****

**INSTITUTO POLITÉCNICO DE LEIRIA**

Administration West

RELATÓRIO DE projeto em sistemas de informação

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

Rodolfo Barreira nº 2180714

Cidália Pinto nº2180709

Programação em Sistemas de informação

14 fevereiro 2021

**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](#_Toc64318235)

[1. Dicionário de significados 2](#_Toc64318236)

[2. Introdução 3](#_Toc64318237)

[3. Metodologia 4](#_Toc64318238)

[4. Arquitetura do Sistema 5](#_Toc64318239)

[5. Gestão do Projeto 6](#_Toc64318240)

[6. Análise 9](#_Toc64318241)

[7. Desenho 11](#_Toc64318242)

[8. Implementação 28](#_Toc64318243)

[9. Testes 30](#_Toc64318244)

[10. Conclusão e trabalho futuro 33](#_Toc64318245)

[11. Webgrafia 34](#_Toc64318246)

[12. Guia de instalação 35](#_Toc64318247)

[13. Anexos 36](#_Toc64318248)

**Índice de ilustrações**

[1 - Arquitetura do sistema 5](#_Toc64317961)

[2 - Tabela de atividades dos utilizadores 6](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317962)

[3 - Gráfico das atividades 7](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317963)

[4 -Página inicial do website 11](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317964)

[5 - Página de produtos 12](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317965)

[6 - Detalhes do produto 13](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317966)

[7- Carrinho do utilizador 14](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317967)

[8 - Histórico de compras 15](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317968)

[9 - Atividade de produtos da aplicação 16](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317969)

[10 - Detalhes dos produto 17](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317970)

[11 - Carrinho na aplicação 18](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317971)

[12 - Caso de uso 19](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317972)

[13 - Estrutura das tabelas da base de dados 22](#_Toc64317973)

[14 - Mockup inicial 23](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317974)

[15 - Estrutura de funcionamento explicada 24](#_Toc64317975)

[16 - Mockup inicial do site 25](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317976)

[17 - Estrutura do frontoffice explicada 26](https://myipleiria-my.sharepoint.com/personal/2180709_my_ipleiria_pt/Documents/Projeto/Modelo_Relatório_Final%20_ProjetoSistemasInformação.docx#_Toc64317977)

[18 - Dashboard do backoffice 27](#_Toc64317978)

[19 - Resposta ao teste de Mudança de password 30](#_Toc64317979)

[20 - Resposta ao teste de mudança falhada de password 30](#_Toc64317980)

[21 - Resposta ao teste sucedido de mudança de password 31](#_Toc64317981)

[22 - Teste de renovação da password 31](#_Toc64317982)

[23 - Teste add product to cart 31](#_Toc64317983)

[24 - Teste de criação de venda 32](#_Toc64317984)

[25 - Teste de mudança de estado 32](#_Toc64317985)

# 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.

# 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.

# 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

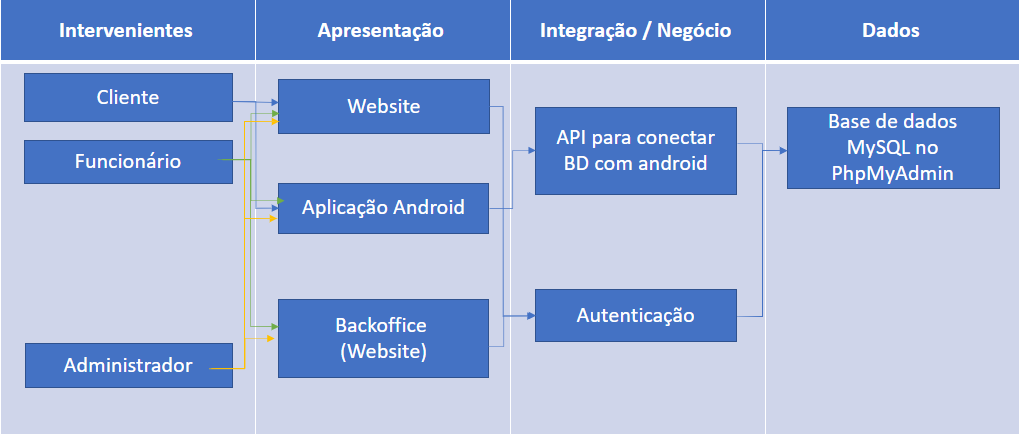
# Arquitetura do Sistema

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.

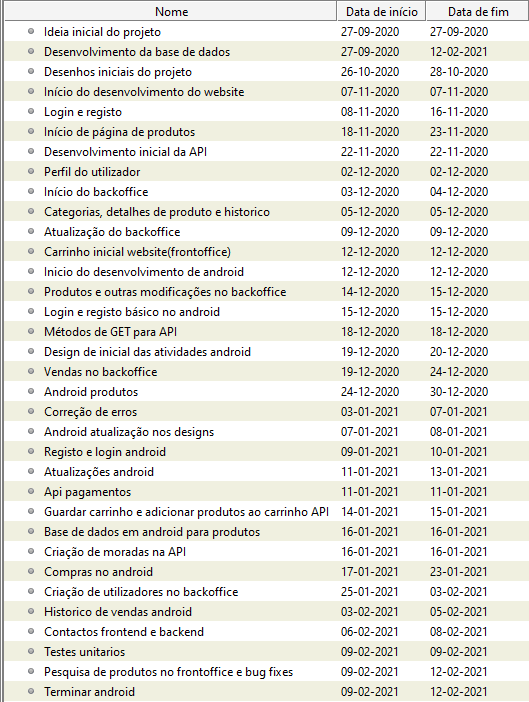


1 - Arquitetura do sistema

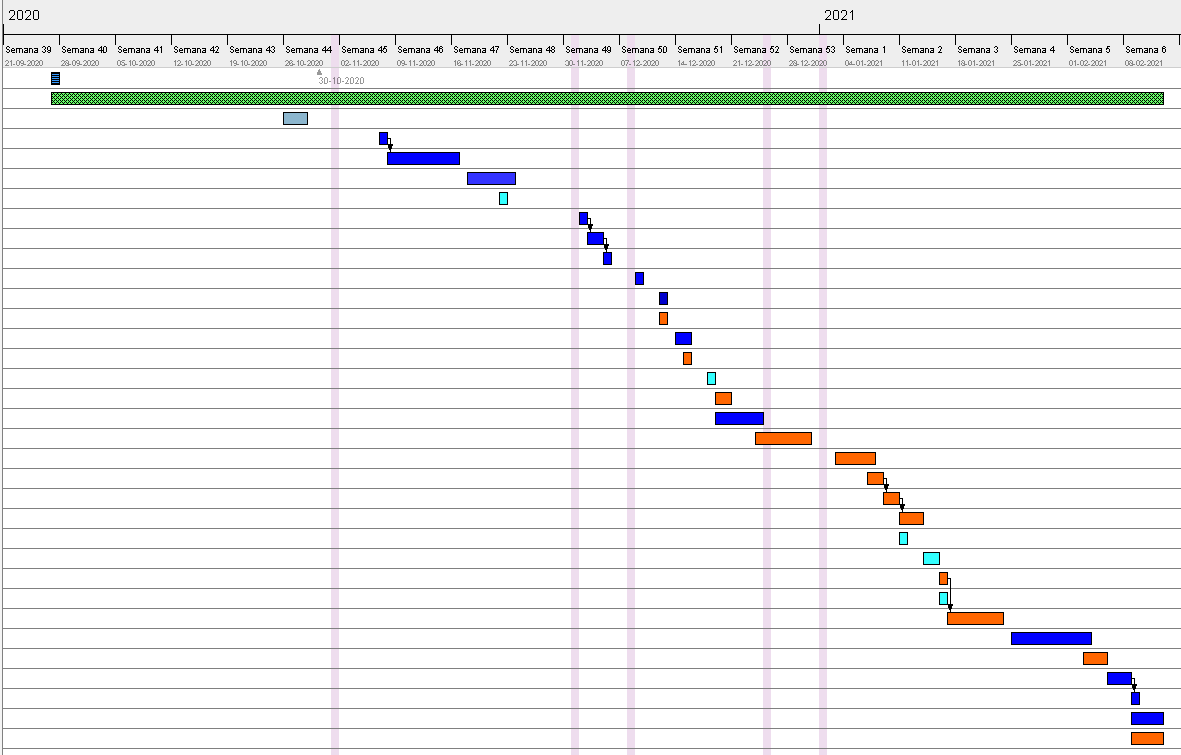
# Gestão do Projeto

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.



2 - Tabela de atividades dos utilizadores



3 - Gráfico das atividades

As principais atividades realizadas no site no ambiente do *frontoffice*, são mostrar produtos, empresas, categorias de produtos, o carrinho, compras, mostrar histórico de compras, perfil do utilizador e permitir a este mudar a *password*. Para além disso é também no *frontoffice* que permite aos utilizadores e empresas poderem contactar a nossa empresa.

No site na área do *backoffice* as atividades realizadas foram de modo a poder executar os métodos *CRUD* aos produtos, empresas, categorias, vendas, clientes, de modo a facilitar o funcionamento administrativo.

Por fim, as atividades na aplicação foram concebidas para mostrar produtos, empresas, categorias, carrinho, criação de compras, histórico, perfil do utilizador e também mudar *password*, tal como no *frontoffice*.

***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 iriamos 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 enfâse a cada um dos membros do grupo nas suas melhores áreas e ainda assim permitir que ambos os programadores trabalhassem em tudo.

# 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**

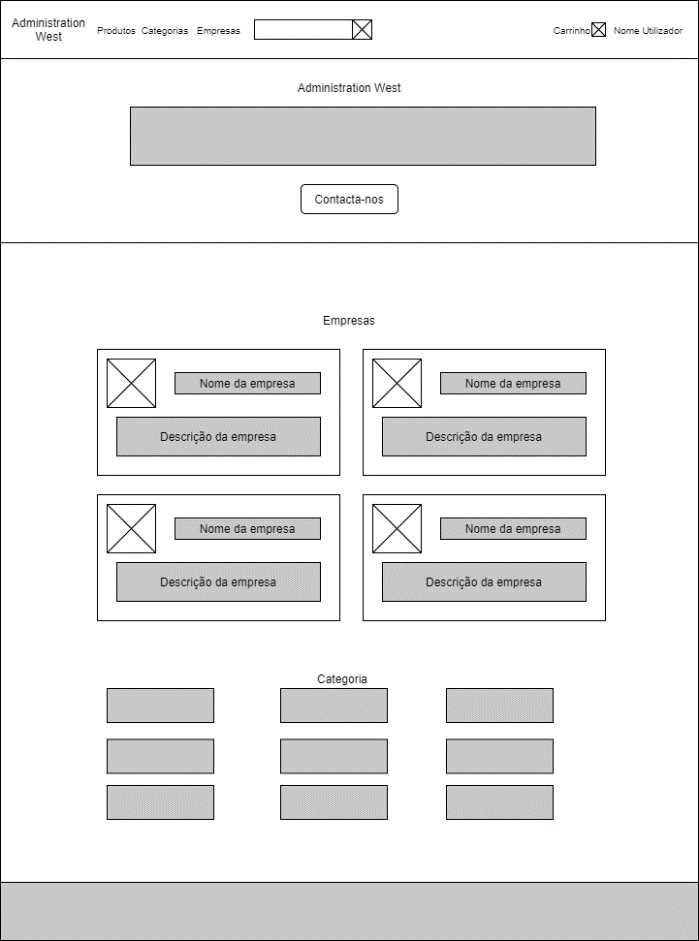
Os requisitos que inicialmente definimos, mas não foram implementados por constrangimentos de tempo, são o pagamento por referências e o envio de faturas por *email* e também o envio de verificação de registo por *email*.

# Desenho

**Protótipos (*wireframes*)**

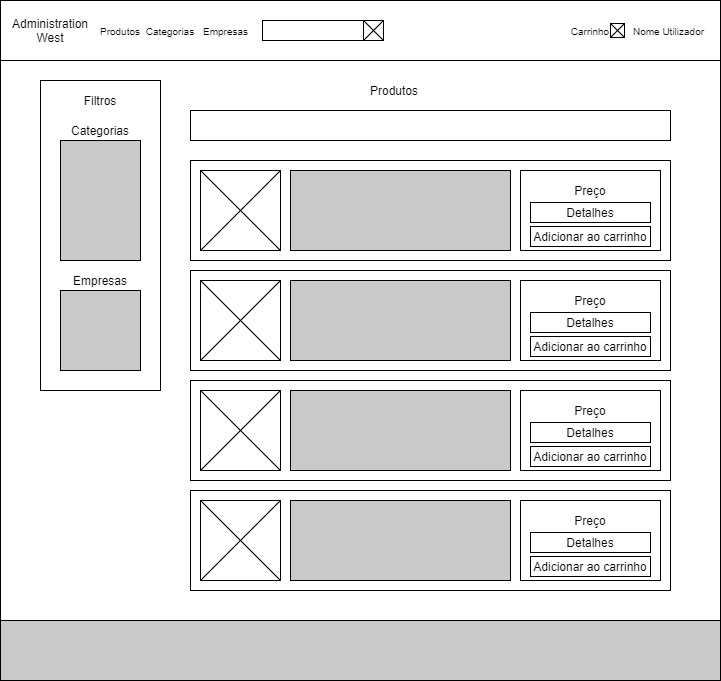
No projeto desenvolvemos vários protótipos de modo a ter uma ideia do que queríamos para o nosso *website*.

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.

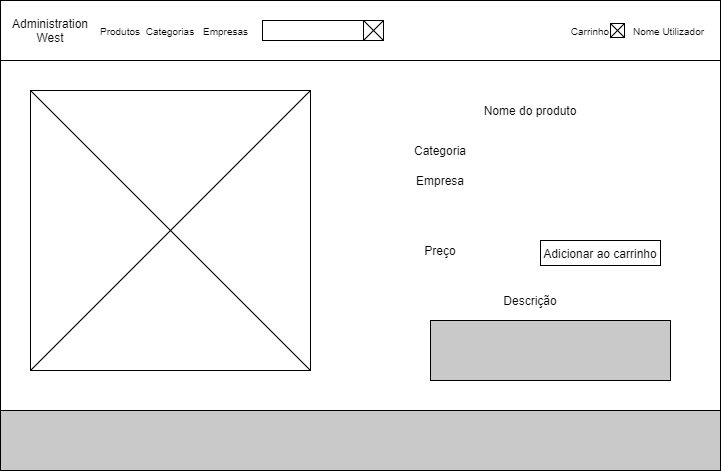
4 -Página inicial do website

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

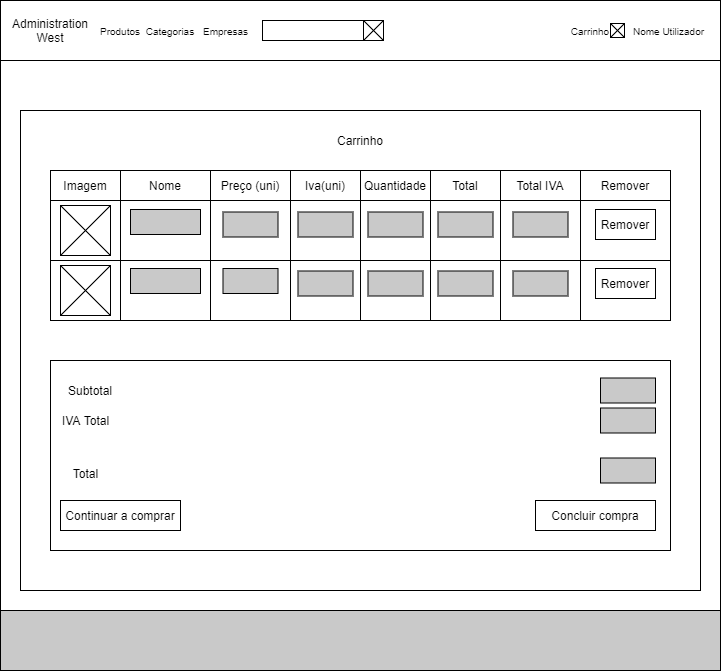
A página de detalhes é a que contém mais informações sobre o produto. Nesta página pode-se observar que está dividida ao meio, metade tem a imagem e a outra metade tem informações sobre o produto. O *design* da página foi posicionado desta forma para aumentar a imagem e para que o utilizador possa observar 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.



6 - Detalhes do produto

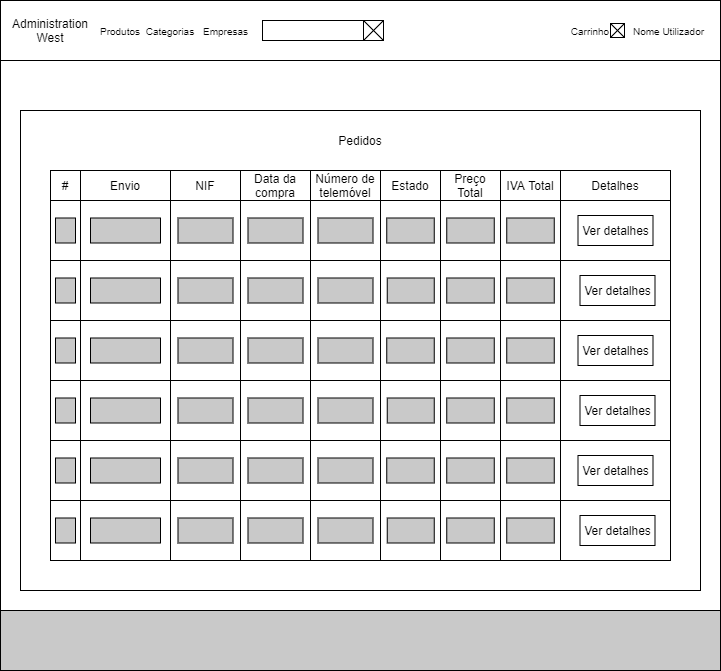
A página do carrinho é a antecessora ao *checkout*, nesta página o cliente pode ver os produtos que pretende comprar, alterar a quantidade ou remover os artigos do carrinho. O *design* desta página é muito simples, uma tabela de produtos que o cliente adicionou, e por baixo uma secção onde mostra os 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.



7- Carrinho do utilizador

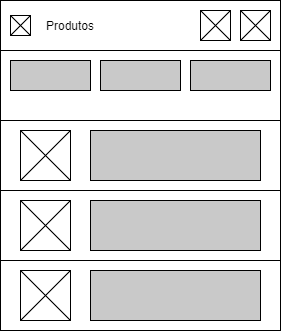
A página de histórico permite que o cliente possa visualizar as compras que já efetuou. Esta tem uma tabela em que cada linha representa uma compra já efetuada. Em cada linha podemos observar que aparece o número identificador da compra, o nome da pessoa que vai receber, o NIF, a data de compra, o número 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.



8 - Histórico de compras

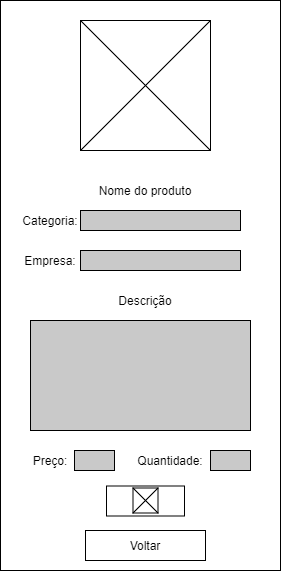
De seguida iremos explicar como é o funcionamento na aplicação.

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.



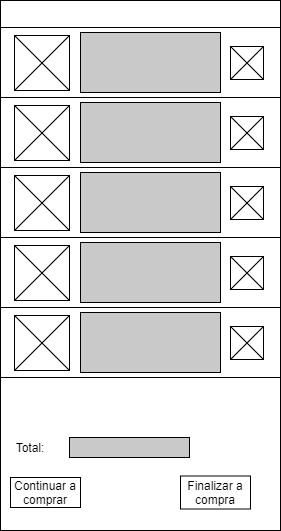
9 - Atividade de produtos da aplicação

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.



10 - Detalhes dos produto

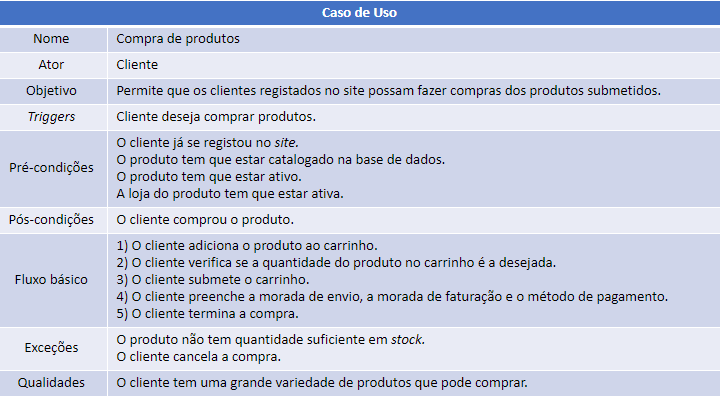
Na atividade do carrinho o cliente pode observar os produtos que adicionou, o total da compra e dois botões, um para continuar a comprar e outro para finalizar. Em cada secção de produto o utilizador pode observar a imagem do produto, informações do mesmo e um botão para remover o produto do carrinho. 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.



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.

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 s*ales\_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.

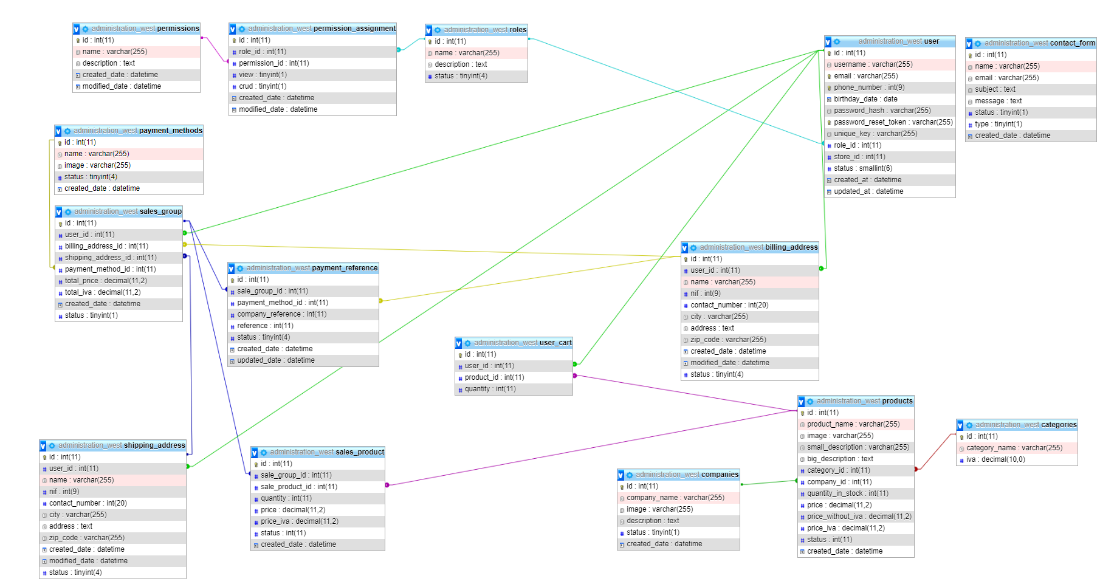
No *sales\_product* é onde são depositados os produtos de cada compra e os respetivos valores, esta tabela é constituída por *id, sale\_group\_id, sale\_product\_id, quantity, price, price\_iva, status e created\_date*. Esta tem associação com *sale\_group*, para saber-se a que compra pertence, e *products*, 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\_method*s, 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.*

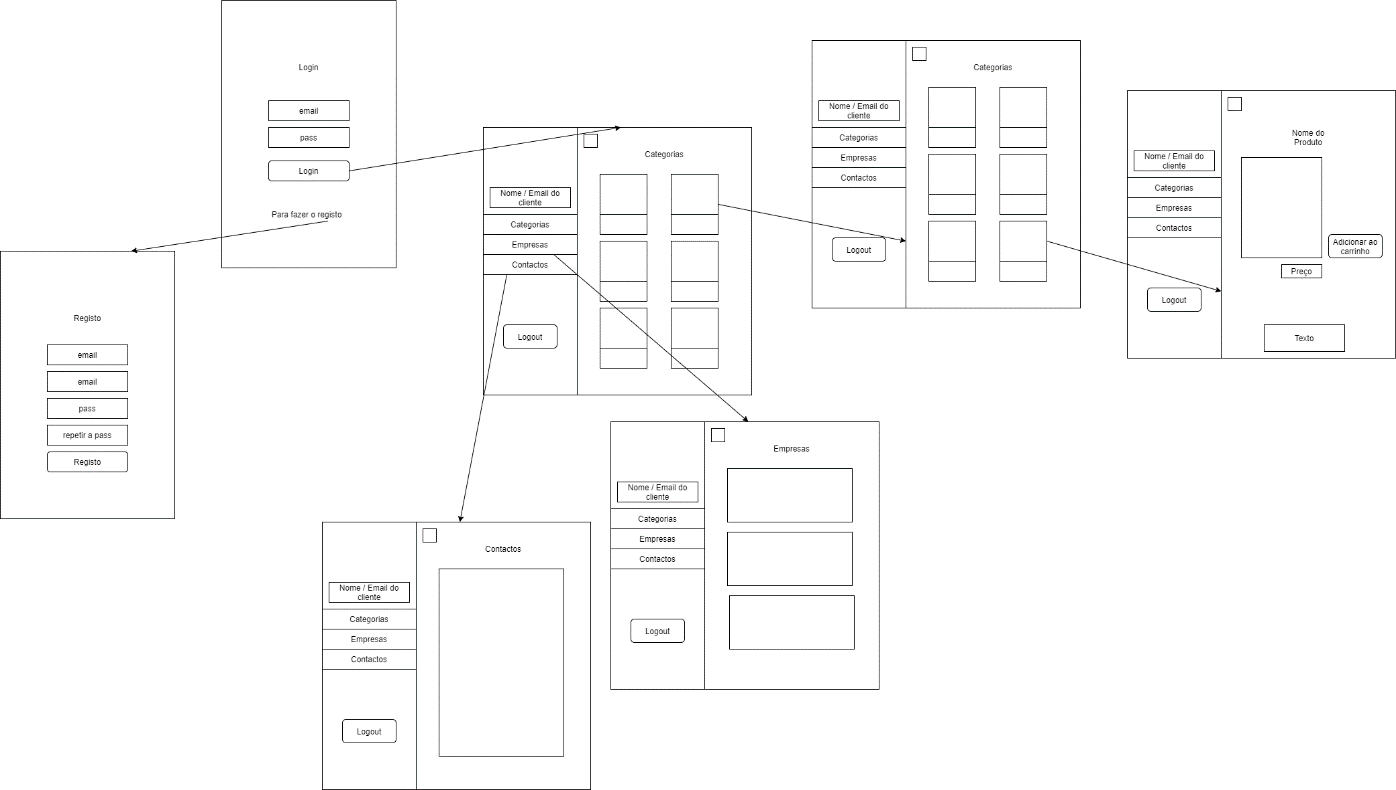
A imagem abaixo descreve a relação que as tabelas têm entre si, descrito anteriormente. Esta imagem será anexada ao relatório para melhor visualização devido ao tamanho desta imagem.



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 registo. 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.

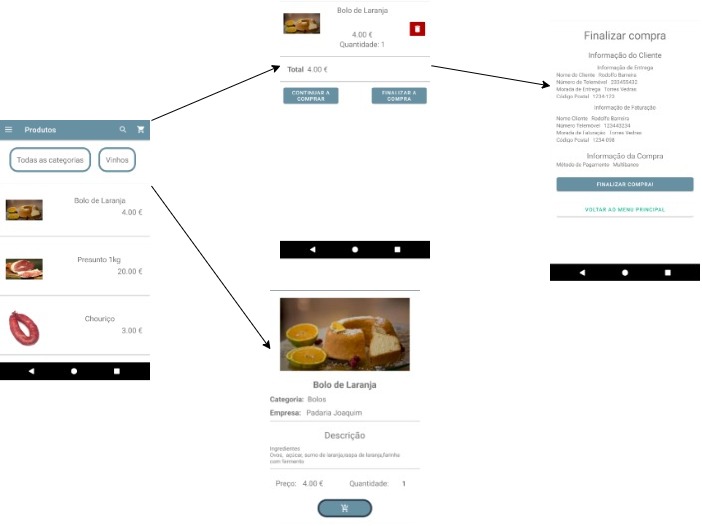


14 - Mockup inicial

***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.

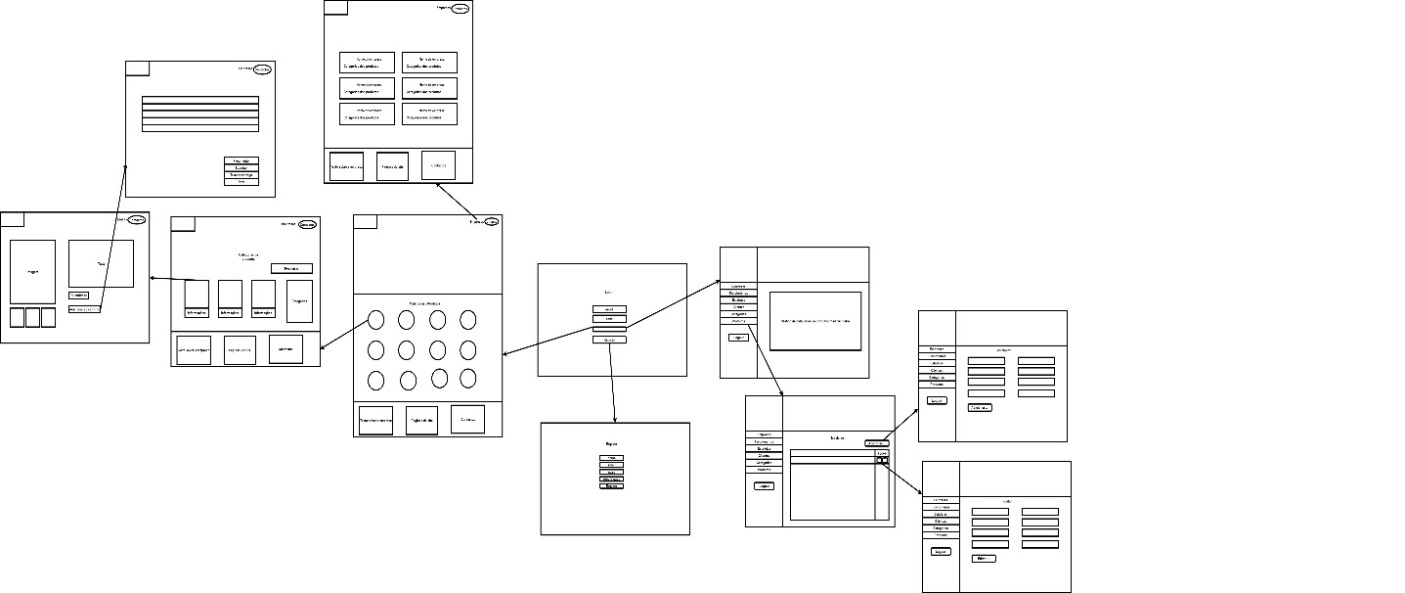
Uma parte da descrição está demonstrada na imagem abaixo.



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.

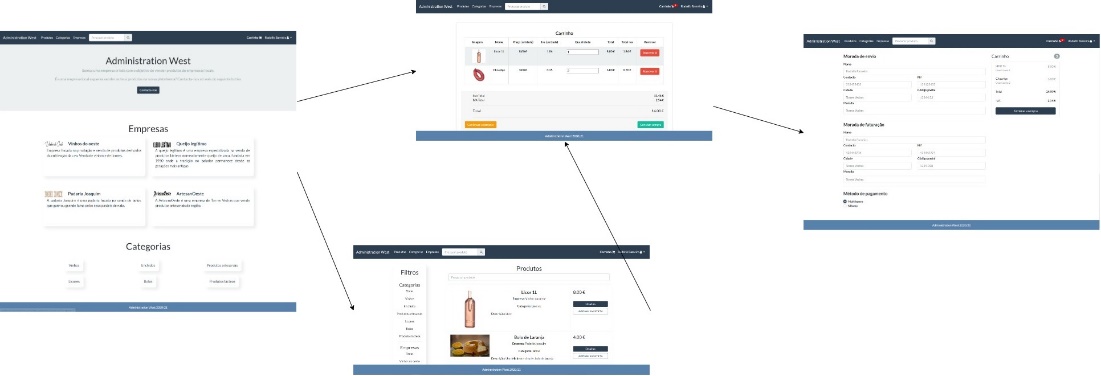


16 - Mockup inicial do site

***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 respetiva. Outra maneira de ir para a página de produtos é clicar diretamente nesta categoria na basse 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”.

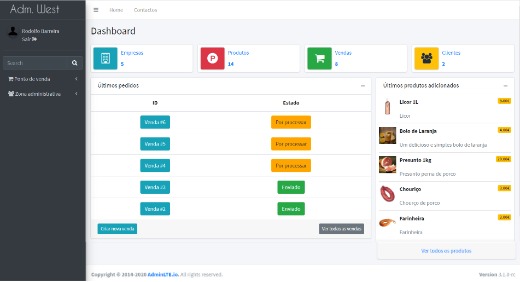
A ilustração abaixo representa uma parte do texto.



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

# 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* iriamos 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 *PHPMyAdmin* 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*.

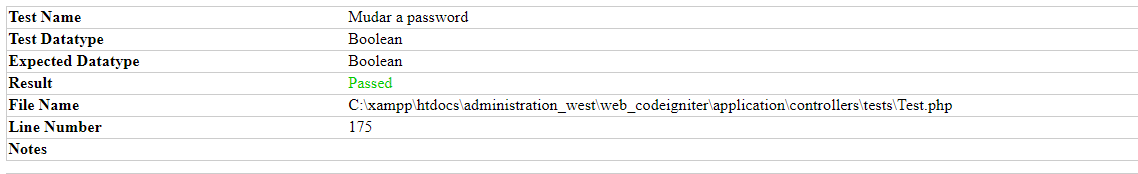
Para dar por concluído o desenvolvimento decidimos fazer testes intensivos à aplicação, nos quais encontrámos vários problemas no *website* e na aplicação, maioritariamente relacionados com permissões e *design*, os quais resolvemos todos os que encontrámos.

# 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.

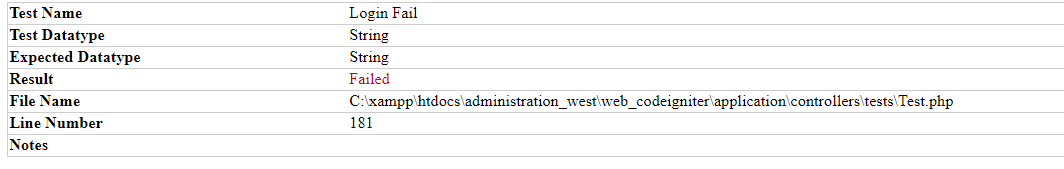
Nas imagens abaixo iremos demonstrar as respostas aos testes desenvolvidos.

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.



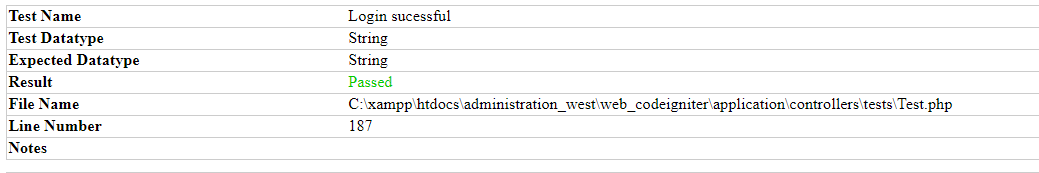
19 - Resposta ao teste de Mudança de password

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.



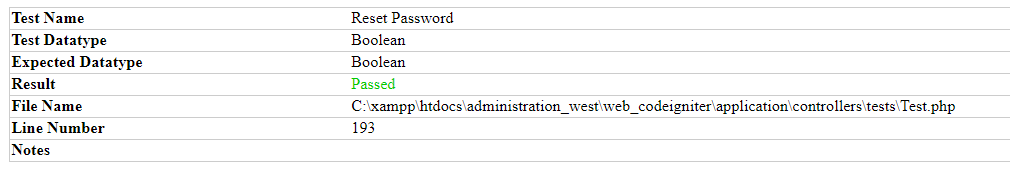
20 - Resposta ao teste de mudança falhada de password

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.



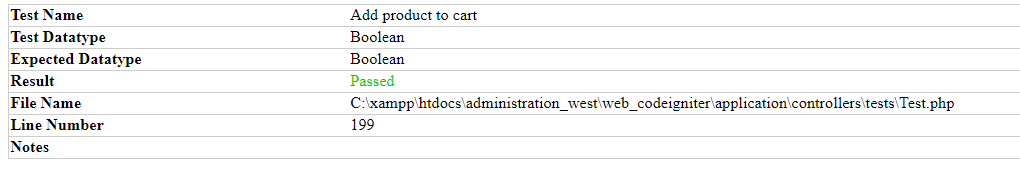
21 - Resposta ao teste sucedido de mudança de password

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.



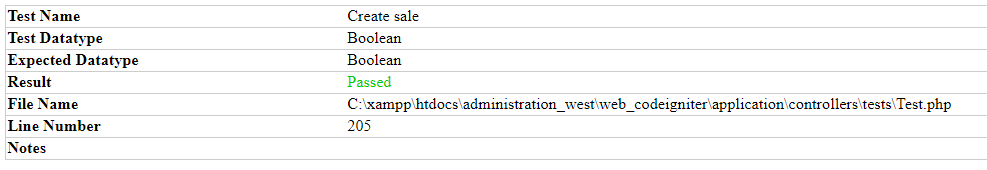
22 - Teste de renovação da password

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.



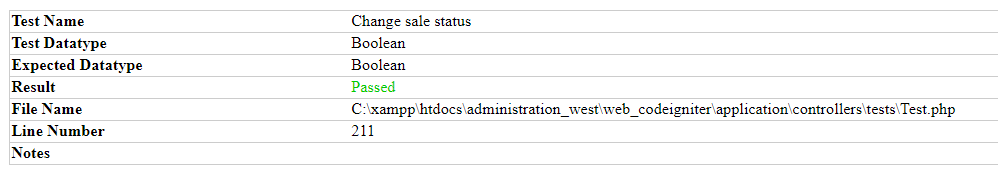
23 - Teste add product to cart

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.



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.



25 - Teste de mudança de estado

# 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 encontrámos 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 gostávamos 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.

# Webgrafia

<https://bbbootstrap.com/snippets/bootstrap-ecommerce-category-product-list-page-93685579> (BBBotstrap – 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/#](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)

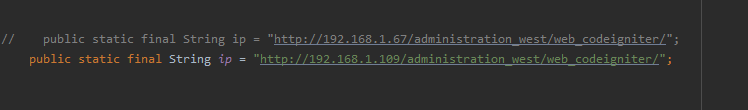
# Guia de instalação

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 *PHPMyAdmin*. 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.



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.

# Anexos

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