertence, o preço e o botão “Adicionar ao carrinho”, e por último tem a descrição do produto. A seguinte imagem mostra o referido.

Administration West Produtos Categorias Empresas ☒

Carrinho ☒ Nome Utilizador

Nome do produto

Categoria

Empresa

Preço

Descrição

6 - Detalhes do produto

A página do carrinho é a antecessora ao *checkout*, nesta página o cliente pode ver os produtos que pretende comprar, alterar a quantidade ou remover os artigos do carrinho. O *design* desta página é muito simples, uma tabela de produtos que o cliente adicionou, e por baixo uma secção onde mostra os valores a pagar. Em cada linha da tabela de produtos, mostra a imagem, o nome, o preço unitário, o IVA unitário, a quantidade, o total, o IVA total e a ação de remover o produto. O cliente pode alterar a quantidade de cada produto e automaticamente o Total e o Total IVA da linha do produto são alterados para o correspondente. Ao mesmo tempo a secção abaixo onde mostra o total das compras é atualizada.

Aqui o cliente pode tomar a decisão de continuar a comprar ou concluir a compra. Se optar por continuar a comprar será redirecionado para a página de produtos e se preferir concluir a compra será redirecionado para o *checkout*. A imagem a seguir mostra o que foi explicado no texto.

Administration West
Produtos
Categorias
Empresas
☒
Carrinho
☒
Nome Utilizador

Carrinho

Imagem	Nome	Preço (uni)	Iva(uni)	Quantidade	Total	Total IVA	Remover
<input checked="" type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Remover"/>
<input checked="" type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Remover"/>

Subtotal
IVA Total
Total
Continuar a comprar

7- Carrinho do utilizador

A página de histórico permite que o cliente possa visualizar as compras que já efetuou. Esta tem uma tabela em que cada linha representa uma compra já efetuada. Em cada linha podemos observar que aparece o número identificador da compra, o nome da pessoa que vai receber, o NIF, a data de compra, o número de telemóvel, o estado da compra, o preço total, o IVA total e por último um botão que redireciona o cliente para mais detalhes. A ilustração abaixo demonstra a descrição.

Administration
West

Produtos

Categorias

Empresas

☒

Nome Utilizador

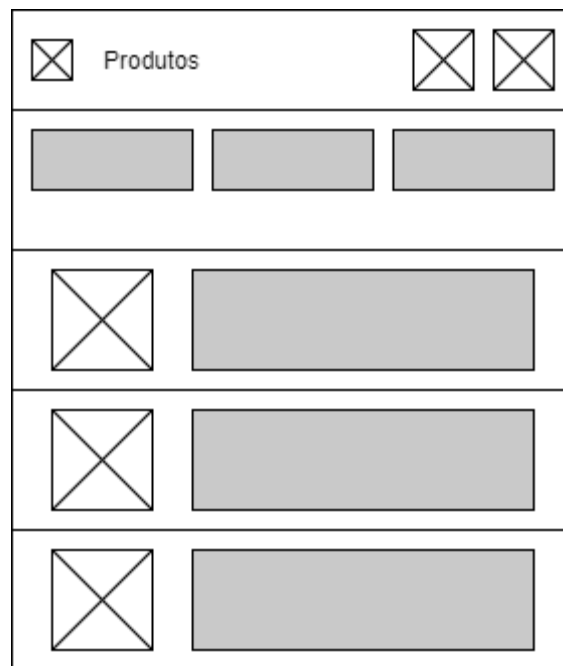
Pedidos

#	Envio	NIF	Data da compra	Número de telemóvel	Estado	Preço Total	IVA Total	Detalhes
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Ver detalhes"/>
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Ver detalhes"/>
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Ver detalhes"/>
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Ver detalhes"/>
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Ver detalhes"/>
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Ver detalhes"/>

8 - Histórico de compras

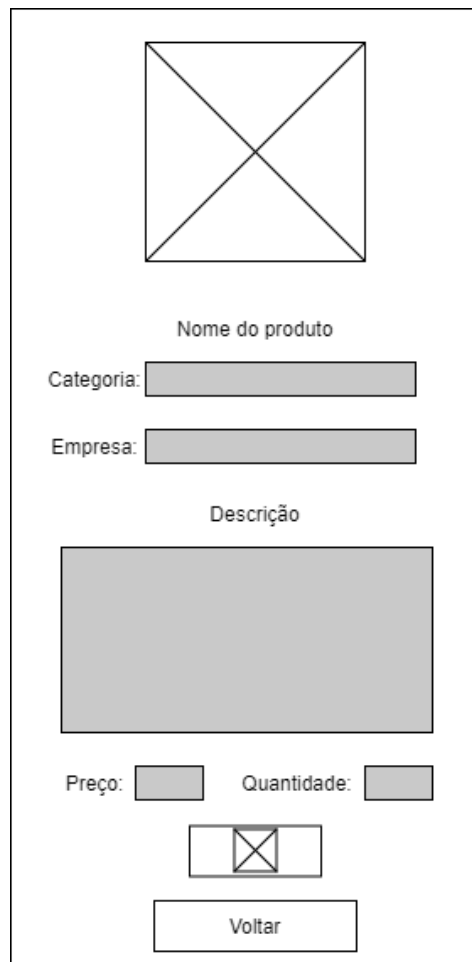
De seguida iremos explicar como é o funcionamento na aplicação.

O cliente inicia numa página de *login*, a qual, depois de efetuado, é apresentado com a página de produtos. Nesta atividade o cliente pode pesquisar o produto que deseja, ir para o carrinho, ver as categorias e por último pode observar os vários produtos. Ao clicar numa categoria o utilizador passa a ver os produtos dessa mesma categoria e ao clicar num produto será redirecionado para os detalhes deste mesmo. A figura abaixo representa a página de produtos.



9 - Atividade de produtos da aplicação

Na atividade de detalhes de produto o utilizador pode observar as informações do produto e adicionar o mesmo ao carrinho. As informações que o cliente tem do carrinho são a imagem, o nome, a categoria e a empresa a que pertence e também a descrição e o preço. A partir destas informações o cliente pode colocar a quantidade que deseja do produto e adicionar o mesmo ao carrinho. Nesta página também existe um botão caso o cliente queira voltar à página inicial. A figura abaixo demonstra o mencionado na descrição.



Nome do produto

Categoria:















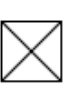

Empresa:

Descrição

Preço: Quantidade:

10 - Detalhes dos produto

Na atividade do carrinho o cliente pode observar os produtos que adicionou, o total da compra e dois botões, um para continuar a comprar e outro para finalizar. Em cada secção de produto o utilizador pode observar a imagem do produto, informações do mesmo e um botão para remover o produto do carrinho. Caso o cliente queira continuar a comprar será redirecionado para a atividade de produtos, ou caso queira finalizar a compra o cliente será redirecionado para a atividade de morada de entrega. A ilustração abaixo representa a explicação.

		
		
		
		
		
<div>Total: </div> <div><div>Continuar a comprar</div><div>Finalizar a compra</div></div>		

11 - Carrinho na aplicação

Casos de uso

O caso de uso fornece o comportamento normal do sistema para uma ação do utilizador. Devido aos casos de uso, podemos fazer o levantamento de requisitos funcionais, não funcionais e de utilização. Os componentes do caso de uso são o nome, o ator, o objetivo, os *triggers*, as pré-condições, as pós-condições, o fluxo básico, as exceções e as qualidades. O nome é a designação do caso de uso, o ator é quem interage com o sistema., o objetivo indica o propósito da realização do caso de uso, os *triggers* são os eventos que dão início ao caso de uso, as pré-condições são os requisitos para que seja realizado, estas devem estar cumpridas antes de iniciar o caso de uso, as pós-condições são as que são cumpridas quando o caso de uso termina, o fluxo básico é a sequência de todos os passos que ocorrem no melhor cenário possível, as exceções são as situações caso não funcione, as qualidades são especificações de qualidade que devem ser cumpridas.

Na imagem abaixo temos um exemplo de caso de uso mais comum entre os utilizadores, comprar produtos.

Caso de Uso	
Nome	Compra de produtos
Ator	Cliente
Objetivo	Permite que os clientes registados no site possam fazer compras dos produtos submetidos.
Triggers	Cliente deseja comprar produtos.
Pré-condições	O cliente já se registou no site. O produto tem que estar catalogado na base de dados. O produto tem que estar ativo. A loja do produto tem que estar ativa.
Pós-condições	O cliente comprou o produto.
Fluxo básico	1) O cliente adiciona o produto ao carrinho. 2) O cliente verifica se a quantidade do produto no carrinho é a desejada. 3) O cliente submete o carrinho. 4) O cliente preenche a morada de envio, a morada de faturação e o método de pagamento. 5) O cliente termina a compra.
Exceções	O produto não tem quantidade suficiente em stock. O cliente cancela a compra.
Qualidades	O cliente tem uma grande variedade de produtos que pode comprar.

12 - Caso de uso

Modelo de dados

As tabelas que existem na base de dados são *user*, *roles*, *permissions*, *permission_assignment*, *categories*, *companies*, *products*, *user_cart*, *billing_address*, *shipping_address*, *sales_group*, *sales_product*, *payment_methods*, *payment_reference* e *contact_form*.

Na tabela *user* são armazenados os utilizadores, a estrutura desta é composta por *id*, *username*, *email*, *phone_number*, *birthday_date*, *password_hash*, *password_reset_token*, *unique_key*, *role_id*, *store_id*, *status*, *created_at*, *update_at*. Esta tabela tem uma relação com a tabela *role* para saber quais as permissões que este tem.

Nos *roles* é onde estão arquivados os tipos de utilizador, é estruturado por *id*, *name*, *description* e *status*.

Nas *permissions* é onde estão arquivadas as permissões que podem ser acedidas no *backoffice*, a estrutura desta tabela é *id*, *name*, *description*, *created_date* e *modified_date*.

No *permission_assignment* é onde estão guardadas as permissões que cada utilizador tem no *backoffice*, e tem ligação à tabela *roles*, para saber que tipo de utilizador pertence a permissão, e também tem ligação à *permissions* para saber que acessos este tem. Esta tabela é constituída por *id*, *role_id*, *permission_id*, *view*, *crud*, *created_date* e *modified_date*.

Na *categories* são classificadas as categorias de produtos, a estrutura da mesma é *id*, *category_name* e *iva*.

Nas *companies* são armazenadas as empresas que têm parceria com a nossa empresa, esta é composta por *id*, *company_name*, *image*, *description*, *status* e *created_date*.

Nos *products* é onde estão catalogados todos os produtos que têm ou tiveram alguma relação com a nossa empresa. Está composta por *id*, *product_name*, *image*, *small_description*, *big_description*, *category_id*, *company_id*, *quantity_in_stock*, *price*, *price_without_iva*, *price_iva*, *status* e *created_date*. Tem ligações à tabela *categories*, para saber a que categoria o produto pertence, e a *companies*, para saber a que empresa o produto pertence.

No *user_cart* são armazenados os produtos e a quantidade que cada utilizador adicionou ao carrinho, isto até o utilizador terminar a compra. É composta por *id*, *user_id*, *product_id* e *quantity*. Tem ligação com *user* e *product*, para saber qual produto pertence a compra e de que utilizador.

No *billing_address* são depositados os dados de faturação de cada cliente que já efetuou uma compra. A estrutura desta é *id, user_id, name, nif, contact_number, city, address, zip_code, created_date, modified_date e status*. A mesma tem ligação com o *user*, para receber o *id* do utilizador.

No *shipping_address* são depositados os dados de envio de cada cliente que já efetuou uma compra. Esta tabela é constituída por *id, user_id, name, nif, contact_number, city, address, zip_code, created_date, modified_date e status*. Tem ligação com o *user* para definir a que utilizador pertence.

No *sales_group* é onde são depositadas as compras, as moradas e o método de pagamento. Esta é formada por *id, user_id, billing_address_id, shipping_address_id, payment_method_id, total_price, total_iva, created_date e status*. Esta tem conexão com *user, billing_address, shipping_address e payment_method*. A conexão com a tabela *user* serve para fazer a conexão ao utilizador, a tabela *billing_address* para identificar a que morada de faturação pertence, a tabela *shipping_address* para indicar a que morada de entrega diz respeito, e *payment_method* para mostrar o método de pagamento escolhido pelo cliente.

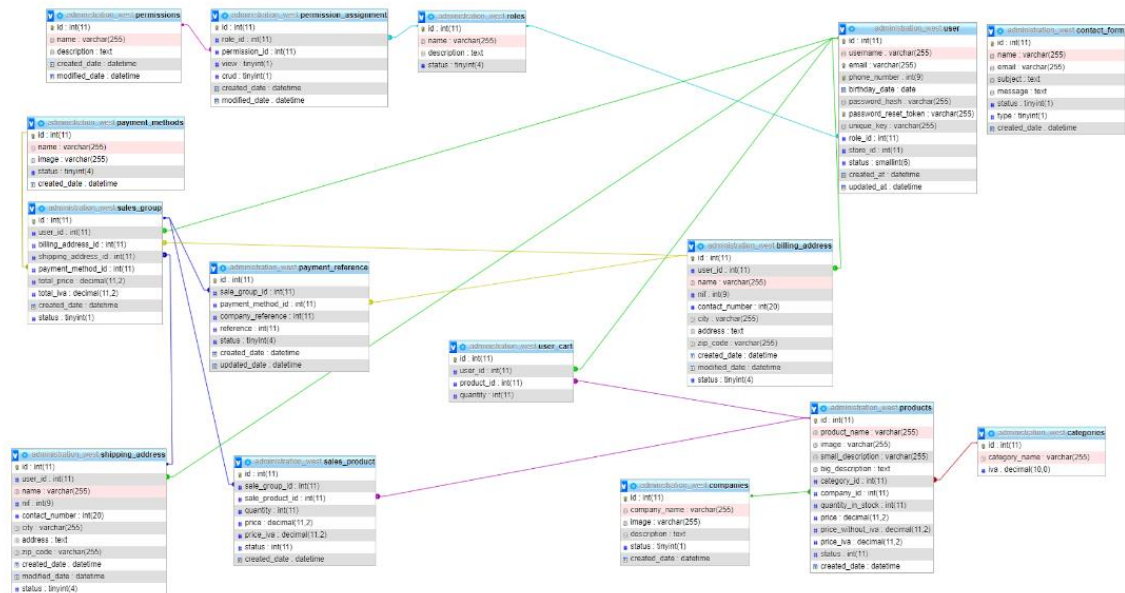
No *sales_product* é onde são depositados os produtos de cada compra e os respetivos valores, esta tabela é constituída por *id, sale_group_id, sale_product_id, quantity, price, price_iva, status e created_date*. Esta tem associação com *sale_group*, para saber-se a que compra pertence, e *products*, para saber-se a que produto esta relacionada.

No *payment_methods* são mantidos os métodos de pagamento. A estrutura da tabela *payment_methods* é *id, name, image, status e created_date*.

No *payment_reference* é planeávamos arquivar as informações sobre referências de pagamento mas acabamos por não implementar. Esta é constituída por *id, sales_group_id, payment_method_id, company_reference, reference, status, created_date e update_date*. Esta comunica com a tabela *sales_group*, para saber a que compra pertence, e a *payment_methods*, para receber o *id* do método de pagamento.

No *contact_form* são recolhidas as mensagens escritas pelos utilizadores para a administração da empresa, esta é constituída por *id, name, email, subject, message, status, type e created_date*.

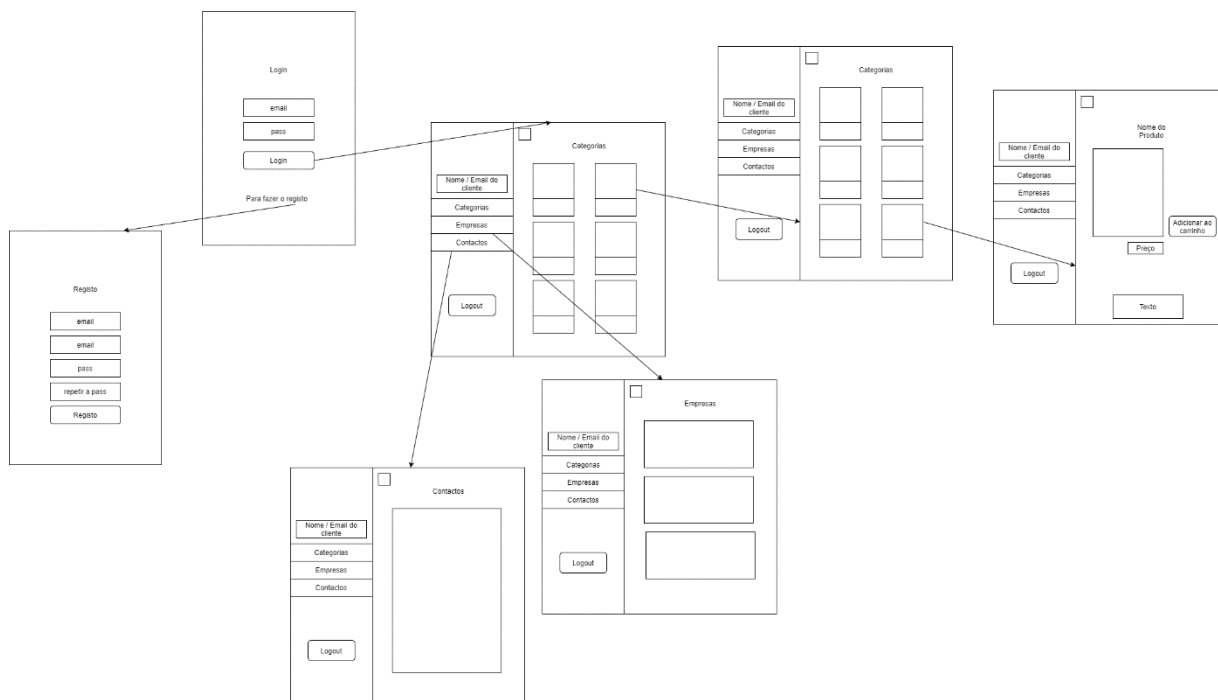
A imagem abaixo descreve a relação que as tabelas têm entre si, descrito anteriormente. Esta imagem será anexada ao relatório para melhor visualização devido ao tamanho desta imagem.



13 - Estrutura das tabelas da base de dados

Mockups

No *mockup* inicial da aplicação *android* o cliente podia ver como primeira janela, o *login* e também botão que direcionava para o registro. Assim que o cliente preenche os seus dados e inicia a sua sessão, vai para o fragmento dos produtos. No fragmento dos produtos o utilizador vê os produtos que a aplicação tem disponíveis. Outros fragmentos que o cliente pode usar são Categorias, Empresas e Contactos. Nas categorias o utilizador pode observar as categorias de produtos. Nas empresas o cliente pode ver as empresas que vendem os seus produtos. Nos contactos o cliente poderá ver os contactos da empresa, como morada, telemóvel e email. Quando o cliente clica em cima de uma categoria isto atualiza os dados para mostrar os produtos respetivos à categoria. Quando o cliente clica em cima de um produto o cliente pode ver os detalhes do produto. Estas funcionalidades mantiveram-se para a atualidade apesar de haver algumas mudanças, tal como alguns métodos novos. A imagem abaixo descreve o texto.



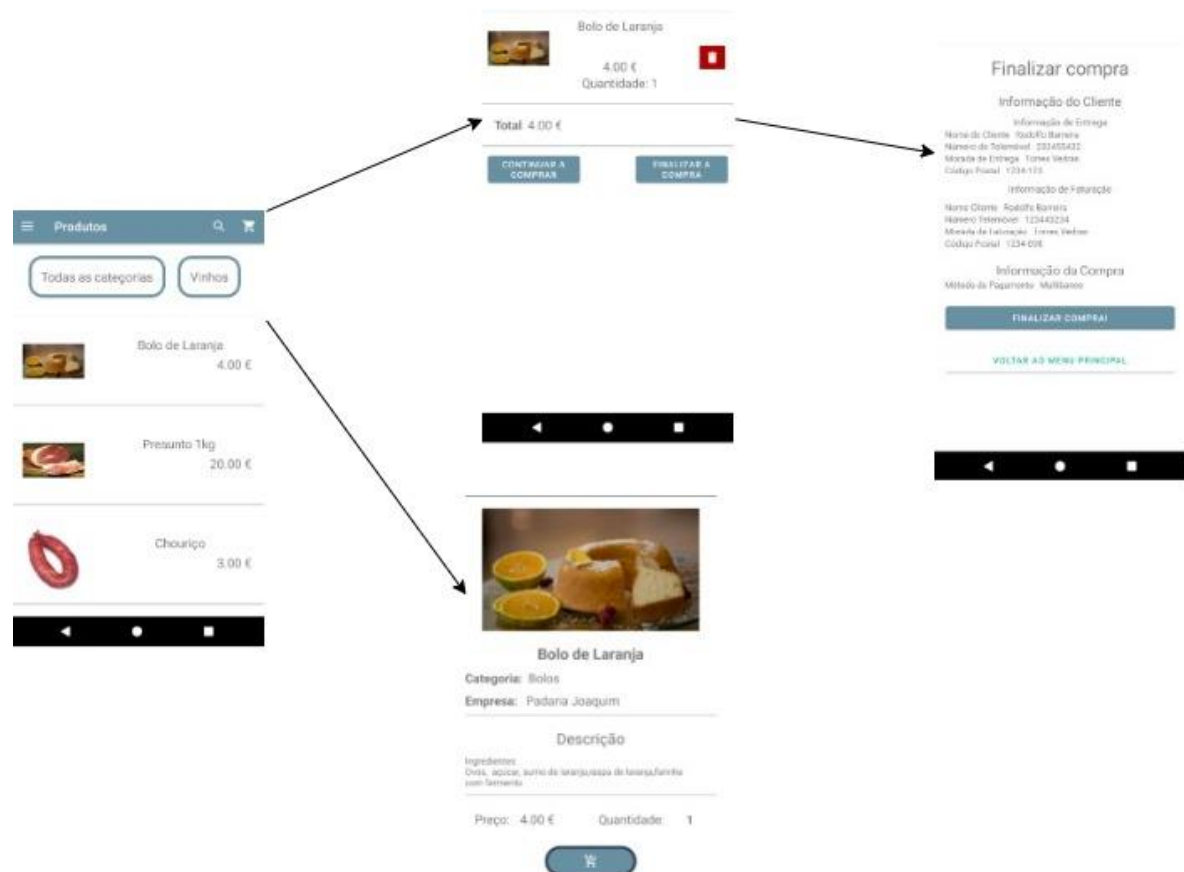
14 - Mockup inicial

Mockup final da aplicação

No *mockup* final da aplicação android, a primeira atividade que o utilizador vê é o login. Nesta aplicação o utilizador também encontra um botão que direciona para a página do registo. Após o *login* o cliente é redirecionado para o fragmento de produtos, onde encontra todos os produtos que a aplicação vende. Neste fragmento o utilizador encontra um método de pesquisa que serve para o cliente encontrar o produto que pretende, e outra de carrinho, que redireciona o cliente para a página de carrinho. Ainda nos produtos o cliente encontra as categorias que existem, e abaixo das categorias, os produtos. Se clicar num dos produtos o cliente é direcionado para os detalhes deste mesmo. Nos detalhes de produto o cliente encontra mais informação sobre o produto e pode escolher a quantidade que deseja comprar e adicionar ao carrinho. Caso não queira tem um botão para voltar à atividade inicial. Na atividade do carrinho o cliente verifica os produtos e encontra dois botões o de continuar a comprar, que direciona o cliente para a atividade inicial, e o finalizar a compra que vai direcionar o cliente para a morada de entrega, onde o mesmo indica a morada a qual pretende receber as compras. Após a morada de entrega o cliente é redirecionado para a atividade de morada de faturação onde indica a morada das faturas. Depois da morada de faturação o cliente é direcionado para a atividade de métodos de pagamento, onde pode escolher qual o método de pagamento que pretende pagar as

compras. Depois de escolher o método de pagamento é encaminhado automaticamente para o checkout. No checkout o cliente encontra algumas informações como as moradas e o método de pagamento utilizados.

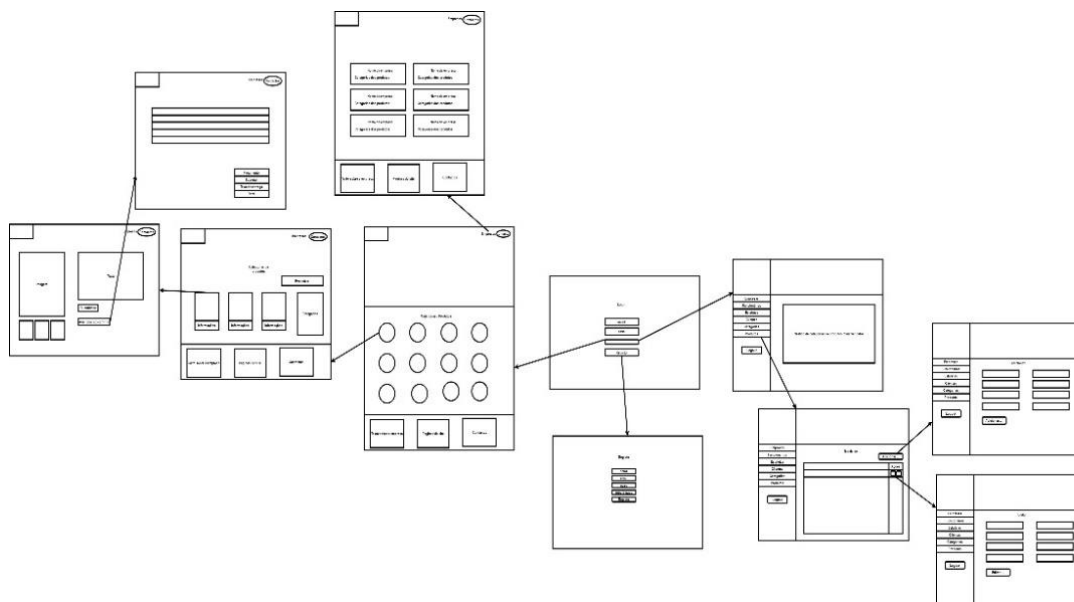
Uma parte da descrição está demonstrada na imagem abaixo.



15 - Estrutura de funcionamento explicada

Mockup inicial do site

No *mockup* inicial da aplicação *web* o cliente quando abria a aplicação via o login e também um botão que direcionava para o registo. Assim que o cliente preenchesse os seus dados e clicasse no botão *login*, era redirecionado para a página inicial. Na página inicial o utilizador via as categorias que a aplicação tinha disponíveis. Nessa página o utilizador podia ir para a página de empresas. Se o utilizador clicasse numa categoria ia para a página de produtos dessa categoria. Se clicasse num produto ia para a página de detalhes de produto onde podia adicionar ao carrinho. Caso fosse um administrador o utilizador começava por ver a página inicial do *backoffice* que tinha uma barra lateral onde podia ver as empresas, funcionários, estafetas, clientes, categorias e produtos. Quando o utilizador clicasse em qualquer um destes iria aparecer a tabela respetiva, onde se podia adicionar, editar e eliminar. A imagem abaixo descreve a descrição.

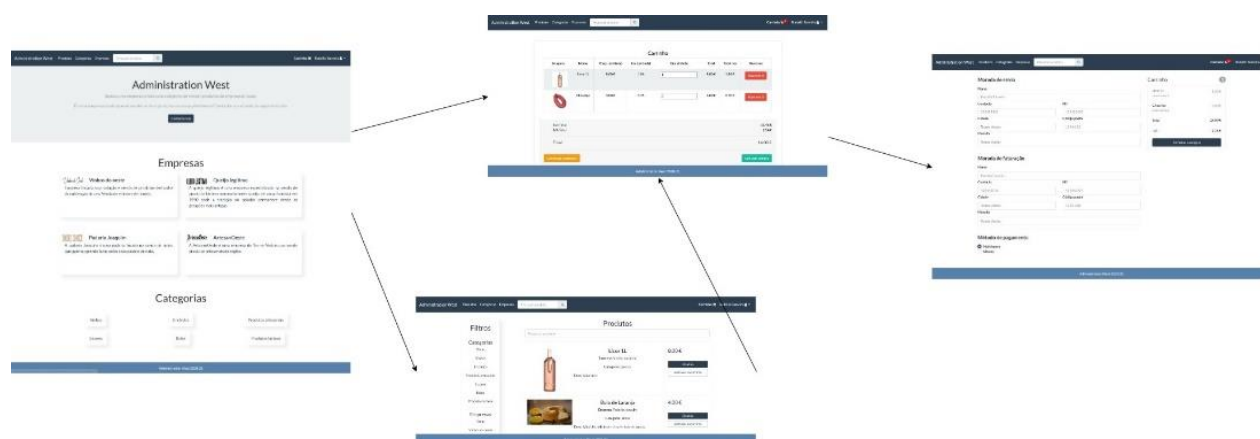


16 - *Mockup inicial do site*

Mockup final do site

No *mockup* final do *site* na área do *frontoffice* a primeira página que o cliente vê apresenta algumas informações como as empresas e as categorias. Ao clicar numa das categorias ou empresas o cliente é direcionado para a página de produtos respectiva. Outra maneira de ir para a página de produtos é clicar diretamente nesta categoria na base superior. Ao escolher um produto para comprar o cliente pode clicar no botão “Adicionar ao carrinho”, que o direciona automaticamente. No carrinho o cliente pode alterar a quantidade do produto que deseja comprar ou eliminá-lo do carrinho. Ainda nesta página o cliente pode tomar a decisão de continuar a comprar ou concluir a compra. Se desejar continuar a comprar o cliente apenas necessita de clicar no botão “Continuar a comprar”, caso deseje concluir a compra apenas tem que clicar no botão “Concluir a compra”. Ao clicar no botão para continuar a comprar será redirecionado para a página de produtos enquanto que se clicar no botão para concluir a compra será direcionado para o checkout. No checkout o cliente tem necessita de preencher ou verificar os seus dados anteriores os quais são, morada de entrega e de faturação, o método de pagamento e para além disso pode ainda verificar os produtos que vai comprar. Quando o cliente desejar concluir apenas precisa de clicar no botão “Terminar a compra”.

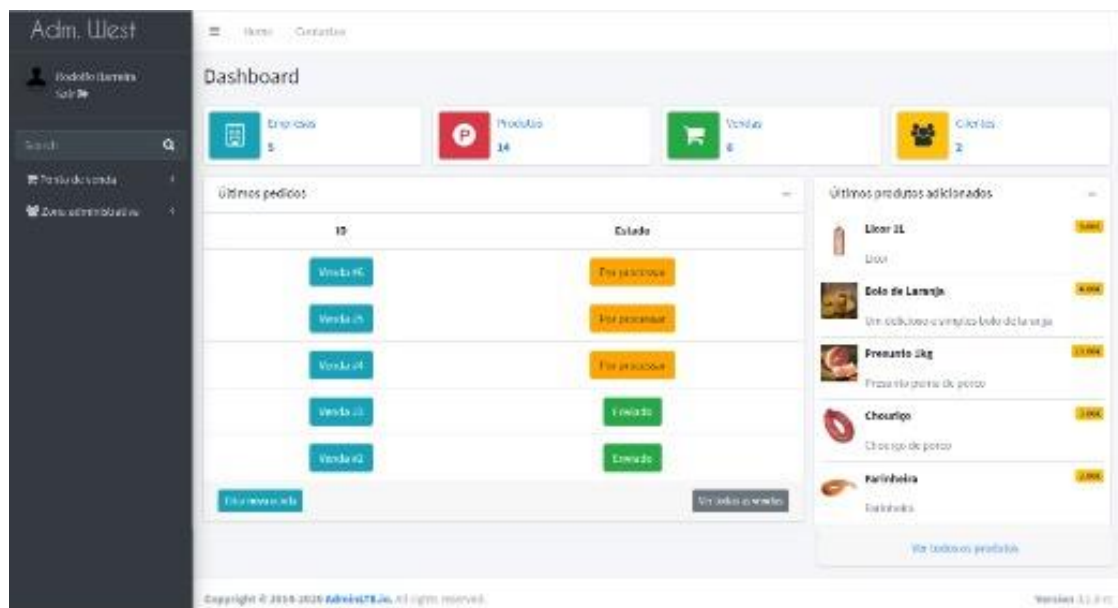
A ilustração abaixo representa uma parte do texto.



17 - Estrutura do frontoffice explicada

Backoffice

No *backoffice* temos acesso à zona administrativa da aplicação, na qual mostra algumas informações especificadas de cada empresa, ou gerais, dependendo do tipo de acesso. Esta página permite direcionar para as páginas restantes. As quais são diferenciadas maioritariamente pelo tipo de informações que contém. Estas utilizam todas *Datatables* para uma boa organização da informação e filtragem sem necessidade de atualização da página.



18 - Dashboard do backoffice

8. Implementação

Para o desenvolvimento do software começámos por fazer os desenhos do projeto, com a aplicação *Draw.io*. Depois de termos os *mockups* decidimos que *framework* iríamos utilizar no projeto de *web* e acabámos por optar pelo *Codeigniter*, devido a ser uma *framework* leve e fácil de implementar e que tem um manual de instruções fácil e com várias funções já incluídas de raiz. A versão que utilizámos do *Codeigniter* foi a versão 3.1.11.

De seguida começámos a desenvolver a base de dados que iria interligar os várias componentes, tanto a aplicação como o website, para isso utilizámos o *PHPMysqlAdmin* para criar tabelas *SQL*. Esta base de dados foi alterada constantemente ao longo do projeto, maioritariamente por necessidade de implementar novas funcionalidades.

Após esse desenvolvimento criámos as bases do *website*, tal como as configurações, implementações de *Bootstrap* e *bootswatch* para os estilos, *Jquery* para o desenvolvimento em *Javascript* simplificado, *FontAwesome* para os ícones customizados, e configurámos também as rotas iniciais, controladores base, e também criámos páginas base de login e registo. Durante esta fase fizemos vários desenvolvimentos estruturais do *website* e iniciámos também o desenvolvimento do perfil, produtos, empresas e categorias. Começámos também então a desenvolver a *API* e o *Backoffice*. No desenvolvimento do *Backoffice* utilizámos um *template* administrativo denominado *AdminLTE* utilizando a versão 3, devido a se enquadrar às nossas necessidades administrativas e por ter um design apelativo e intuitivo. Utilizámos também *Datatables* para as tabelas administrativas que é um *plugin* para tabelas que utiliza *Jquery* e que permite efetuar filtros e carregar as tabelas nas páginas sem necessidade de atualização destas mesmas.

Terminada essa fase iniciámos o desenvolvimento da aplicação *Android* utilizando *JAVA*, começando por algumas atividades base e modelos necessários. Começámos então o *login* e o registo, com materiais de *design* do *Google* para *android*. Nesta fase fomos “saltando” entre o desenvolvimento *web* e a aplicação conforme íamos fazendo funcionalidades na aplicação íamos melhorando-as também no *website*, tal como os produtos, categorias e empresas. Terminado isso começámos a aplicar as conexões da *API* aos respetivos locais, através da biblioteca *Volley* que permite criar acessos à *API*, para poder ter uma boa sincronia de dados.

Conforme realizámos os métodos para adicionar os produtos ao carrinho fizemos também a criação de venda no *website* utilizando ferramentas do *Codeigniter* de criação de carrinho. Após isso decidimos começar a criação destas funcionalidades na aplicação, e foi aqui que encontrámos mais dificuldades, devido a não termos uma forma definida de guardar os dados do carrinho em ambos os lados em simultâneo, ou seja, se um utilizador adicionasse um produto do lado *web* não estaria na aplicação sem ser terminada a venda. Acabámos assim por decidir criar métodos que permitissem isso, substituindo os antigos, ou seja através de comunicação constante cada vez que um produto fosse adicionado ao carrinho, este fosse guardado numa tabela na base de dados *SQL*.

Terminado o método de efetuar compras na aplicação decidimos terminar o *Backoffice* e *Frontoffice* aos poucos, corrigindo problemas e adicionando algumas funcionalidades e detalhes e também tentar terminar os detalhes finais da aplicação, tal como o histórico de compras e edição de *passwords*. Implementámos também nesta fase o *SweetAlerts* ao *website*, que é uma versão melhorada dos alertas predefinidos de *Javascript*.

Para dar por concluído o desenvolvimento decidimos fazer testes intensivos à aplicação, nos quais encontrámos vários problemas no *website* e na aplicação, maioritariamente relacionados com permissões e *design*, os quais resolvemos todos os que encontrámos.

9. Testes

Os testes unitários foram desenvolvidos e executados com uma biblioteca da *framework* *Codeigniter*, o *Unit_test*. Os testes desenvolvidos foram mudar *password*, *login*, adicionar produto ao carrinho, realizar uma compra e mudar o estado desta mesma.

Nas imagens abaixo iremos demonstrar as respostas aos testes desenvolvidos.

No teste de mudança de *password* vemos que a alteração da *password* é executada com sucesso. Neste teste precisamos de enviar o *id* do utilizador, a *password* atual e a *password* para a qual pretende alterar. A imagem abaixo demonstra a resposta ao teste descrito.

Test Name	Mudar a password
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\administration_west\web_codeigniter\application\controllers\tests\Test.php
Line Number	175
Notes	

19 - Resposta ao teste de Mudança de password

No teste de *Login fail* é observado que o teste não é executado com sucesso. Neste teste nós pretendemos mostrar que quando a *password* muda o *login* não pode ser realizado com a *password* anterior. Neste teste precisamos de enviar o *id* do utilizador, o *email* e a *password* antecedente ao teste “Mudar a password”. A ilustração abaixo comprova o descrito.

Test Name	Login Fail
Test Datatype	String
Expected Datatype	String
Result	Failed
File Name	C:\xampp\htdocs\administration_west\web_codeigniter\application\controllers\tests\Test.php
Line Number	181
Notes	

20 - Resposta ao teste de mudança falhada de password

O seguinte teste é o oposto do anterior, neste é constatado que o *login* é executado com sucesso com a *password* alterada. Neste teste é solicitado o *id* do utilizador, o *email* e a *password* alterada com sucesso no teste “Mudar a *password*”. A imagem abaixo representa o teste descrito.

Test Name	Login sucessful
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\administration_west\web_codeigniter\application\controllers\tests\Test.php
Line Number	187
Notes	

21 - Resposta ao teste sucedido de mudança de password

No seguinte teste realizado com sucesso, é verificado que o utilizador vai alterar a *password* outra vez. Neste teste é requisitado o *id* do utilizador, a *password* atual e a *password* para a qual pretende alterar. A figura abaixo descreve o teste.

Test Name	Reset Password
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\administration_west\web_codeigniter\application\controllers\tests\Test.php
Line Number	193
Notes	

22 - Teste de renovação da password

No teste “*Add product to cart*” comprovamos que é possível adicionar um produto ao carrinho com sucesso. Neste teste é pretendido enviar o *id* do utilizador, o *id* do produto e a quantidade desejada do produto. A imagem abaixo relata o texto.

Test Name	Add product to cart
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\administration_west\web_codeigniter\application\controllers\tests\Test.php
Line Number	199
Notes	

23 - Teste add product to cart

No teste “*Create sale*” averiguamos a possibilidade de criar uma compra. Este teste é realizado com sucesso. Neste teste é solicitado o envio do *id* do utilizador e o método de pagamento. A imagem abaixo comprova a descrição.

Test Name	Create sale
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\administration_west\web_codeigniter\application\controllers\tests\Test.php
Line Number	205
Notes	

24 - Teste de criação de venda

No teste “*Change sale status*” é confirmado que é possível modificar o estado da compra, com sucesso. Neste teste é requisitado o *id* do utilizador para saber a última venda. A ilustração abaixo demonstra o teste.

Test Name	Change sale status
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\administration_west\web_codeigniter\application\controllers\tests\Test.php
Line Number	211
Notes	

25 - Teste de mudança de estado

10. Conclusão e trabalho futuro

O trabalho aqui apresentado traduz o desenvolvimento da aplicação ao longo de 5 meses de duração, com o objetivo de desenvolver um website e uma aplicação *android* onde as empresas locais pudessem realizar vendas *online*.

Durante o desenvolvimento do projeto encontramos algumas dificuldades, maioritariamente durante a etapa de criação do carrinho na aplicação móvel, estas dificuldades foram relacionadas com a conexão entre a aplicação e o *website*, devido à sincronia entre ambos. Para além dessas dificuldades houve algumas funcionalidades que acabámos por não implementar devido a alguns constrangimentos de planeamento de tempo. Essas funcionalidades eram uma funcionalidade que permitia preencher informações de pagamento, relacionado com a tabela *payment_reference* que criámos. Pretendíamos também ter um método que enviasse *emails* de confirmação aos utilizadores, tal como no registo, para poder ativar a conta.

Independentemente dessas dificuldades e funcionalidades não implementadas fomos sempre desenvolvendo os vários componentes referidos com uma certa agilidade, iniciando cedo no planeamento e desenhos da aplicação e também na criação de uma base de dados para facilmente começarmos a implementar o *website*. Nessa fase conseguimos também iniciar a *API* e o *Backoffice*, e criar várias funcionalidades de acesso aos dados através da *API*.

No desenvolvimento da aplicação foi onde começámos a atrasar o desenvolvimento e então acabámos por ter algumas dificuldades, mas apesar disso conseguimos progredir e não ficar presos nesses problemas. Para os ultrapassar fomos nos reunindo e discutindo formas de prosseguir, e então conseguimos assim concluir a aplicação com as funcionalidades pretendidas para ter uma aplicação funcional para comércio *online*.

Terminado o desenvolvimento do *backoffice* e a parte de correção de erros no projeto, demos assim este como concluído, e tendo em conta o que foi proposto, julgamos que cumprimos a expectativa e que apesar das dificuldades que tivemos e algumas falhas iniciais, desenvolvemos uma aplicação que cumpria o nosso objetivo. Se no futuro tivéssemos a possibilidade gostaríamos de terminar este projeto e possivelmente torna-lo numa realidade. Assim terminado isto sentimos que este projeto foi uma experiência muito importante para o nosso desenvolvimento como programadores.

11. Webgrafia

<https://bbbootstrap.com/snippets/bootstrap-ecommerce-category-product-list-page-93685579> (BBBootstrap – 10/02/2021)

<https://developer.android.com/docs> (Developers Android - de 15/12/2020 até 13/02/2021) - guia oficial do android

<https://www.draw.io/> (Draw.io - de 26/10/2020 até 29/10/2020) - onde desenhamos os layouts de *android*

https://github.com/Tjay98/administration_west (GitHub – de 27/09/2020 até 14/02/2021) - GitHub do projeto

<https://trello.com/b/Pz1RHaaM/administrationwest> (Trello – de 27/09/2020 até 14/02/2021) - aplicativo de gestão do projeto

<https://fontawesome.bootstrapcheatsheets.com/#> Utilizado constantemente para colocar ícones do fontawesome no website. Última consulta dia 13/02/2021

<https://datatables.net> (Ferramenta *JavaScript* de customização de tabelas. Utilizado para as tabelas do backoffice. Última consulta 30/01/2021)

<https://adminlte.io> (AdminLte 3, última consulta dia 5/02/2021 utilizado como esqueleto para o design administrativo)

<https://codeigniter.com> (Codeigniter última consulta dia 1/02/2021 framework de PHP utilizada)

<https://sweetalert2.github.io> (Alertas de javascript última consulta 10/02/2021)

https://codeigniter.com/userguide3/libraries/unit_testing.html (Codeigniter última consulta 09/02/2021)

12. Guia de instalação

Para instalar e utilizar o projeto é necessário utilizar um cliente de simulação de servidor, que tenha *MYSQL* e serviços *web apache*. No nosso projeto utilizámos o *XAMPP*.

Depois de aberto a aplicação de servidor é necessário importar a base de dados no serviço *MYSQL* utilizado, no nosso caso o *PHPMysqlAdmin*. O nome do ficheiro da base de dados está localizado na pasta das bases de dados e chama-se:

“administration_west base de dados final.sql”

Para colocar a conexão à *API* a funcionar é necessário saber o endereço *IP* local do computador. Para isso, no *Windows* é necessário ir à consola e escrever *IPconfig* e pegar o valor do *IPv4* para ser utilizado na aplicação, no ficheiro *JAVA* no caminho `android/app/src/main/java/com/exemple/administration_west/pages/ProductFragment` no string final chamado *ip* tal como na imagem abaixo.

```
// public static final String ip = "http://192.168.1.67/administration_west/web_codeigniter/";  
public static final String ip = "http://192.168.1.109/administration_west/web_codeigniter/";
```

A versão do *Android* recomendada para utilizar foi a versão 8.0 na *API* 26, devido ao facto de ter sido a utilizada durante o desenvolvimento, de modo a não haver tantas diferenças.

13. Anexos

Com este documento anexamos vários componentes referidos ao longo do relatório.