

App Install & Go

Roberto Filipe Manso Barreto
(nrº 21123, regime diurno)

Orientação de
Luís Gonzaga Martins Ferreira

LICENCIATURA EM ENGENHARIA EM SISTEMAS INFORMÁTICOS
ESCOLA SUPERIOR DE TECNOLOGIA
INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE

Identificação do aluno

Roberto Filipe Manso Barreto
Aluno número 21123, regime diurno
Licenciatura em Engenharia em Sistemas Informáticos

Orientação

Luís Gonzaga Martins Ferreira

Informação sobre o Estágio

Motorline Eletrocelos S.A
Travessa do Sobreiro, 29 Rio Côvo (Sta. Eugénia) 4755-474 Barcelos
Eng. Helder Remelhe

Resumo

Resumo do trabalho realizado. Deve ser sucinto, e cobrir todo o relatório: uma introdução ao problema que se pretendeu resolver, um pequeno resumo da abordagem realizada, e algumas conclusões do trabalho atingido.

Poderão ser criados vários parágrafos, até para que cada um corresponda às três fases de introdução, desenvolvimento e conclusão.

Não é relevante colocar no resumo o local de estágio ou a referência ao curso. Essa informação já consta da capa.

Abstract

This is the translation of the previous text. It should say the exact same thing. Please do not use directly Google Translator.

Agradecimentos

[A secção de agradecimentos é a parte pessoal do documento, e o único sítio onde o aluno pode escrever de forma menos formal, usando o tipo de linguagem que lhe parecer adequado para as pessoas a quem agradece.]

Conteúdo

1. Implementação

1.1 Web scraper

Após uma reunião com o cliente foi percebido que o catálogo de produtos Motorline não se encontra em um servidor, esta informação encontra-se apenas diretamente no website da empresa, sendo assim viu-se a necessidade de criar um web scraper.

Web scraping é uma terminologia dada para o processo de obter uma página web, ler a página e obter dados desta, geralmente utilizando bots. O grande problema com web scraping é que pode ser facilmente detetado. Tendo em conta este problema surgiram duas grandes formas principais de realizar web scraping, a mais comum sendo realizar um pedido para obter uma página web e ler então esta, sendo assim um processo rápido e simples. A segunda forma de realizar web scraping é através da simulação da ação humana conseguindo abrir o navegador pesquisar pela página desejada, descarregar a página e daí ler esta, tornando-se então em um processo lento e complexo. A grande diferença entre estas duas formas é a velocidade, visto que a segunda forma tem de esperar que o navegador inicie, de seguida terá de esperar que a página carregue e apenas após este processo poderá ser lida a página web.

Na reunião mencionada anteriormente foi decidido que o web scraper iria apenas correr 1 vez por mês de forma a evitar a sobrelocação do servidor, não existindo problema visto que o catálogo não é atualizado regularmente. Para agilizar a realização do web scraper foi disponibilizado pela empresa a estrutura do website a seguir para obter as informações da página web.

1.1.1 Implementação web scraper

De forma a implementar e testar o web scraper sem sobreloitar o servidor, foi então descarregado todo o website localmente, conseguindo assim simular o mesmo.

Para implementar o web scraper foi optado pela abordagem mais simples, realizar um pedido para obter a pagina web, ler a página para obter os dados e guardar os dados.

Para isto foi optado pela linguagem python devido à facilidade desta lidar com grandes quantidades de dados. De forma a facilitar a localização dos dados na página foi utilizada a biblioteca bs4, também conhecida como beautiful soup, esta biblioteca permite alimentar com uma página web e de seguida realizar pesquisas sobre esta página baseado em tags e atributos dos elementos.

Tendo esta base em conta foi então primeiramente estudado que dados seriam necessários, sendo estes então:

1. Categorias e subcategorias de produtos;
2. Produtos de cada categoria e subcategoria;
3. Documentação dos produtos;
4. Imagens e videos dos produtos;

Para guardar estes dados foi utilizado um dicionário que contém primeiramente como chaves as categorias de produtos, para cada categoria contém mais um dicionário com as subcategorias de produtos e para cada categoria existe uma lista de produtos, contendo o nome de produto, imagem de amostra e url do mesmo. Por fim a chave produtos contém a lista de todos os produtos, sendo cada produto representado também por um dicionário, que contém como chaves os atributos do mesmo. A utilização dicionários e listas para guardar estes dados deve-se a que o objetivo será guardar estes dados em json e a transformação é simplificada utilizando estas estruturas devido à sua proximidade com a estrutura json.

```
{
    "produtos": [
        {
            "nome": "nome do produto",
            "categoria": "categoria do produto",
            "subcategoria": "subcategoria do produto",
            "imagem-amostra": "imagem de amostra do produto",
            "imagens": [
                "links de imagens de produtos"
            ],
            "descricao": "descrição do produto",
            "documentacao": [
                {
                    "nome": "Nome da documentação",
                    "url": "link da documentação"
                }
            ],
            "placas-controlo": [
                {
                    "nome": "nome da placa de controlo",
                    "url": "documentação placa de controlo"
                }
            ],
            "informacao-geral": "imagem de informação geral",
            "desenho-tecnico": "imagem de desenho técnico",
            "videos": [
                {
                    "name": "nome do video",
                    "url": "link do video"
                }
            ]
        },
        "categorias": {
            "nome_categoria": {
                "nome_subcategoria": [
                    {
                        "nome": "nome produto",
                        "link": "link página do produto",
                        "imagem-amostra": "link imagem de amostra do produto"
                    }
                ]
            }
        }
    ]
}
```

Figura 1.1: Estrutura dos dados obtidos

Após uma análise da estrutura do website foi percebido que a página geral de produtos possui todas as categorias de produtos, assim como também as subcategorias de produtos com urls para as páginas que contém todos os produtos das subcategorias. Sendo assim foi primeiramente percebido que cada conjunto é uma secção, pelo que é obtido todas as secções de categorias e para cada uma destas secções é obtido o título da secção que equivale ao nome da categoria e também todos os correspondentes a clicáveis. Os clicáveis corresponde às subcategorias de cada categoria estes clicáveis contém também um url que redireciona para a página de produtos da subcategoria.

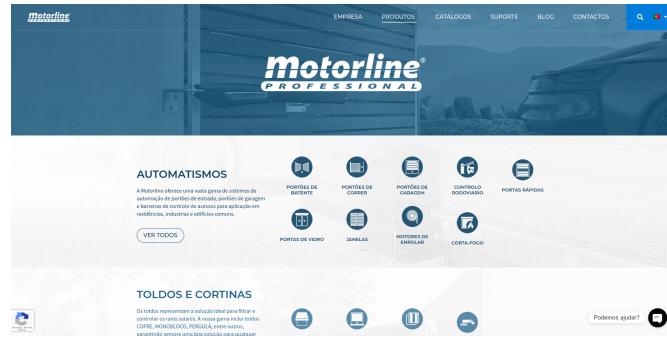


Figura 1.2: Página geral de produtos

Sendo assim já é possível identificar cada categoria e subcategoria, assim como também o url da página de produtos para cada subcategoria. Mas após alguma análise dos dados foi percebido que estas não contêm acentuação devido à sua formatação no website. Para resolver este problema foi pesquisado por ferramentas capazes de corrigir estes erros ortográficos. Pelo que foi descoberto que a biblioteca mais utilizada em python para resolver este problema é a biblioteca spellchecker, esta ferramenta é a mais utilizada devido à sua capacidade de corrigir erros ortográficos em diversas linguagens. Sendo assim sempre que uma categoria e subcategoria é obtida, antes de ser guardada, esta é corrigida.

Após isto cada url é aberto e são obtidos os urls de produtos e imagens de amostra dos produtos, para isto foi obtido todos os elementos clicáveis existentes na secção de produtos de cada página, sendo que cada um corresponde a um produto, para obter o nome do produto correspondente foi utilizado o nome contido no url da página de produto, sendo que todos os produtos seguem a mesma estrutura, sendo esta, /produtos/nome-produto. Sendo que em urls não é permitido utilizar acentuação e espaços, então todos os nomes foram corrigidos utilizando a mesma ferramenta mencionada anteriormente.

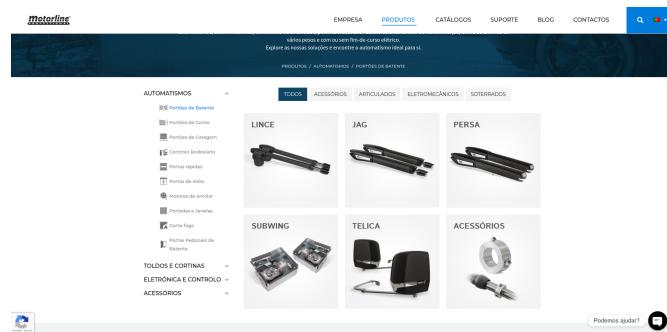


Figura 1.3: Página de produtos de uma subcategoria

Neste momento após correr o código foi percebido que existiam algumas páginas de produtos em que este não conseguia obter produtos, pelo que um erro era atirado, para perceber exatamente que páginas de produtos este erro acontecia, sempre que um erro era detetado este url seria adicionado a uma nova chave do dicionário mencionado anteriormente, esta chave tem o nome misses e contém todos os urls em que algum erro aconteceu. Foi então neste momento que foi percebido que nem todas as páginas de produtos são iguais e após uma reunião com o cliente este expôs que existem páginas de produtos e de detalhes de produtos que são muito diferentes das restantes.

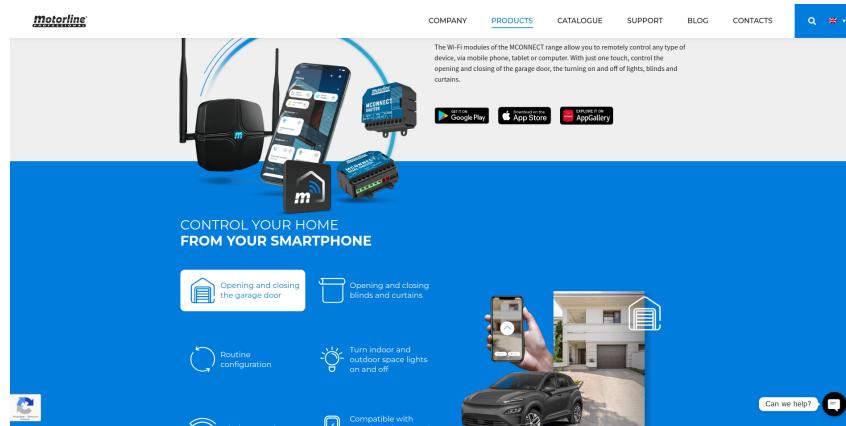


Figura 1.4: Página de produtos de uma subcategoria distinta

De forma ao restante do projeto não ser atrasado foi então decidido primeiramente obter todos os produtos que contêm páginas semelhantes, sendo assim para cada página de produto foi obtido o título que corresponde ao nome do produto, de seguida foi obtida a descrição do produto, o elemento que contém esta tem como id produto-descrição. As imagens dos produtos são disponibilizadas através de urls na secção da galeria do produto, sendo assim são obtidas todas as imagens desta galeria e de seguida todos os seus urls.

A documentação dos produtos pode ser disponibilizada através de urls para os manuais, ou com uma lista dropdown com todos os manuais disponíveis para download, sendo assim são obtidos todos os urls da secção de documentação, assim como os seus nomes e todas as opções de documentação do dropdown se este existir.

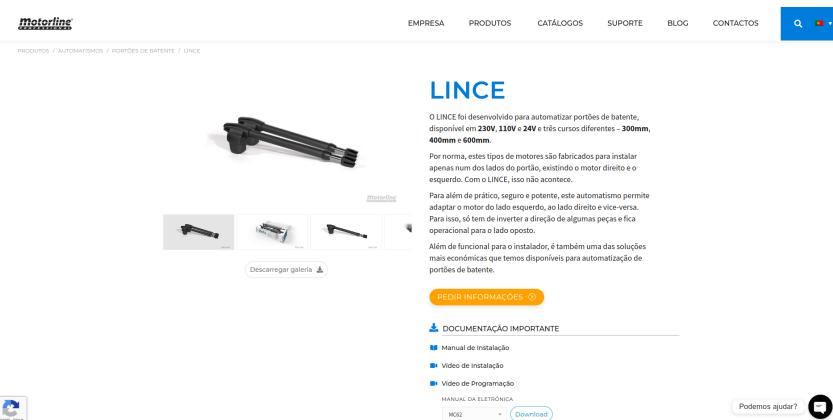


Figura 1.5: Página de detalhes de produto, secção inicial

As imagens de desenho técnico e informação geral estão disponibilizadas na secção correspondente ao nome de cada uma, sendo assim obtidas estas secções e caso estas existam são obtidas as imagens e os seus urls.

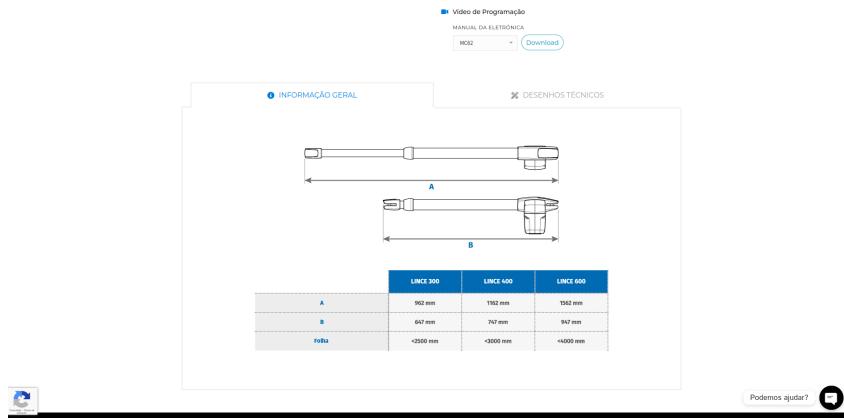


Figura 1.6: Página de detalhes de produto, secção de informações

Os videos de produtos estão disponíveis na secção de videos, sendo que cada secção de videos contém o nome do video e por sua vez o video. Estes videos são demonstrados utilizando um elemento iframe, este elemento contém um url para o video, mas após tentar visualizar este url, foi percebido que não é possível obter o vídeo a partir deste. Sendo assim foi investigada a plataforma vimeo, esta é a plataforma que contém todos os videos de produtos, pelo que para cada um é gerado um id unico e este poderá ser acedido através do url geral da plataforma seguido do id do video. Este id está também colocado no elemento iframe, pelo que este é obtido e acrescentado ao url da plataforma conseguindo assim guardar todos os videos de produtos.

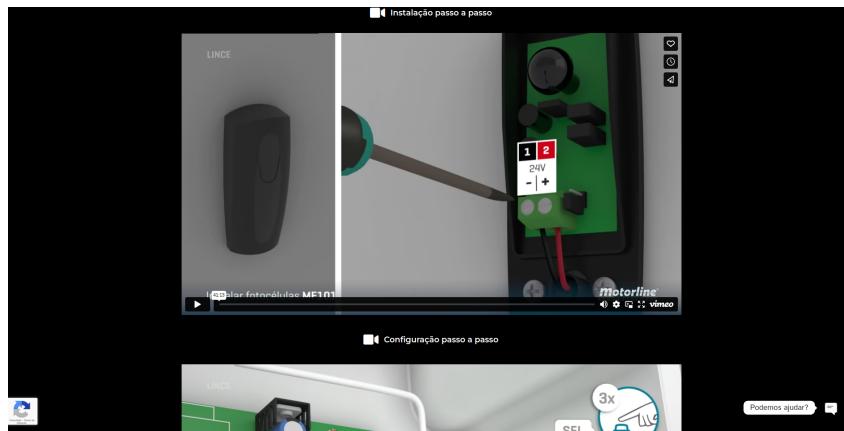


Figura 1.7: Página de detalhes de produto, secção de videos

1.1.1.1 Implementação no website

Após se verificar que eram obtidos pelo menos 80% dos produtos totais foi então decidido testar no website. Para isto foi utilizada a biblioteca requests, com a qual é realizado um pedido get a cada url necessário para se obter a página web. Assim que o código foi corrido e a resposta analisada foi percebido que o website bloqueia este tipo de solução recebendo a resposta demonstrada pela figura ??

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<p>Additionally, a 403 Forbidden
error was encountered while trying to use an ErrorDocument to handle the request.</p>
</body></html>
```

Figura 1.8: Resposta obtida quando o pedido ao url da página web

Através da resposta obtida foi então percebido que seria necessário alterar a abordagem visto que a abordagem anterior não seria possível utilizar. A abordagem opcional a seguir seria simular a ação humana abrindo um navegador e pesquisando pelo url desejado.

Após uma investigação foi descoberto que existem ferramentas que permitem controlar o dispositivo onde correm impedindo a utilização deste enquanto se encontram a correr, assim como ferramentas que apenas recebem o navegador a utilizar e abrem uma nova janela deste navegador para realizar a pesquisa. Visto que o processo de obter os produtos seria demorado, foi optado pela segunda opção visto que seria possível continuar com trabalho em paralelo com a obtenção de dados. Sendo assim a ferramenta mais recomendada para realizar esta operação é a biblioteca selenium, esta biblioteca permite realizar exatamente o processo referido anteriormente com a possibilidade de escalar com multi threading, permitindo abrir diversas janelas do navegador simultaneamente, diminuindo drasticamente o tempo de execução para obter os dados, esta funcionalidade não foi explorada devido a limitações de hardware, mas seria uma importante implementação futura.

Utilizando a biblioteca selenium foi primeiramente indicado qual o navegador a utilizar, neste caso foi escolhido o chrome devido a este já estar instalado no dispositivo. Após se indicar qual o navegador a utilizar, é necessário para cada página indicar qual elemento esperar que carregue, pois assim que a pesquisa é efetuada a página poderá demorar a carregar pelo quem se deverá indicar a espera pelo elemento que se deseja obter. Visto que a página carregada poderá não conter o elemento a obter foi então implementado um tempo de espera máximo de 5 segundos, assim que este tempo expira a operação é abortada e o url é adicionado à lista de urls com erros.

1.1.1.2 Melhoria de implementação

Após se completar o processo foi então decidido melhorar a implementação resolvendo os erros encontrados em urls específicos. De forma a perceber exatamente quais os urls que possuem erros foi então direcionado os dados obtidos para um ficheiro json. Sendo assim os urls com erros eram os indicados na figura ??.

```
"misses": [
    "https://motorline.pt/produto/acessorios-para-portoes-de-correr/",
    "https://motorline.pt/produto/acessorios-portoes-garagem/",
    "https://motorline.pt/produto/zuma/",
    "https://motorline.pt/produto/acessorios-barreira-eletromecanica/",
    "https://motorline.pt/produto/acessorios-para-portas-de-vidro/",
    "https://motorline.pt/produto/adaptador-tub/",
    "https://motorline.pt/produto/acessorios-motores-enrolar/",
    "https://motorline.pt/produto/cortina-corta-fogo-flama/",
    "https://motorline.pt/produto/acessorios-para-toldos/",
    "https://motorline.pt/produto(mb17)/",
    "https://motorline.pt/produto(mbn25)/",
    "https://motorline.pt/produto(mim1100u)/",
    "https://motorline.pt/produto/acessorios-controlo-de-acessos/",
    "https://motorline.pt/produto(stop)/",
    "https://motorline.pt/produto/acessorios-fotocelulas/"
],
```

Figura 1.9:Urls com erro primeira interação

Após uma primeira análise foi possível perceber que grande maioria dos erros provém de urls de acessórios de produtos, isto deve-se ao facto de os acessórios de produtos encontrarem-se na página de produtos de subcategoria e serem tratados como um url de detalhes de produto, sendo assim sempre que se trata de um url de acessórios seria necessário correr código para obter dados de destes ao invés de detalhes de produtos. Para desenvolver este código foi primeiramente analisada a página de acessórios de produtos(Figura ??), esta página contém para cada acessório um elemento do tipo artigo o qual contém uma imagem, título e descrição, esta descrição por vezes contém urls para os produtos aos quais este acessório se refere, pelo que sempre que estes urls são detetados, os nomes dos produtos são guardados para futuramente realizar a ligação entre os acessórios e os produtos, visto que não existem produtos com nomes iguais. Foi percebido que nos urls a palavra accessórios está sempre contida pelo que sempre que esta é detetada em um url é corrido o código referente à obtenção de acessórios.

Acessórios para Motores de Batente

Batente Lince

Batente em alumínio, para aplicação em toda a linha **LINCE**, para limitação na abertura, evitando assim a utilização de batente no portão.



RL180

Acessório que permite a abertura 180° de um motor **SUBWING**.



KIT BATENTE

Batente mecânico para um motor **SUBWING**.



Figura 1.10: Exemplo de página de acessórios

Após correr o novo código criado foi percebido que a quantidade de urls com erros diminuiu, mas existiam produtos com páginas de detalhes de produtos comuns pelo que estas foram analisadas e foi percebido que um erro ocorria devido a por vezes as páginas não conterem vídeos ou imagens de documentação, sendo que o código foi alterado para apenas obter estes dados se os elementos existirem na página. Após correr novamente o código foi percebido que a quantidade de falhas obtidas diminuiu drásticamente (Figura ??). Mas mesmo assim ainda existiam 3 falhas a ocorrer e após uma análise foi percebido que estas falhas estavam a ocorrer devido a:

1. Uma página de subcategoria de produtos conter um serviço;
2. Um produto conter uma página de detalhes de produto com subprodutos;
3. Existir uma página de adaptadores de produtos;
4. Um produto conter uma página de detalhes diferente das demais;

```
"misses": [
    "https://motorline.pt/produtos/elettronica-e-controlo/casa-inteligente/",
    "https://motorline.pt/produto/zuma/",
    "https://motorline.pt/produto/adaptador-tub/",
    "https://motorline.pt/produto/cortina-corta-fogo-flama/"
],
```

Figura 1.11: Exemplo de página de acessórios

Visto que a página de adaptadores de produtos segue uma estrutura similar à estrutura dos acessórios este foi o primeiro a ser abordado e resolvido, correndo o código de obter detalhes de acessórios sempre a palavra acessórios ou adaptadores se encontra no url. De seguida foi percebido que para resolver o problema de existirem serviços e subprodutos o diagrama de base de entidade relação teria de ser alterado pelo que primeiramente foi resolvido o problema do produto que contém uma página de detalhes diferente das demais.

Este produto para além da dificuldade de ser uma página completamente diferente as informações encontram-se espalhadas pela página (Figura ??), pelo que estas deveriam

ser combinadas para construir os detalhes do produto. Após a obtenção destes dados foi percebido que as imagens do produto têm dados escritos por cima que não se encontram na imagem original, sendo assim foi decidido em conversa com o cliente que estas seriam obtidas com screenshots e guardadas num servidor de imagens.



Figura 1.12: Exemplo de página de produto incomum

De forma a resolver o problema de existirem produtos com subprodutos, foi primeiramente alterada a base de dados adicionando uma nova ligação na tabela produtos para ela própria sendo assim um subproduto possui a chave estrangeira de produto principal preenchida. Após a alteração na base de dados, foi então seguido para a obtenção dos dados do catálogo. Para os dados foi primeiramente obtido todos os dados do produto principal e de seguida os dados específicos a cada subproduto, sendo que a organização do produto principal é igual aos restantes mas com um dado extra com os subprodutos.

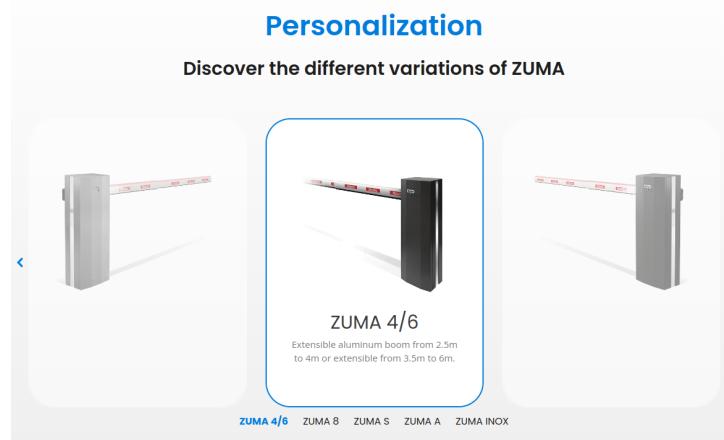


Figura 1.13: Exemplo de página de produto com subprodutos

O último problema a solucionar é a existência de serviços, iniciou-se então pela análise do serviço existente, este possui então descrição e imagens como os produtos, mas possui também vídeos direcionados a plataformas diferentes, registo de atualizações do serviço planos de pagamento de serviços, sendo que cada plano contém diversas ofertas e por fim produtos do serviço. Possuindo então a estrutura de um serviço foi adicionado à base de dados todas as tabelas de suporte a estes dados e realizadas as ligações necessárias. Após isto foi percebido que existem dados que não são necessários na descrição pelo que o cliente indicou que seria mais indicado guardar estes dados como screenshots.

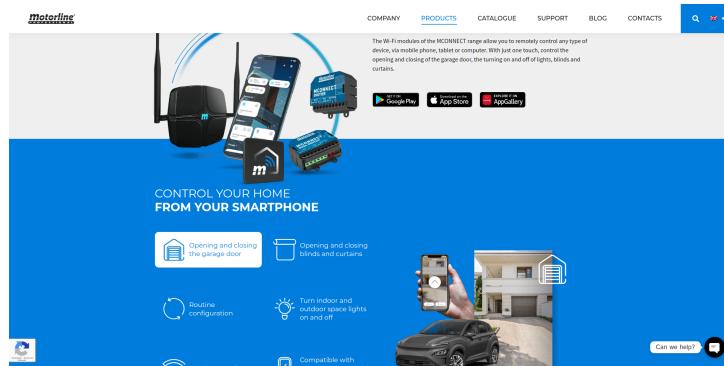


Figura 1.14: Exemplo de página de serviço

1.1.1.3 Armazenamento de dados

Após se obter os dados dos produtos, é necessário guardar estes na base de dados para disponibilizar para a sua utilização no backend. Para realizar esta operação existem duas opções, criar um serviço para inserir produtos e realizar um pedido a este serviço, ou então conectar diretamente o web scraper à base de dados. Visto que não seria de grande interesse conectar diretamente à base de dados, foi decidido criar um serviço que recebe um produto e o insere na base de dados. O grande problema que surgiu com esta solução é que os pedidos ocorrem de forma sequencial, mas com pouco tempo de espera entre estes, o que levava a que o limite máximo de conexões com a base de dados fosse extrapolado. Isto acontece porque para cada serviço chamado é criado uma nova conexão à base de dados, todas as operações são realizadas e por fim a conexão é terminada, mas enquanto estas operações estão a decorrer, o servidor poderá receber mais pedidos, o que leva a que mais conexões sejam criadas, atingindo assim rapidamente o limite de conexões da base de dados. Como solução para este problema foi acrescentado uma espera de 0.5 segundos a cada pedido. A inserção de serviços decorreu com o mesmo processo.

A inserção de categorias decorre enviando as categorias a inserir em um array, sendo inseridas todas num pedido, já as subcategorias, visto que não se sabe o id da categoria referente, foi utilizado o nome da categoria pois este é único, sendo assim é enviado o nome da categoria e as subcategorias referentes, sendo assim todas inseridas com a referência para a sua categoria.

1.2 Serviços Backend

De forma a realizar a integração entre a aplicação *frontend* e os dados, foi necessário desenvolver uma API para dar suporte a todos os serviços necessários para a aplicação. API sigla para *Application Programming Interface* disponibiliza um conjunto de funções e dados que facilita as interações entre aplicações e permite que troquem informação ?. Esta ferramenta apesar de ser desenvolvida para trabalhar em conjunto com outros programas, ela são em sua grande maioria desenvolvidas para serem entendidas e utilizadas por outros programadores no desenvolvimento dos seus programas ?.

1.2.1 Serviços REST Full

Explicar o que é

1.2.2 Organização do projeto

Antes de iniciar a implementação, foi definido qual a estrutura de projeto a seguir, pelo que a escolhida foi MVC devido a ser a mais comum e estabelecida. Sendo assim a organização do projeto segue a seguinte estrutura:

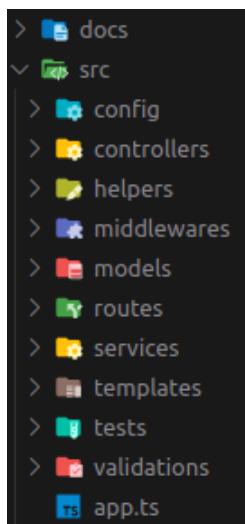


Figura 1.15: Exemplo de página de produto incomum

- **docs** - Documentação gerada;
- **src** - Base de todo o projeto;
- **config** - Ficheiros de configuração do projeto;
- **controllers** - Controladores para cada pedido;
- **helpers** - Ficheiros com funções gerais utilizadas regularmente;
- **middlewares** - Ficheiros com os middlewares da api;

- **models** - Classes criadas para representação de base de dados e para as entidades de resposta;
- **routes** - Rotas existentes;
- **services** - Serviços para cada pedido;
- **templates** - Templates de email a serem enviados;
- **tests** - Testes de código realizados;
- **validations** - Validações a realizar a nível de modelo de negócio e validação de dados;
- **app** - Ficheiro de início do projeto;

1.2.3 Definição de rotas base

Após a definição da estrutura do projeto foi então definido as rotas base a existir, estas são rotas que se referem a cada tipo de utilizador. Para melhor organização destas rotas e aplicação de regras foram definidos 3 routers, user para utilizadores se sessão, professional para técnicos e company para empresas. De forma a definir para o projeto qual o router a utilizar em cada pedido foi então definido que:

- **http://baseurl:port/professional** - Encaminhar para router de técnicos;
- **http://baseurl:port/company** - Encaminhar para router de empresas;
- **Restantes** - Encaminhar para router de user;

1.2.4 Middlewares

Um middleware comporta-se como uma ligação entre porções de código, sendo possível este também executar código.

1.2.4.1 Linguagem

O bem essencial em uma boa comunicação entre duas partes é a utilização da mesma linguagem, sendo assim foi necessário perceber qual a linguagem a utilizar quando se responde a um pedido. Para este fim foi então desenvolvido um middleware, o objetivo deste é verificar se existe a chave language no cabeçalho do pedido, caso esta exista é então obtido a linguagem e guardada nas variáveis locais do pedido. Em caso de esta tag não existir, foi então decidido que a aplicação responderá em português por omissão, este valor poderá ser futuramente alterado de forma simples.

1.2.4.2 Autenticação

De forma a assegurar a autenticação dos utilizadores que necessitam desta foi então decidido implementar JsonWebToken, este tipo de autenticação baseia-se em a utilização de tokens com tempo de expiração, sendo que enquanto o token estiver válido, o utilizador poderá realizar pedidos e assim que este token expirar este terá de se autenticar novamente para obter um novo token. A utilização de tokens permite também assegurar que os pedidos são realizados com tokens gerados pela api através de utilização de uma chave de assinatura de token, impedindo assim a utilização de tokens gerados por utilizadores.

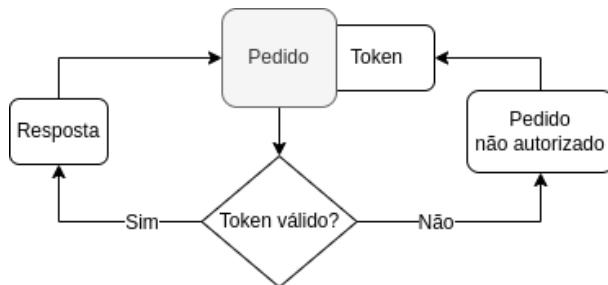


Figura 1.16: Exemplo de página de produto incomum

A grande valia da utilização a técnica de autenticação mencionada anteriormente é a segurança desta, mas este nível de segurança leva a que as aplicações que não necessitam de um nível de segurança muito alto se tornem impráticas. Isto acontece porque estes tokens têm geralmente uma duração muito curta como por exemplo 15 minutos, e sempre que um token de sessão expira o utilizador teria de realizar novamente o login.

A solução deste problema sem a perda de segurança significativa veio pelo meio da utilização de tokens de duração maior em conjunto com os tokens de duração curta, sendo que enquanto o token de grande duração estiver válido, novos tokens de curta duração são gerados para o utilizador nunca perdendo assim a sua sessão. Estes tokens de grande duração tem por nome tokens de refresh e os tokens de curta duração têm por nome tokens de sessão. Sempre que o utilizador termina a sua sessão o token de refresh deverá ser apagado.

Sempre que um utilizador realiza um pedido o seu token de sessão deverá ser validado, caso este seja válido, o seu token de refresh deverá também ser validado e apenas após isto o utilizador estará autenticado. Caso o token de sessão ou de refresh esteja expirado, este continuará a estar sem autorização para realizar o pedido, mas poderá pedir um novo token de sessão enquanto o seu token de refresh estiver válido, isto acontece sem realizar novamente o login e sem o utilizador perceber.

Além das funcionalidades atrás mencionadas é possível também associar dados em formato json a um token jwt, esta funcionalidade foi utilizada para enviar o id do utilizador a qual pertence este token e também o cargo do mesmo.

1.2.4.3 Validação de Papel

Com finalidade de garantir que apenas empresas podem realizar os pedidos de empresas e apenas técnicos e empresas podem realizar os pedidos de técnicos foi então criado um middleware que valida se o utilizador que realizou o pedido tem permissões para o mesmo.

Este middleware interliga-se com o middleware anterior pois como mencionado o cargo do utilizador em questão é enviado no token, sendo assim é obtido este cargo e realizada uma comparação, com o cargo desejado. Para isto foram criados 2 middlewares diferentes, um valida o cargo de empresas e o outro o cargo de técnicos. Visto que as empresas podem realizar operações de técnicos então no middleware de técnicos é verificado se o token corresponde a um utilizador empresa ou a um utilizador técnico, já no middleware de validação de empresa é verificado se o utilizador tem cargo de empresa.

1.2.5 Controllers

Assim que um pedido consegue passar por todos os middlewares sem ser impedido, este é então redirecionado para um controller. Um controller é

1.2.5.1 Estruturação dos controllers

De forma a evitar que o código destes controllers varie em termos de estrutura, foi então decidido desenhar uma estrutura de controller e aplicar esta perante o demais código. A estrutura deste segue as seguintes etapas:

1. Obter dados do pedido
2. Validar se os dados obrigatórios são obtidos
3. Validar o pedido perante o modelo de negócio
4. Executar a lógica do pedido
5. Formular a resposta e enviar
6. Em caso de erro este deverá ser capturado e processado de forma a enviar um erro para o utilizador

Para garantir que esta estrutura sempre será aplicada foram utilizados snippets de código que permitem criar um modelo de estrutura de código sendo apenas necessário escrever a palavra chave e toda a estrutura é escrita, necessitando de seguida de efetuar as alterações necessárias perante o contexto.

1.2.5.2 Execução da lógica de negócio

A execução da lógica de negócio passa por direcionar os dados para a ação correta, sendo que esta ação geralmente resulta em uma operação de base de dados. Inicialmente foi desenvolvida toda a validação de código e todas as operações de base de dados diretamente na execução da lógica de negócio, após uma revisão desta organização de código com o professor orientador, foi decidido separar esta funcionalidades, surgindo assim a componente de validação de dados, a componente de operações de base de dados e por fim a componente de lógica de negócio que implementa a componente de operações de base de dados. Sendo assim de forma a evitar que estas operações sobre a base de dados estejam em conjunto com o direcionamento dos dados, foram então criados modelos para cada tabela. Cada modelo contém um conjunto de operações sobre a sua tabela correspondente, estas operações estão contidas sobre métodos que podem receber dados para executar na operação e devolver a resposta da mesma.

1.2.5.3 Validação dos dados

A validação dos dados é necessária de forma a evitar erros a nível de servidor com dados em falta e também para aplicar as regras de negócio, garantindo assim que estas são cumpridas. Para realizar estas validações é primeiramente verificado que todos os dados são recebidos, de seguida estes são enviados para um validador. O validador executa todas as verificações necessárias a nível de regras de negócio e em caso de alguma regra não ser cumprida, é então atirado um erro.

1.2.5.4 Formulação da resposta

Assim como mencionado anteriormente o bem mais importante numa boa comunicação é a utilização da mesma linguagem, sendo assim a resposta do servidor deverá utilizar a linguagem indicada pelo utilizador. De forma a realizar esta tradução foi utilizado o mesmo conceito que é utilizado para a tradução de aplicações android onde nestas é criado um ficheiro que contém um conjunto de chaves e a cada chave corresponde um texto, para cada tradução estas chaves têm de existir de forma a ser possível obter o texto correto para cada chave. Sendo assim foi utilizado um ficheiro json contendo as chaves das linguagens suportadas, a cada linguagem corresponde um conjunto de outras chaves que contém todas as traduções necessárias, utilizando neste caso numeração, em vez de palavras. Esta abordagem permite que de forma fácil futuramente seja possível adicionar outras linguagens ao servidor.

Para dar suporte a este ficheiro foi criada uma operação que recebe a chave desejada e a linguagem desejada, devolvendo o texto correspondente, sendo assim na Formulação da resposta esta operação é executada indicando a chave da resposta a enviar e a linguagem desejada obtendo o texto traduzido, sendo então este devolvido para o utilizador.

1.2.5.5 Processamento de erros

Visto que não é de interesse enviar para o utilizador erros do próprio servidor, foi então decidido controlar estes, para isso foi criado um erro customizado, tendo este por base o erro da própria linguagem. Este erro recebe por parâmetro o código da tradução

da mensagem de erro. Esta abordagem permite também evitar que sempre que um erro é lançado o sistema pare. Mesmo com esta abordagem acontece que sempre que um erro é lançado por base de dados, erro de código ou de biblioteca, o erro original é chamado, pelo que foi decidido que sempre que é detetado um erro que não é do tipo do erro customizado, então será devolvido um erro com mensagem de erro de servidor evitando que dados sensíveis e desnecessários para o utilizador sejam devolvidos.

1.2.6 Logging

Logging é um processo que permite guardar informação(logs) sobre um evento. Neste contexto logging poderá ser utilizado para realizar a monitorização de pedidos e/ou monitorização de utilização de recursos do software através da análise de pedidos. Estas informações poderão até auxiliar na toma de decisões sobre o software e em quais funcionalidades deste software colocar mais atenção.

Neste projeto logging foi aplicado sobre os pedidos recebidos, assim como também os erros registados, pois uma vez que os erros são tratados, uma dificuldade encontrada foi a identificação dos erros, para resolver este problema foi então decidido que sempre que um erro que não é customizado é detetado, é registado um log, este log contém informações sobre o pedido, data e hora do pedido, dados recebidos e assim como também a descrição original do erro. Esta implementação permite assim realizar monitorização de erros auxiliando assim na identificação dos serviços mais problemáticos e para quais serviços dirigir mais recursos.

1.2.6.1 Morgan

Morgan é uma ferramenta que permite extrair dados de um pedido, assim como também a criação de logs, este atua como um middleware do servidor, recebendo qual o tipo de log a ser escrito, sendo estes tipos definidos pela ferramenta, neste caso foi utilizado o tipo combinado que permite obter todas as informações referentes ao pedido, este tipo de log recebe também a ligação ao ficheiro onde escrever estes logs. Os principais dados obtidos pela ferramenta são a data e hora do pedido, o tipo de pedido, o serviço pedido, os dados recebidos, a resposta devolvida e também a descrição do sistema utilizado para realizar o pedido, com estes dados é possível saber que plataforma é mais utilizada no software, quais as horas de maior utilização e quais os serviços mais executados, estes dados permitem direcionar mais recursos para uma indicada plataforma e/ou serviço, assim como também escolher os melhores horários de manutenção dos servidores.

1.2.7 Documentação

De forma a ser possível manter todo o projeto desenvolvido foi criada documentação a diferentes níveis, sendo esta desenvolvida a nível de serviços explicando o objetivo do serviço e que dados este recebe, como também a nível de código explicando o código desenvolvido em cada script existente. Para estes níveis de documentação foram utilizadas diferentes ferramentas, para documentação de serviços, foi utilizada a ferramenta swagger e para a documentação foi utilizado typedoc.

1.2.7.1 Typedoc

Typedoc é uma ferramenta que faz utilização de comentários de código para gerar a sua documentação, esta documentação utiliza chaves específicas para detetar as informações de documentação, estas permitem também criar categorias para melhor organizar toda a documentação. Esta documentação permite também a interligação entre si mesma permitindo ao visualizador desta seguir todo o processo. Após a realização de geração de documentação, a ferramenta gera um website onde é possível navegar por toda a documentação gerada.

Durante a implementação desta ferramenta foi detetado que a categorização de documentação não se encontrava funcional devido a um problema encontrado pelos desenvolvedores da ferramenta, foi decidido então reduzir a versão da mesma para uma com a funcionalidade ativada, mas esta não se encontrava compatível com a versão mais atualizada de typescript, pelo que não foi possível explorar esta funcionalidade. Para contornar o problema foi então decidido explorar outra funcionalidade menos utilizada da ferramenta, esta funcionalidade permite converter qualquer documentação em módulos, estes módulos podem então ser categorizados, o problema destes módulos é que cria um modelo genérico do código não sendo facilmente identificado as tipagens de scripts. Estes módulos permitem também a categorização dos mesmos permitindo assim um nível de organização da documentação gerada.

Modules - Controllers			> Settings	
AccountActivationController	AccountConfirmController	AddAnswerImagesController	▼ Modules	
AddCategoriesController	AddProductController	AddServiceController	Install & Go Documentation	
AddSubcategoriesController	AddTopicController	AddTopicImagesController	AccountActivationController	
AnswerCommentController	AnswerTopicController	AskNewActivationCodeController	AccountActivationService	
AskNewPasswordController	ChangeTopicVisibilityController	DeleteNotificationController	AccountConfirmController	
EndTopicController	GetAnswerSonsController	GetCategoriesController	AccountConfirmService	
GetCompanyProfessionals	GetForumCategoriesController	GetLanguagesController	AddAnswerImagesController	
GetMoreAnswersController	GetNotificationsController	GetProductDetailsController	AddAnswerImagesService	
GetProductsController	GetProfileController	GetTopicDetailsController	AddCategoriesController	
GetTopicsController	LikeAnswerController	LikeTopicController	AddProductController	
LoginController	NewSessionTokenController	PreventAccessController	AddProductService	
RegisterCompanyController	RegisterProfessionalController	RemoveAnswerController	AddServiceController	
RemoveProfessionalAccountController	RemoveTopicController	SetBestAnswerController	AddServiceService	
SetDeviceTokenController	SetLanguageController	SetUpPasswordController	AddSubcategoriesController	
TestCleanupPdController	TestsCleanupCatController	TestsCleanupController	AddSubcategoriesService	
UpdateProfileController			AddTopicController	
			AddTopicImagesController	
			AddTopicImagesService	
			AddTopicService	
Modules - DbModels				
Answer	Categories	Languages		
Notifications	ProductControlBoard	ProductImages		
ProductManuals	ProductVideos	Products		
ServiceImages	ServicePlanOfferings	ServicePlans		
ServiceUpdateLogs	ServiceVideos	Services		
Subcategories	Topic	User		
Modules - Interfaces				
IKeyValue	IPlatformVideo	IProduct		
IService	IServicePlan	IServiceUpdate		
ITopic	ITopicAnswer	IUser		

Figura 1.17: Página inicial da documentação gerada

1.2.7.2 Swagger

Swagger é uma ferramenta que permite gerar documentação a nível de serviços, esta documentação é acessível a partir do mesmo servidor indicando uma rota para o mesmo, evitando assim outro servidor para hospedar a documentação, a base de toda a documentação encontra-se em um ficheiro no formato json. Esta documentação poderá ser gerada a partir de comentários a nível de código ou a partir de um ficheiro em formato json como mencionado anteriormente. Este ficheiro em formato json poderá ser mantido manualmente ou automaticamente.

Apesar da ferramenta disponibilizar a geração automática de documentação a partir de comentários de código, foram encontrados alguns problemas com esta funcionalidade, acabando por não gerar a documentação, pelo que foi optado por manter a documentação manualmente com o ficheiro json. Esta ferramenta oferece diversas funcionalidades como autenticação, definição de estruturas de dados para os serviços e exemplos de respostas para os mesmos, ambas estas funcionalidades foram exploradas conseguindo assim um bom suporte de documentação para qualquer utilizador.

The screenshot shows the Swagger UI interface for the 'Install&Go API'. At the top, it says 'Install&Go API 0.0.0 (GAS)'. Below that, there's a 'Servers' dropdown set to 'http://12.48.213.162:7856/api/v1 - External docs'. On the right, there's an 'Authorize' button. The main area is divided into sections: 'Products' (with GET /products, POST /products, GET /products/{productId}, DELETE /tests/product/{productId}/cleanup, and DELETE /tests/category/{catName}/{subName}/cleanup), 'Services' (with POST /services), and 'Categories' (with GET /categories). Each section lists its corresponding REST endpoints and methods.

Figura 1.18: Documentação swagger

This screenshot shows a detailed view of an API endpoint in the Swagger UI. The endpoint is `DELETE /tests/category/{catName}/{subName}/cleanup`. The 'Parameters' section includes required parameters: `catName` (path), `subName` (path), and `language` (header). The 'Responses' section shows two possible outcomes: a successful response (200 OK) with a media type of `application/json` containing the example value `"Cleanup done"`, and an error response (500 ERROR) with no links. There is also a 'Links' section which is currently empty.

Figura 1.19: Exemplo de documentação de serviço

Bibliografia

Jin, Brenda, Saurabh Sahni & Amir Shevat. 2018. *Designing web apis.* O'Reilly Media, Inc.

Masse, Mark. 2011. *Rest api design rulebook.* O'Reilly Media, Inc.