

**Instituto Superior de Tecnologias Avançadas do Porto**

**AMF Shoes Platform**

**Order Management Web Application**

**Sergii Shtachenko (726) | Pedro Soares (735)**

**Tecnologias da Internet IV**

**Licenciatura em Informática**

**2020**

**Table of Contents**

[**1.** **Software Project Description** 4](#_Toc44024821)

[**1.1.** **Project overview** 4](#_Toc44024822)

[**1.2.** **Purpose of the project** 4](#_Toc44024823)

[1.2.1. Main project goals 4](#_Toc44024824)

[1.2.2. Project development measurement 4](#_Toc44024825)

[**1.3.** **Project background and scope** 4](#_Toc44024826)

[1.3.1. Current situation 4](#_Toc44024827)

[1.3.2. Business events and workflow 4](#_Toc44024828)

[1.3.3. Competing products and benchmarking 4](#_Toc44024829)

[**1.4.** **Mockups and prototypes** 4](#_Toc44024830)

[1.4.1. Product scenarios 5](#_Toc44024831)

[1.4.2. User stories and scenarios 5](#_Toc44024832)

[**1.5.** **Stakeholders** 6](#_Toc44024833)

[1.5.1. Development team 6](#_Toc44024834)

[1.5.2. Client 6](#_Toc44024835)

[1.5.3. Customer type 6](#_Toc44024836)

[**1.6.** **General constraints** 6](#_Toc44024837)

[1.6.1. Solution constraints 6](#_Toc44024838)

[1.6.2. Schedule constraints 6](#_Toc44024839)

[1.6.3. Budget constraints 6](#_Toc44024840)

[**1.7.** **Conventions and definitions** 7](#_Toc44024841)

[1.7.1. Key terms 7](#_Toc44024842)

[1.7.2. Unified Modelling Language (UML) 7](#_Toc44024843)

[**2.** **Software Requirements** 8](#_Toc44024844)

[**2.1.** **Actors and use cases** 8](#_Toc44024845)

[**2.1.1.** Actors 8](#_Toc44024846)

[**2.1.2.** Use case list 8](#_Toc44024847)

[**2.1.3.** Use case diagrams 9](#_Toc44024848)

[**2.2.** **Domain and functional requirements** 10](#_Toc44024849)

[**2.2.1.** Detailed User-Stories 10](#_Toc44024850)

[**2.3.** **Non-functional requirements** 11](#_Toc44024851)

[**2.3.1.** Database requirements 11](#_Toc44024852)

[**2.3.2.** Performance requirements 11](#_Toc44024853)

[**2.3.3.** Capacity and storage requirements 11](#_Toc44024854)

[**2.3.4.** Security requirements 11](#_Toc44024855)

[**2.3.5.** Scalability and extensibility requirements 11](#_Toc44024856)

[**2.3.6.** Maintenance requirements 11](#_Toc44024857)

[**2.3.7.** Robustness or fault-tolerance requirements 11](#_Toc44024858)

[**2.3.8.** Dependability requirements 11](#_Toc44024859)

[**2.3.9.** Reliability requirements 11](#_Toc44024860)

[**2.3.10.** Availability requirements 12](#_Toc44024861)

[**2.3.11.** Privacy requirements 12](#_Toc44024862)

[**2.3.12.** Usability requirements 12](#_Toc44024863)

[**2.3.13.** Accessibility requirements 12](#_Toc44024864)

[**2.3.14.** Compliance Requirements 12](#_Toc44024865)

[**2.3.15.** Standards Requirements 12](#_Toc44024866)

[**2.3.16.** User documentation requirements 12](#_Toc44024867)

[**2.3.17.** Internationalization requirements 12](#_Toc44024868)

[**2.3.18.** Training requirements 12](#_Toc44024869)

[**2.3.19.** Cultural and political requirements 12](#_Toc44024870)

[**2.3.20.** Legal requirements 12](#_Toc44024871)

[**2.3.21.** Release requirements 12](#_Toc44024872)

[**3.** **Software Design** 13](#_Toc44024873)

[**3.1.** **Existing software architecture** 13](#_Toc44024874)

[**3.2.** **Design goals** 13](#_Toc44024875)

[**3.3.** **Proposed software architecture** 13](#_Toc44024876)

[**3.4.** **Algorithms and Data Structures** 13](#_Toc44024877)

[**3.4.1.** Data structures 13](#_Toc44024878)

[**3.4.2.** Algorithms for problem solving 14](#_Toc44024879)

[**3.5.** **Object oriented analysis and design** 14](#_Toc44024880)

[**3.5.1.** Modular design and encapsulation 14](#_Toc44024881)

[**3.5.2.** Class diagrams 14](#_Toc44024882)

[**3.5.3.** Packages and class interfaces 15](#_Toc44024883)

[**3.6.** **User Interface Design** 15](#_Toc44024884)

[**3.6.1.** Interface design models 15](#_Toc44024885)

[**3.6.2.** Interface design process 16](#_Toc44024886)

[**4.** **Software Construction** 17](#_Toc44024887)

[**4.1.** **Integrated development environment** 17](#_Toc44024888)

[**4.2.** **Component based software development** 17](#_Toc44024889)

[**4.3.** **Test driven software development** 17](#_Toc44024890)

[**4.4.** **Object oriented programming** 17](#_Toc44024891)

[**4.4.1.** Programming language classes 17](#_Toc44024892)

[**4.4.2.** Client-side programming 18](#_Toc44024893)

[**4.4.3.** Server-side programming 18](#_Toc44024894)

[**4.4.4.** Middleware programming 18](#_Toc44024895)

[**4.5.** **Software deployment** 19](#_Toc44024896)

[**5.** **Software Testing** 19](#_Toc44024897)

[**5.1.** **Test plans and objectives** 19](#_Toc44024898)

[**5.2.** **Features and functions to be tested** 19](#_Toc44024899)

# **Software Project Description**

### **Project overview**

Desenvolvimento de uma solução Java Web que permite ao cliente, uma empresa produtora de calçado de segurança, disponibilizar aos seus clientes uma plataforma de gestão de encomendas dos seus produtos.

### **Purpose of the project**

### Main project goals

Permitir aos utilizadores efetuar o lançamento das suas encomendas, consultar o estado das mesmas bem como o seu histórico, através de “user friendly and intuitive interface”, assente numa arquitetura Model-View-Controller.

### Project development measurement

O desenvolvimento do projeto terá uma duração aproximada de 6 meses, e será dividido em duas fases. Na primeira fase, sem custos e em contexto de UC, será feita a modelação, desenvolvimento, testes e implementação da plataforma. Na segunda fase, como hipótese de continuação do projeto, será feita a integração desta com o ERP existente na empresa para consultar a informação dos clientes, produtos e as entregas.

### **Project background and scope**

### Current situation

Atualmente os clientes da empresa não dispõem de uma plataforma para comprar em quantidade e gerir as suas encomendas. Os métodos atuais levam a atrasos no processamento das encomendas e no feedback sobre as mesmas é reduzido e dependente dos comerciais.

### Business events and workflow

O cliente consulta um catálogo de produtos, envia por email as referências dos produtos e os tamanhos e quantidades de cada um para o Comercial. O Diretor de Operações atribui uma data de produção e expedição no ERP, cabendo ao comercial gerir os contactos e o envio de documentação para o cliente deste momento até à entrega da encomenda.

### Competing products and benchmarking

É uma solução inovadora, pois parece não existir no mercado um produto semelhante, e construída à medida do cliente. As funcionalidades da plataforma são semelhantes a uma loja on-line, no entanto estão ajustadas à medida do cliente e ao seu modelo de negócio armazenista.

### **Mockups and prototypes**

O desenvolvimento da plataforma passa sempre pela fase de prototipagem de design. Assim, foram definidos vários cenários de utilização em termos funcionais e de interface.

### Product scenarios



Figura 1 - Listar os Itens da Encomenda para o Utilizador



Figura 2 - Listar o histórico de Encomendas do Utilizador

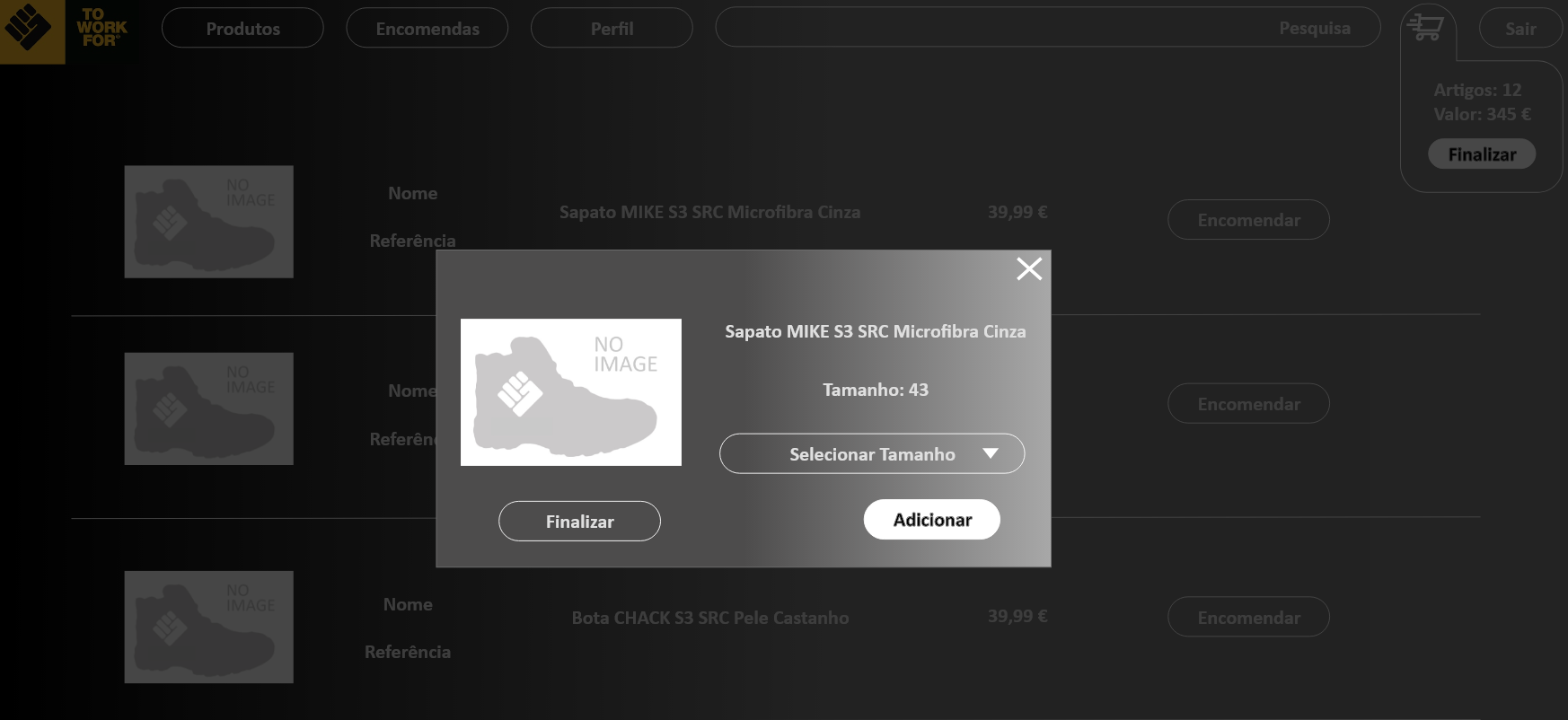


Figura 3 - Utilizador seleciona Produto, Tamanho e adiciona à Encomenda

### User stories and scenarios

Convidado

* Um visitante da plataforma pretende conhecer os produtos, registar-se como utilizador e aceder à plataforma para os poder comprar.

Utilizador

* + - * + Um Utilizador pretende ver e pesquisar dentro de uma lista de produtos disponíveis, escolhendo as quantidades e tamanhos de um determinado produto e adicioná-lo ao seu carrinho de compras.
        + O Utilizador pretende consultar e gerir o seu carrinho, validando e submetendo a encomenda, cuja informação do estado, envio e faturação fica disponível para consulta em histórico

Comercial

* + - * + O Comercial pretende listar e pesquisar todos os clientes da empresa, e criar encomendas em nome destes quando solicitado.

Administrador

* + - * + O Administrador pretende criar, listar, editar e eliminar informações sobre os produtos, clientes e utilizadores, e gerir os acessos à plataforma.

### **Stakeholders**

### Development team

Sergii Shtachenko e Pedro Soares

### Client

AMF, Lda

### Customer type

Administrador - Colaborador da AMF responsável pela atribuição dos acessos à plataforma para utilizadores, e gestão dos produtos disponíveis. Tem acesso total às funcionalidades da plataforma e ao histórico dos todos clientes.

Comercial - Colaborador da AMF que gere uma carteira de clientes. Vai ter possibilidade de lançar as encomendas em nome dos seus clientes e ter acesso ao seu histórico.

Utilizador - Colaborador da empresa cliente. Vai ter a possibilidade de lançar encomendas em nome da empresa a que pertence. Uma empresa-cliente poderá ter vários utilizadores a lançar encomendas em seu nome, mas cada utilizador pode ser associado apenas a uma empresa-cliente.

### **General constraints**

### Solution constraints

A solução não deve depender da OS do cliente nem requer instalação de qualquer componente.

### Schedule constraints

Conclusão da primeira fase até ao dia 20 de Junho de 2020. Conclusão da segunda fase conforme disponibilidade do cliente.

### Budget constraints

Sem restrições de orçamento.

### **Conventions and definitions**

### Key terms

Comercial - colaborador da AMF que faz parte do departamento comercial cuja principal função é gerir uma carteira dos Clientes.

Cliente - uma empresa nacional ou estrangeira que tem interesse na compra e distribuição dos produtos da AMF.

Utilizador - um colaborador do Cliente.

Produto - calçado de segurança produzido e/ou comercializado pela AMF.

Item - unidade de compra que é um Produto de um determinado Tamanho.

Tamanho - característica do item em escala europeia entre 35 e 48.

Referência - uma designação interna da AMF para identificar o Produto.

Encomenda - um documento de compra composto por um ou mais produtos com quantidades discriminados por tamanho.

### Unified Modelling Language (UML)

Foi elaborado um diagrama UML para estruturação e planeamento das classes da aplicação, um diagrama de casos de utilização para cada tipo de utilizador, e Modelo Relacional da Base de Dados.

# **Software Requirements**

* 1. **Actors and use cases**

### Actors

Convidado (nível 0) - utilizador não identificado (sem login)

Utilizador (nível 1) - colaborador de uma empresa-cliente

Comercial (nível 2) - colaborador da AMF

Administrador (nível 5) - colaborador da AMF

### Use case list

Convidado

* Listar produtos
* Login
* Solicitar acesso

Utilizador

* Listar produtos
  + Adicionar produto no carrinho de compras
* Ver carrinho de compras
  + Editar produto no carrinho de compras
  + Eliminar produto do carrinho de compras
  + Finalizar encomenda
* Ver histórico das compras/entregas

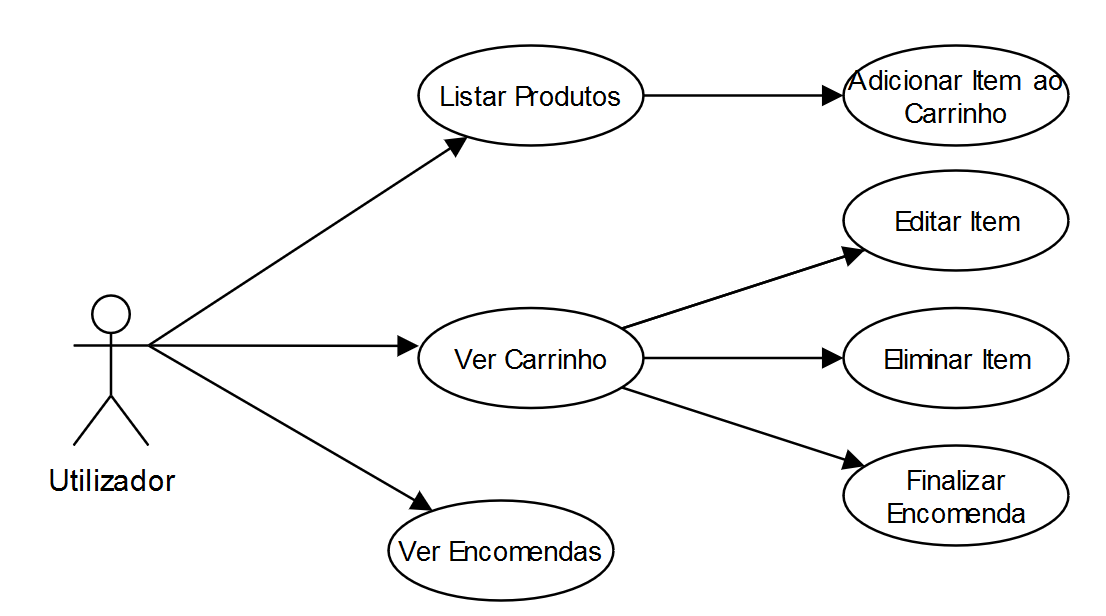
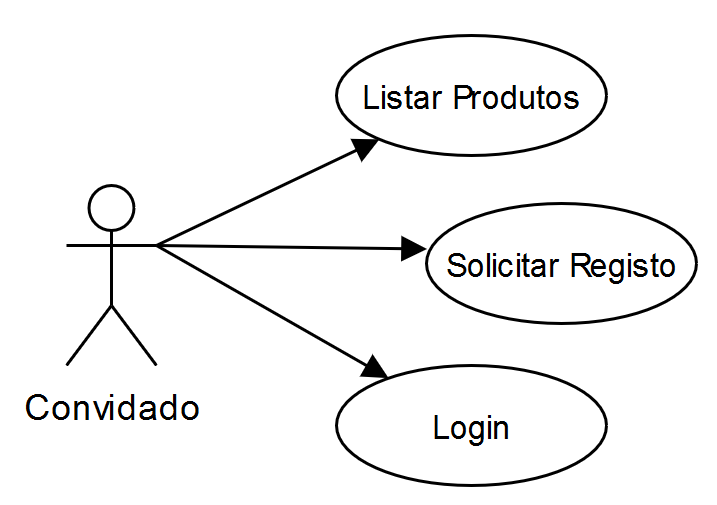
Comercial

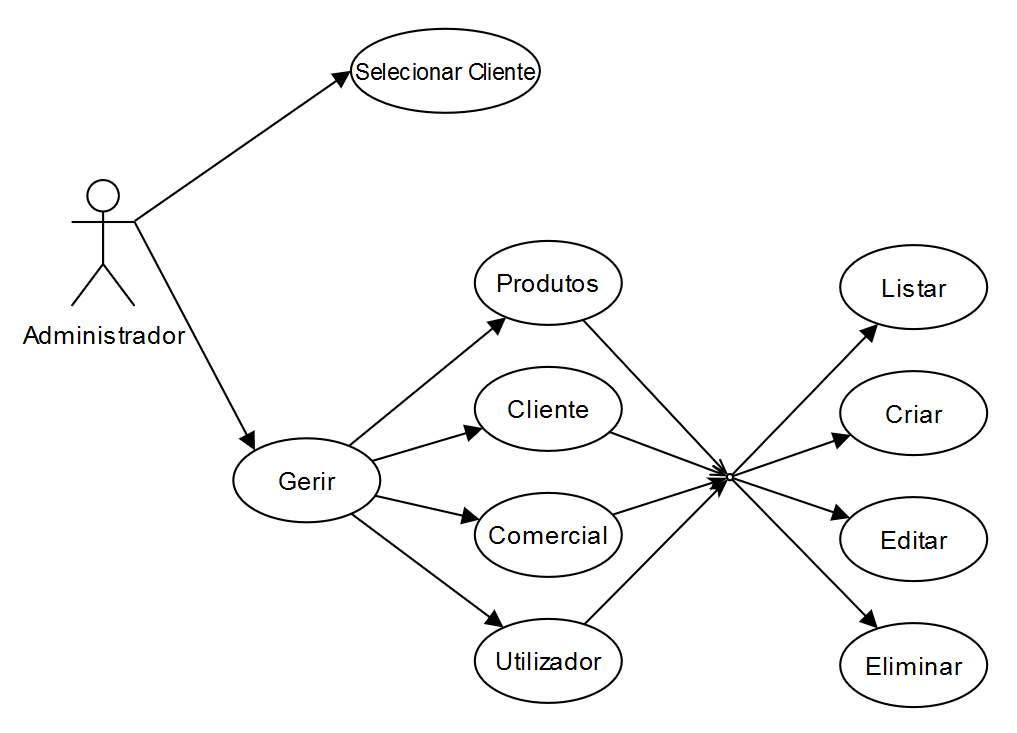
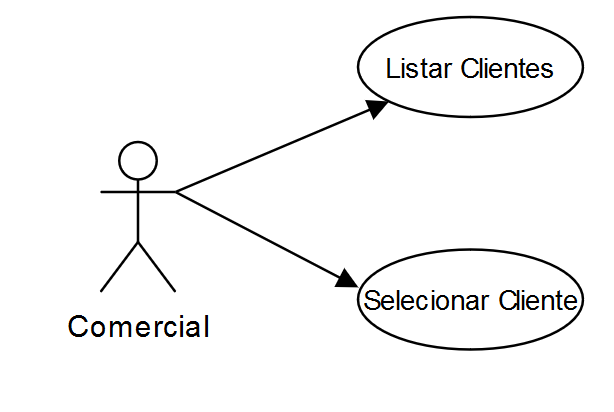
* Listar clientes
* Escolher cliente
  + Listar produtos
    - Adicionar produto no carrinho de compras
  + Ver carrinho de compras
    - Editar produto no carrinho de compras
    - Eliminar produto do carrinho de compras
    - Finalizar encomenda
  + Ver histórico das compras/entregas

Administrador

* Listar utilizadores
  + Criar utilizador
  + Editar utilizador
  + Eliminar utilizador
* Listar clientes
* Escolher cliente
  + Listar produtos
    - Adicionar produto no carrinho de compras
  + Ver carrinho de compras
    - Editar produto no carrinho de compras
    - Eliminar produto do carrinho de compras
    - Finalizar encomenda
  + Ver histórico das compras/entregas

### Use case diagrams





### **Domain and functional requirements**

### Detailed User-Stories

Convidado

Um convidado ouviu falar da empresa e pretende conhecer os seus produtos.

O convidado gostou da apresentação dos produtos e pretende registar-se como utilizador para os poder comprar.

O convidado já tem o seu registo ativo e pretende entrar na plataforma para comprar produtos.

Utilizador

O Utilizador pretende gerir a sua informação pessoal e mudar a password com segurança.

O Utilizador pretende ver uma lista com todos os produtos disponíveis, para selecionar os que pretende comprar.

O Utilizador sabe qual produto quer comprar e pretende pesquisar esse produto dentro da lista.

O Utilizador escolhe as quantidades e tamanhos de um determinado produto e adiciona-os ao seu carrinho de compras.

O Utilizador lista todos os produtos adicionados ao carrinho de compras, e pretende confirmar ou atualizar as quantidades.

O Utilizador submete o pedido na plataforma.

O Utilizador pretende saber o estado da sua encomenda.

O Utilizador lista o histórico de todos os pedidos realizados através da plataforma.

Comercial

O Comercial pretende listar todos os clientes da empresa.

O Comercial pretende pesquisar um cliente específico na lista de clientes.

O Comercial pretende selecionar um cliente para poder interagir com a plataforma em seu nome.

O Comercial tem acesso à plataforma com todas as funcionalidades de um cliente.

Administrador

O Administrador tem acesso total à plataforma com todas as funcionalidades de um cliente e de um comercial.

O Administrador lista, cria, edita e elimina utilizadores, clientes e produtos na plataforma.

### **Non-functional requirements**

### Database requirements

A base de dados deverá ser ser normalizada na 3ª forma normal (3FN) e implementada em MySQL, refletir a estrutura funcional e orientada a objetos da plataforma e ter a capacidade de guardar os milhares de registos que vão ser gerados.

Por convenção, cada registo terá associado uma data, que permitirá distingui-lo dos demais com elevada precisão, e um estado que servirá para fazer a eliminação lógica desse registo, por exemplo, no caso de um produto ser descontinuado.

### Performance requirements

A capacidade de resposta da plataforma deve depender apenas da velocidade de conexão do utilizador.

A sua estrutura gráfica leve e a diversificação e reaproveitamento de código não vão gerar lentidão no carregamento da página. O sistema deverá ser capaz de suportar 50 utilizadores em simultâneo.

### Capacity and storage requirements

Os registos associados à plataforma deverão ocupar um máximo de 1GB de memória física.

### Security requirements

Apenas utilizadores registados devem poder aceder à plataforma com uma palavra passe e gerir os próprios dados.

Convidados apenas devem poder consultar os produtos disponíveis.

O administrador terá acesso aos dados de todos os utilizadores, com exceção das suas palavras passe, que deverão estar encriptadas.

### Scalability and extensibility requirements

A plataforma deve ser codificada numa lógica orientada a objetos, permitindo crescer e expandir-se até aos limites físicos de memória.

### Maintenance requirements

A construção da plataforma deverá permitir uma manutenção rápida e intuitiva, permitindo ao administrador adicionar ou remover funcionalidades utilizando as funções já implementadas.

### Robustness or fault-tolerance requirements

A codificação deverá impedir tentativas de entrada por routing, e todos os erros deverão ser tratados pelo sistema.

### Dependability requirements

A plataforma deverá ser disponibilizar as suas funcionalidades de forma fiável e persistente.

### Reliability requirements

A plataforma deverá fornecer serviços corretamente e conforme o esperado pelos utilizadores, pelo menos durante 95% do tempo.

O período em que a plataforma não estiver acessível não poderá exceder 1 hora.

### Availability requirements

A plataforma deverá estar sempre em funcionamento e ser capaz de fornecer serviços úteis aos utilizadores.

### Privacy requirements

O tratamento dos dados dos utilizadores deverá obedecer às normas do Regulamento Geral de Proteção de Dados, de 25 de Maio de 2018.

### Usability requirements

A plataforma deverá ser de utilização intuitiva, com a informação mais relevante facilmente visível para o utilizador e com os principais caminhos devidamente identificados.

Um utilizador deverá ser capaz de utilizar todas as capacidades da plataforma ao fim de um dia de experiência.

### Accessibility requirements

A plataforma deverá ser acessível a qualquer trabalhador das empresas clientes.

A apresentação das páginas irá privilegiar a visibilidade em detrimento da quantidade de informação, bem como a integração com as funcionalidade nativas do sistema operativo.

### Compliance Requirements

O funcionamento da plataforma deverá estar de acordo com as normas em vigor em qualquer circunstância.

### Standards Requirements

A plataforma deverá respeitar os padrões do Software Engineering Standards Committee da IEEE Computer Society. Também na arquitetura da plataforma iremos utilizar os padrões MVC (Model-View-Controller) e DAO (Data-Access-Object).

### User documentation requirements

A implementação da plataforma será baseada nos requisitos e funcionalidades descritas no presente documento.

### Internationalization requirements

Sendo o cliente uma empresa internacional, a informação da plataforma deverá ser apresentada em inglês.

### Training requirements

A plataforma será voltada para o cliente, totalmente intuitiva e de fácil utilização.

Deverá ser ministrada uma formação inicial para o Administrador do Sistema e para os Comerciais da empresa.

### Cultural and political requirements

A plataforma é agnóstica a questões políticas e culturais.

### Legal requirements

A plataforma deve obedecer a todos os normativos legais em vigor relativos ao desenvolvimento de software.

### Release requirements

A primeira versão da plataforma será entregue ao cliente, totalmente funcional, testada e aprovada por este, dentro do prazo estipulado previamente.

O alojamento ficará a cargo do cliente.

# Software Design

### **Existing software architecture**

A plataforma a desenvolver será totalmente nova e feita de raiz, não havendo necessidade de conhecer qualquer arquitetura previamente desenvolvida.

### **Design goals**

Apresentar uma plataforma com um design limpo e profissional, que destaque a usabilidade.

As funcionalidades apresentadas ao utilizador devem ser de fácil compreensão e identificação.

As informações apresentadas devem privilegiar a qualidade em detrimento da quantidade. Serão apresentadas as informações mais relevantes ficando os detalhes escondidos mas disponíveis à distância de um clique do utilizador.

A plataforma deve ter um design totalmente responsivo e funcional independente do tamanho do equipamento utilizado.

### **Proposed software architecture**

A arquitetura de um sistema tem elementos utilitários, elementos de interação, elementos que fazem parte do domínio do problema, elementos de conexão, elementos de persistência, etc. Dessa forma, na arquitetura definimos os elementos que precisarão ser utilizados no software e como eles se interligam. Esta plataforma será desenvolvida na arquitetura Model-View-Controller (MVC) que possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o Model, o Controller e a View, executa o que lhe é definido e nada mais do que isso. Tem a grande vantagem de isolar as regras de negócios da lógica de apresentação e isto possibilita a existência de várias interfaces que podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, proporcionando assim muito mais flexibilidade e oportunidades de reutilização das classes. A comunicação entre interfaces e regras de negócios é definida através de um controlador, que torna possível a separação entre as camadas. Quando um evento é executado na interface gráfica, como, por exemplo, um clique em um botão, a interface irá comunicar com o controlador que por sua vez se comunica com as regras de negócios, acontecendo o processo inverso que traz ao utilizador o feedback visível da sua ação.

### **Algorithms and Data Structures**

### Data structures

A informação dos Models será organizada em classes que representam os objetos (ObjectBean) e classes DAO que fazem a ligação dos objetos com a Base de Dados (ObjectDao).

Nas classes do tipo ObjectBean serão definidas as “Regras de Negócio”, ou seja, as Constantes, os Atributos, os Construtores e os métodos Getters e Setters.

Nas classes do tipo ObjectDao serão definidas as funções que permitem fazer o CRUD (Create, Read, Update, Delete) dos objetos através da ligação com a base de dados.

Ainda no domínio dos Models, será criada uma classe acessória com a função de criar a ligação à base de dados e que é instanciada cada vez que os Models pretendem comunicar com a base de dados.

### Algorithms for problem solving

A estrutura dos Controllers será definida por um algoritmo com base numa estrutura Switch-Case que permitirá escolher a função a chamar por cada uma das ordens recebidas da interface.

A estrutura dos restantes algoritmos irá respeitar as boas práticas de desenvolvimento, ou seja, irão começar sempre pela declaração das variáveis, seguindo-se os diversos comandos. Os algoritmos desenvolvidos não deverão ter complexidade, optando-se por compartimentar os algoritmos em funções mais pequenas.

Iremos utilizar estruturas condicionais várias (while, for, forEach e if/else) para validações e estruturas de teste e tratamento de erros (try/catch) para testar os blocos de código e tratar os respetivos erros.

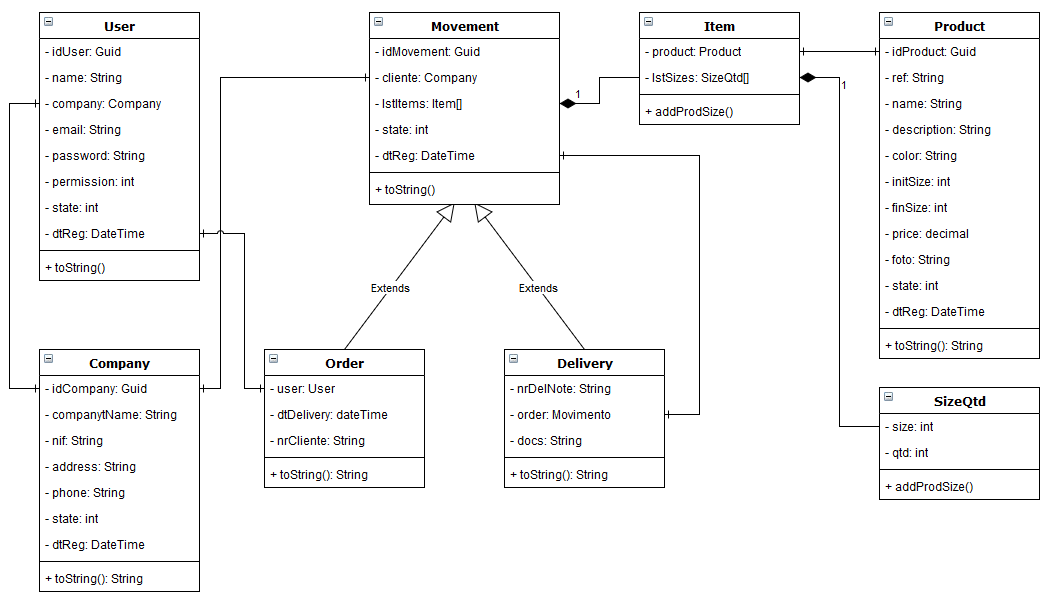
Um dos algoritmos de destaque é aquele que permite codificar as palavras passe dos utilizadores segundo o *hash* MD5 - getMD5().

### **Object oriented analysis and design**

### Modular design and encapsulation

As classes que compõem esta solução apresentam um design modular assente na compartimentação do código, que permite encapsular a informação mantendo-a escondida do exterior e visível apenas dentro das próprias classes. Assim, os atributos de cada classe são privados e apenas os métodos são públicos. Conseguimos assim garantir que não é possível manipular diretamente os atributos, para o fazer devem usar-se os métodos da classe.

### Class diagrams



### Packages and class interfaces

A informação desta solução encontra-se dividida em 3 pastas: Web pages, que define as Views, as diversas interfaces com o utilizador; Models, que define as regras de negócio e as suas interações com a base de dados, Controllers, que integra o middleware de ligação entre as regras de negócio e o interface;

Cada uma das classes do tipo ObjectBean atrás apresentadas terá associada uma classe do tipo *ObjectDao* que fará a interação com a base de dados utilizando uma classe que faz essa ligação. Entre as diversas classes existirão as mais frequentes relações de composição, agregação e associação, destacando-se também a existência de relações de herança entre a Superclasse *Movement* e as classes *Order* e *Delivery* onde aplicaremos o conceito de polimorfismo, permitindo que as classes derivadas sejam capazes de invocar os métodos da classe base, comportando-se de maneira diferente, embora apresentem a mesma assinatura.

### **User Interface Design**

Para conseguirmos cumprir o requisito de apresentar uma solução *user friendly and intuitive interface* foi fundamental pensar numa plataforma *page load less*, ou seja, minimizar o carregamento das páginas utilizando com frequência *client side programming* com JSON e Ajax. Tal opção de engenharia permitiu-nos criar páginas dinâmicas e interativas, que possibilitaram, entre outras funcionalidades, o carregamento do carrinho de compras em tempo real e o respetivo cálculo dos totais atualizados.

### Interface design models

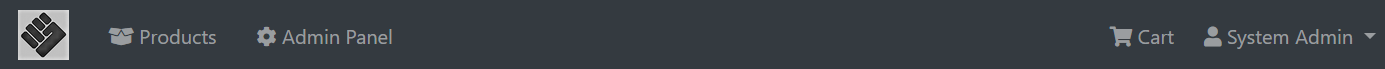


Figura 4 - Modelo de Barra de Navegação do Administrador

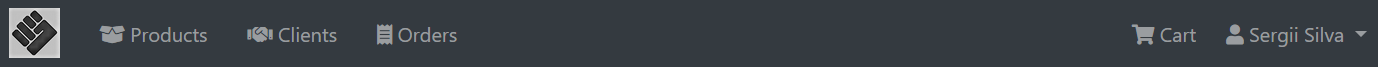


Figura 5 - Modelo de Barra de Navegação do Comercial

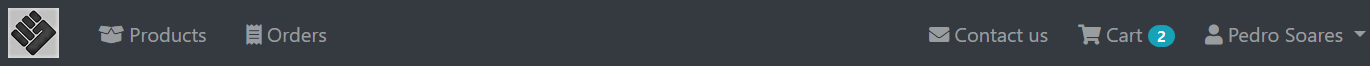


Figura 6 - Modelo de Barra de Navegação do Utilizador Registado

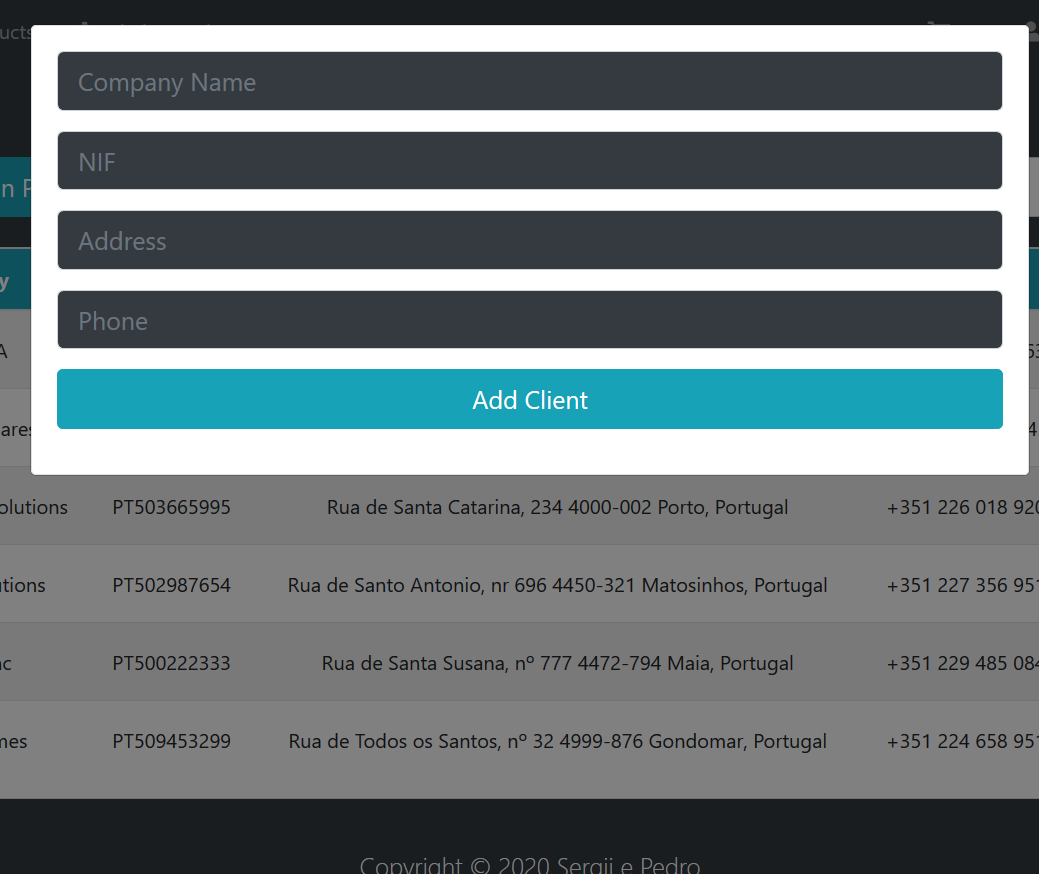
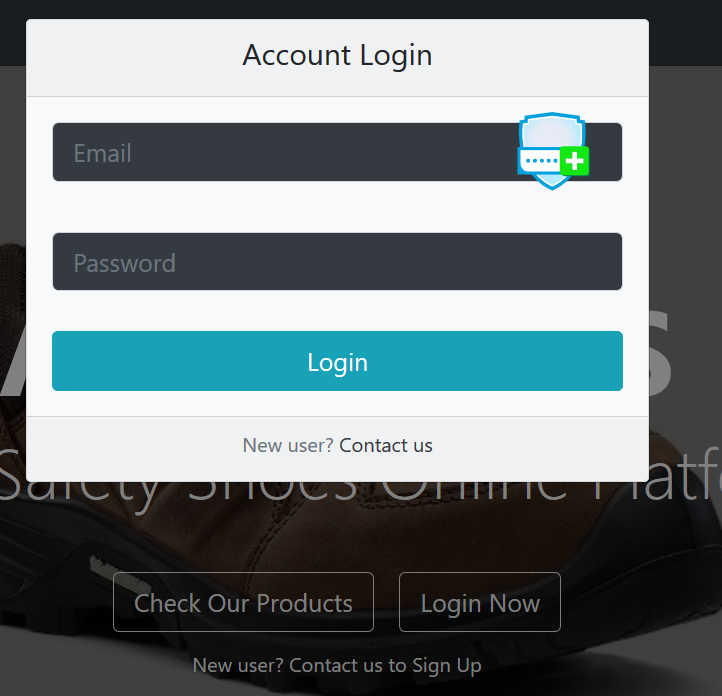


Figura 8 - Modelos de Janela Modal

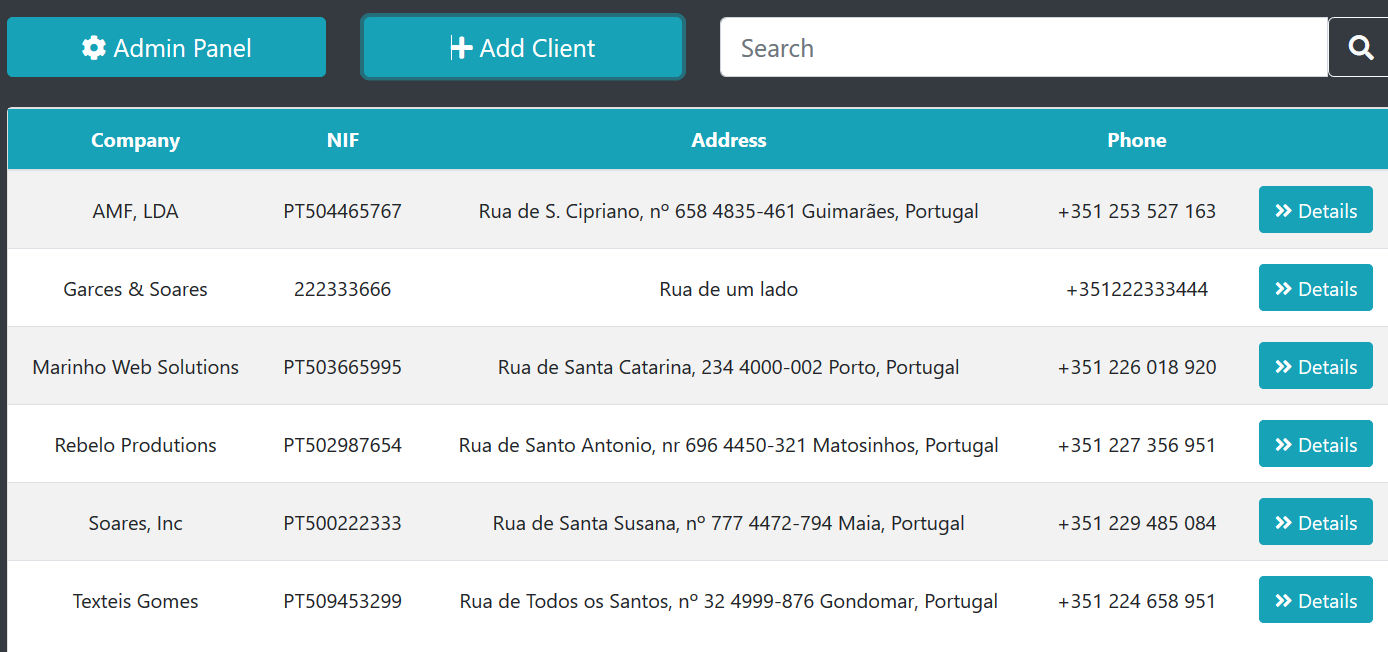


Figura 9 - Modelos de tabela

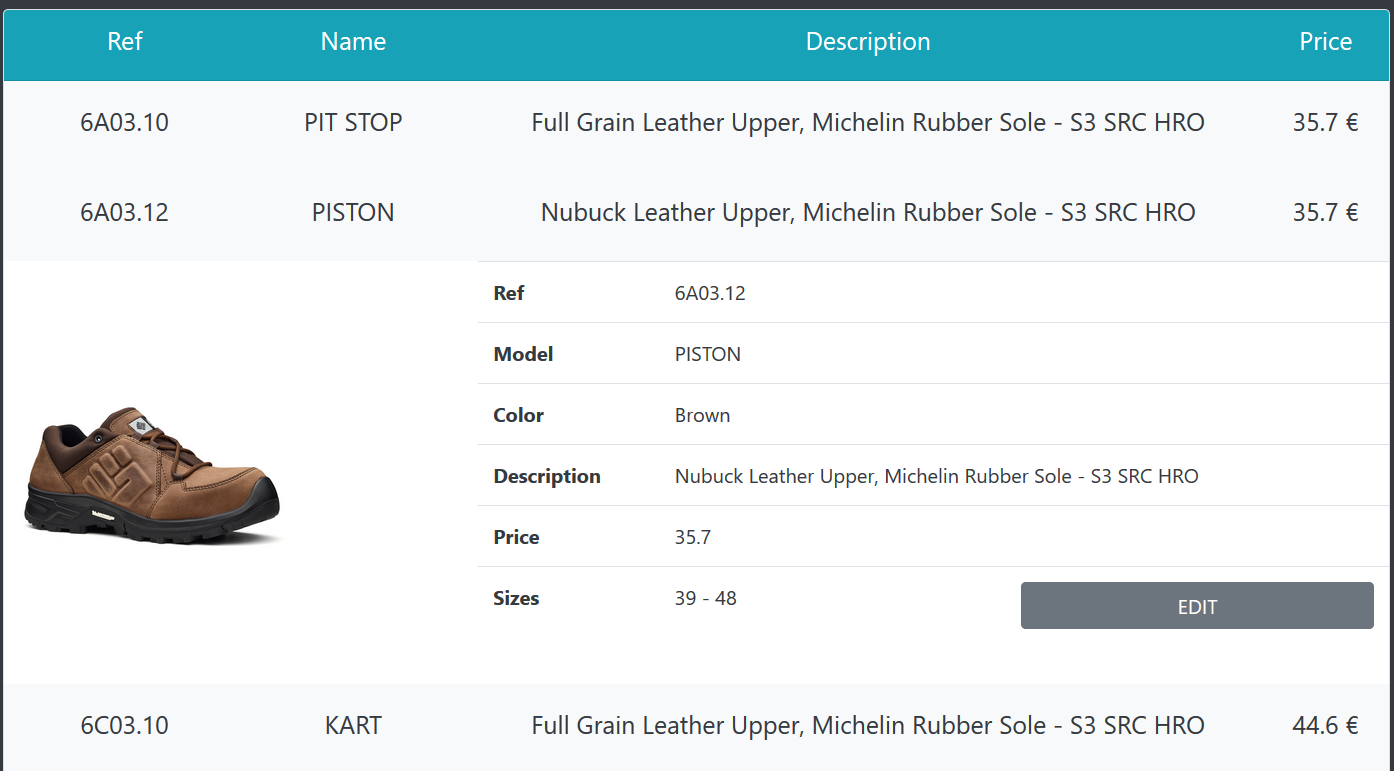


Figura 10 - Modelo de tabela dinâmica

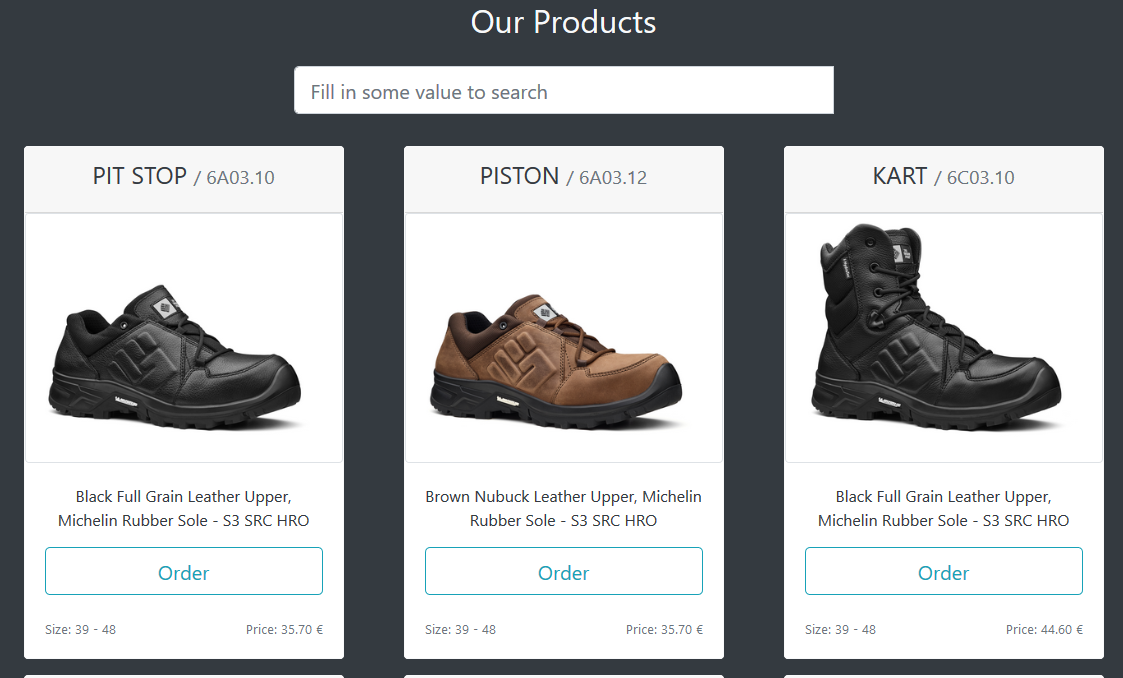


Figura 11 - Modelo de lista de produtos apresentados ao utilizador

### Interface design process

O processo de design da interface baseou-se nos pré-requisitos definidos e foi todo realizado a partir de uma página HTML em branco e com recurso a Bootstrap, dado o seu potencial e responsividade nativa. Pretendeu-se um esquema de cores pouco agressivo, com fundo escuro, janelas e espaços brancos para destaque do produto e uma grande preocupação com a facilidade de utilização patente no destaque colorido dos botões.

# Software Construction

### **Integrated development environment**

O desenvolvimento da aplicação foi realizado em Java através do IDE Apache NetBeans, com ligação através do servidor Apache Tomcat 9.

A base de dados foi criada e gerida através do software MySql Workbench. Algumas consultas foram realizadas diretamente na aplicação de ligação à base de dados própria do Apache NetBeans.

### **Component based software development**

O desenvolvimento do nosso software privilegiou a reutilização de componentes. Como exemplo, o mesmo algoritmo de pesquisa é utilizado em todas as pesquisas realizadas, independentemente do objeto procurado uma vez que é apenas dependente dos parâmetros passados. Da mesma forma, funções semelhantes para classes diferentes são construídas com a mesma estrutura mudando apenas os parâmetros e as referências.

### **Test driven software development**

No desenvolvimento da nossa solução não foram usados testes automatizados. Contudo, o desenvolvimento assentou no princípio de testar a funcionalidade de cada pedaço de código antes de avançar para o seguinte. Assim, foi possível fazer a validação ou ajuste que conduziu ao sucesso de cada função.

### **Object oriented programming**

O paradigma da programação orientada a objetos deve estar presente em todos os projetos de desenvolvimento atuais. No nosso caso, não foi exceção. Cada entidade da nossa plataforma é um objeto e todas as referências entre entidades acontecem através de objetos inteiros, ou seja, um User está associado a uma Company e, ao ser criado, recebe a totalidade desse objeto e não apenas uma referência por “id”. Desta forma conseguimos aceder a atributos de um objeto dentro de outro objeto.

### Programming language classes

O desenvolvimento das classes foi realizado na linguagem de programação Java.

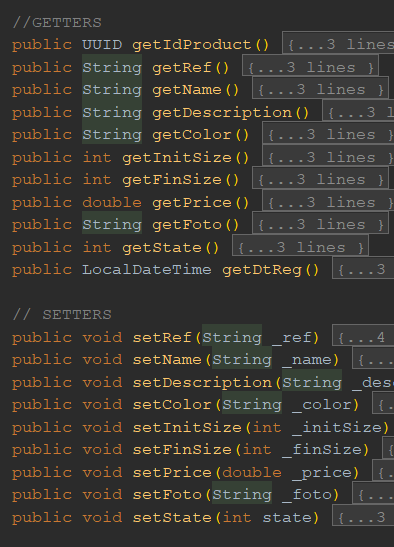
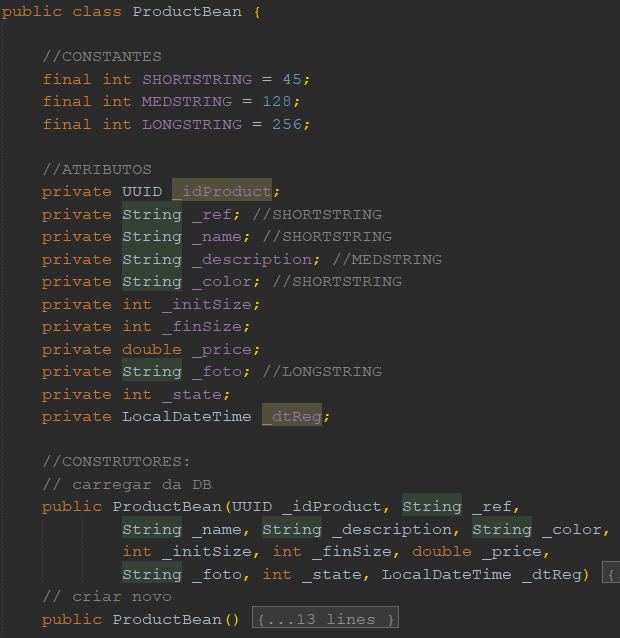


Figura 12– Exemplo de classe Java do tipo *ObjectBean*

### Client-side programming

Do lado do cliente, a estruturação da página apresentada utiliza HTML, a sua estilização usa a tecnologia CSS e a dinâmica da página é dada pela utilização da tecnologia Javascript. Para atingir os requisitos de usabilidade e de responsividade da interface, estas tecnologias foram integradas usando classes de Bootstrap.

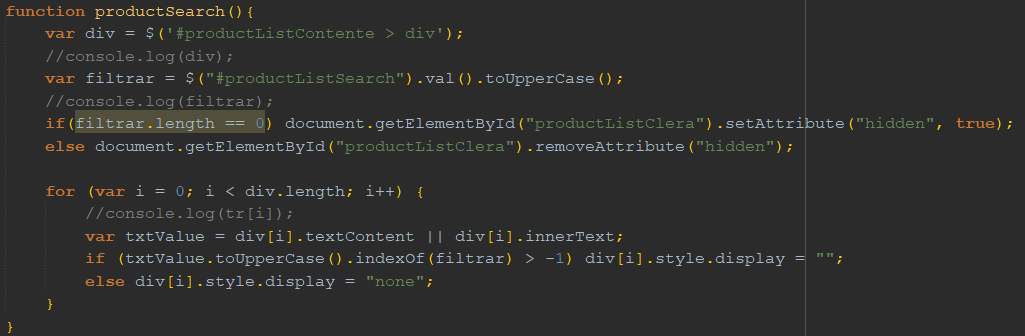


Figura 13– Exemplo de função JavaScript para procurar e mostrar um determinado elemento

### Server-side programming

Do lado do servidor, foi utilizada tecnologia Java Servlets e Java Server Pages (jsp), integrar código Java dentro de uma página HTML escondendo-o do utilizador, e também a biblioteca JSTL (Java Standard Tag Library), uma coleção de tags JSP que encapsula as funcionalidades principais do JSP, permitindo tarefas estruturais comuns, como iteração e expressões condicionais, diretamente no HTML.

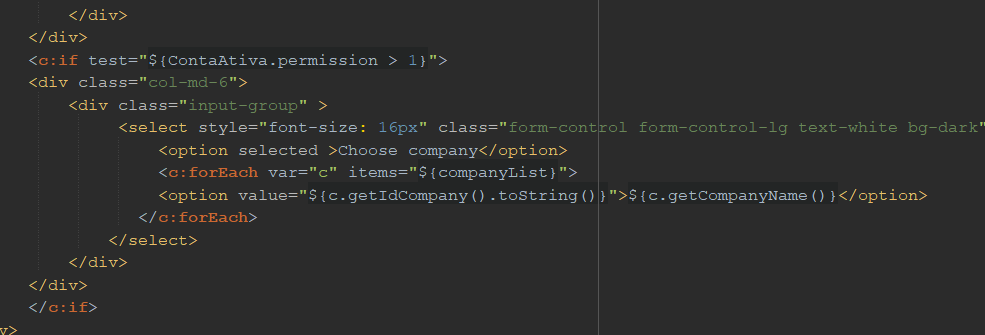


Figura 14– Exemplo de utilização de JSTL em JSP: estrutura de iteração *forEach* dentro de estrutura condicional *if/else*

### Middleware programming

Para consolidar a ligação entre o lado do servidor e o lado do cliente foram utilizadas as tecnologias JSON (JavaScript Object Notation) e Ajax (Asynchronous JavaScript And XML). Estas duas tecnologias integradas permitem um carregamento assíncrono de dados armazenados sendo possível enviar e receber dados do servidor sem precisar recarregar a página inteira.

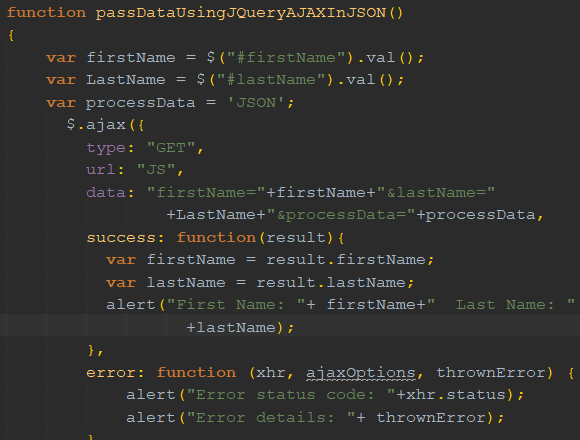
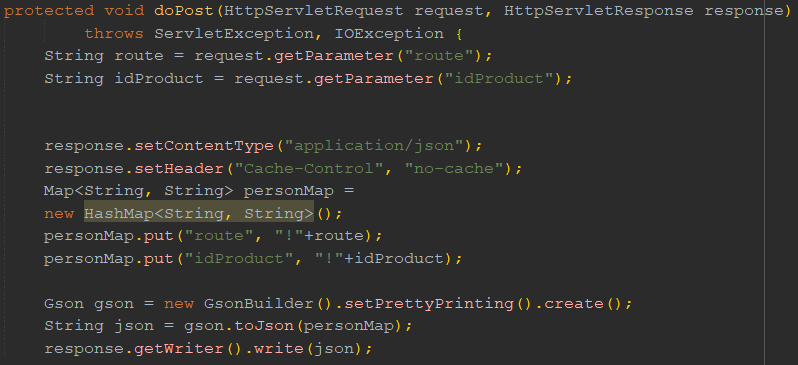


Figura 15 – Exemplo de implementação de JSON e Ajax

### **Software deployment**

O deploy da nossa plataforma será realizado apenas nas nossas máquinas para apresentação ao docente, ressalvando que, dado o estado de avançado desenvolvimento, esta se encontra praticamente pronta para deploy nos servidores da empresa cliente.

## Software Testing

### **Test plans and objectives**

Antes de ser entregue ao cliente, a plataforma será alvo de uma bateria de testes funcionais e não funcionais, garantindo que toda a estrutura e funcionalidades da plataforma estão devidamente implementadas e capazes de dar resposta a todos os cenários de utilização previstos possíveis.

### **Features and functions to be tested**

Todos os casos de utilização previstos serão testados exaustivamente. A estes serão acrescentados testes unitários de funções independentes, testes de integração e performance do sistema, testes de sobrecarga de utilização, testes de segurança e de usabilidade.

Um teste será considerado um sucesso cumprir 97% dos critérios estabelecidos. Pretende-se que a plataforma apresente um funcionamento de excelência.