Sistemas de Informação 2014/2015

Loja Online

4º ano do Mestrado Integrado em Engenharia Informática e Computação

João Carlos Costa Pinto – ei11091 – ei11091@fe.up.pt
João Manuel Mesquita Cardoso – ei11100 – ei11100@fe.up.pt
Luís Filipe Ferreira Araújo – ei11059 – ei11059@fe.up.pt
Miguel Ferreira da Cunha Poeira – ei11143 – ei11143@fe.up.pt
Wilson da Silva Oliveira – ei11085 – ei11085@fe.up.pt

Índice

Introdução

Este trabalho, no âmbito da Unidade Curricular de Sistemas de Informação do 4º ano do Mestrado Integrado em Engenharia Informática e Computação, visa o desenvolvimento de uma aplicação, que possa ser aplicada numa loja online, em *html5* de forma a permitir consultar um catálogo *online*, os respectivos *stocks*, registar uma encomenda e criar utilizadores.

Para o desenvolvimento e demonstração desta aplicação, aplicamos a mesma a uma loja de venda de telemóveis, sendo assim criado no *ERP (Enterprise Resource Planning)* os artigos e campos necessários para este cenário.

Tentamos aproveitar o ERP de forma a criar valor, tanto para o cliente como para a própria loja. Isto porque o ERP faz a gestão de muitos pontos que são importantes para o cliente e que também podem potenciar determinados produtos que a loja pretende demonstrar.

Framework e Tecnologias Adotadas

Foi utilizado *Node.js* para o servidor *web* e *AngularJS* para o *front-end*, ambas frameworks de *javascript*.

O *Node.js* é uma plataforma *cross-platform* baseada no motor de *Javascript GoogleV8* para desenvolver aplicações do lado do servidor rápidas e escaláveis, como tal, as aplicações de *node.js* são escritas em *Javascript*. O *Node.js* usa um modelo *non-blocking* e *event-driven* que o torna eficiente, leve e optimiza a escalabilidade, perfeito, desta forma, para aplicações real-time e com grande volume de dados que correm em vários dispositivos distribuídos.

O *AngularJS* é uma *framework cross-platform* para aplicações *web* mantida pela *Google,* escrita em *Javascript* e que facilita o desenvolvimento da parte do *front-end* da aplicação *web* (desenvolvimento de vistas dinâmicas), através de uma arquitetura *MVC* (*Model-View-Controller*).

Optámos por esta solução pois queríamos usar tecnologias novas e recentes, visto *Node.js* e *AngularJS* estarem muito em voga hoje em dia e dado que estamos a utilizar estas mesmas tecnologias na Unidade Curricular de Laboratório de Desenvolvimento de Software. Já estávamos um pouco ambientados a desenvolver com as mesmas ferramentas, sendo que, em vez de haver uma base de dados nossa, como em LDSO, é feito o pedido à *API RESTful* que comunica com o *ERP*. A utilização destas tecnologias também nos permite o desenvolvimento de páginas *web* mais dinâmicas e completas, o que se traduz numa melhor usabilidade por parte do utilizador final.

A *API RESTful* foi feita em *C#,* pois é esta a linguagem para a qual o *ERP* já tem uma *API*, o que fez com que o trabalho se tenha desenvolvido de uma forma mais rápida. Além disso, *C#* facilita o desenvolvimento de *APIs*, ao fornecer ferramentas que tornam este trabalho mais fácil, célere e direto.

User stories

São estas as *user stories* que nós consideramos mais essenciais na nossa aplicação web:

- Eu, como um **utilizador não registado**, quero poder registar-me no *website* para me poder autenticar, aceder aos meus dados e fazer encomendas.
- Eu, como **utilizador não registado**, quero poder ver todos os produtos disponíveis no *website*.
- Eu, como **utilizador não registado**, quero poder ver todos os produtos em promoção.
- Eu, como **utilizador não registado**, quero poder ver um produto em específico.
- Eu, como **utilizador não registado**, quero poder filtrar os produtos existentes por um parâmetro em específico.
- Eu, como **utilizador registado**, quero poder fazer *logout* no sistema para poder terminar a minha sessão.
- Eu, como **utilizador registado**, quero poder ver as meus dados de utilizador.
- Eu, como **utilizador registado**, quero poder adicionar um produto ao carrinho.
- Eu, como **utilizador registado**, quero poder finalizar a minha encomenda e desta forma encomendar todos os produtos existentes no meu carrinho.
- Eu, como **utilizador registado**, quero poder ver informação comum sobre as minhas encomendas já efetuadas.
- Eu, como **utilizador registado**, quero poder ver em detalhe as minhas encomendas efetuadas.

Core Views

A nossa aplicação *web* tenta tirar o máximo partido das capacidades do *ERP*, de modo a promover os produtos que possam trazer maiores benefícios para o vendedor e para os clientes.

Para conseguir este objectivo, a nossa página principal apresenta alguns produtos selecionados pelo *ERP*, evidenciando produtos específicos, como por exemplo um produto recente, ou com uma boa promoção, isto é, produtos cuja venda é bastante benéfica para o vendedor e para os clientes.

A aplicação possui também uma página com todos os produtos disponíveis na loja que permite ao utilizador filtrar os produtos por vários parâmetros, como o preço, sistema operativo, marca, entre outros.

Dispomos também de uma página específica para apresentar todas as promoções disponíveis na loja (produtos assinalados com desconto no *ERP*).

Existe, ainda, a possibilidade de adicionar produtos a um carrinho de compras virtual, que, numa fase posterior, se traduzirá na criação da encomenda final.

Finalmente, é possível ao utilizador visualizar as suas encomendas, numa vista geral, com informação do preço final e quantidade de produtos, ou numa vista mais especifica com indicação dos produtos e quantidades adquiridas, bem como o seu preço individual.

No desenvolvimento desta aplicação decidimo-nos focar na apresentação e seleção de produtos, assim como na criação de encomendas para demonstrar a capacidade do *ERP* quando associado a uma plataforma *web*.

O valor acrescentado está presente em todas as páginas do nosso site: desde a visualização do estado de uma encomenda que tenhamos efetuado, até à ordenação criteriosa de produtos que a loja vende.

Esta ordenação tem em conta vários parâmetros: *stock*, desconto, se é um produto novo e se é um produto já um pouco antigo, correndo o risco de ficar obsoleto. A loja tem

todo o interesse em vender todos os telemóveis que tenham mais *stock.* E, sendo assim, todos estes, quando estão em igualdade nos outros critérios, aparecem em primeiro lugar em todas as listagem de produtos. O mesmo acontece com os restantes critérios acima referidos.

Da mesma forma, esta ordenação faz com que o cliente encontre algo mais à sua medida, dado que são apresentados primeiro os produtos que têm um preço inferior ao habitual (desconto) ou aqueles que são a última novidade. Considerando que a tecnologia está em constante evolução, o último modelo de telemóvel que está disponível na nossa loja, será sempre uma boa sugestão para pôr no topo da lista de pesquisa de todos os clientes.

Desta forma, concilia-se ambos os interesses dos clientes e dos lojistas.

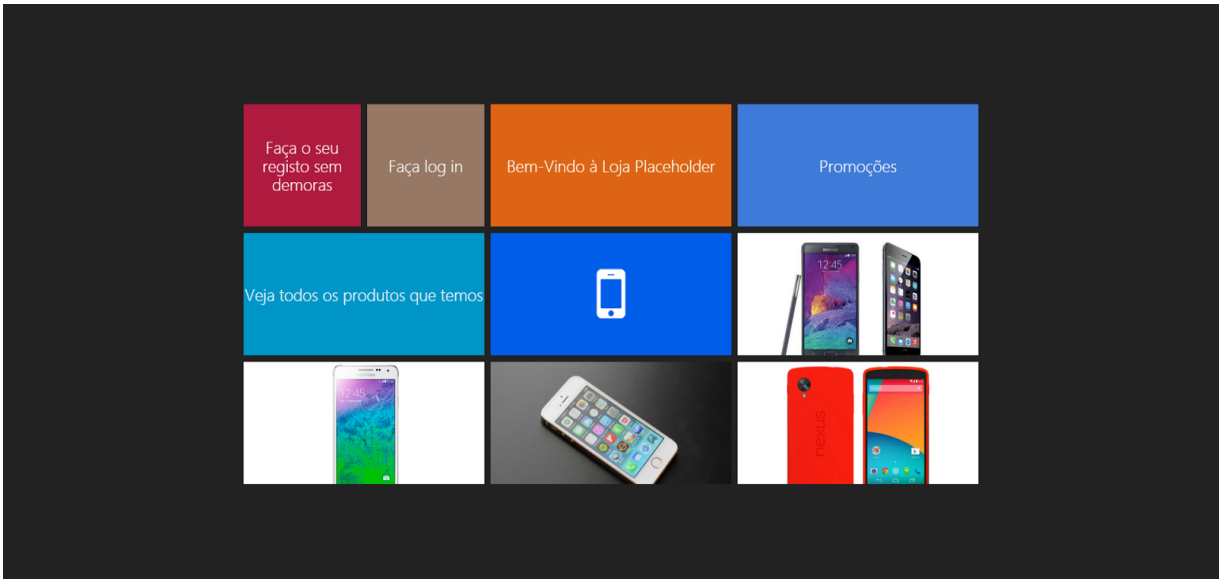


Fig. 1 - Homepage user não logado

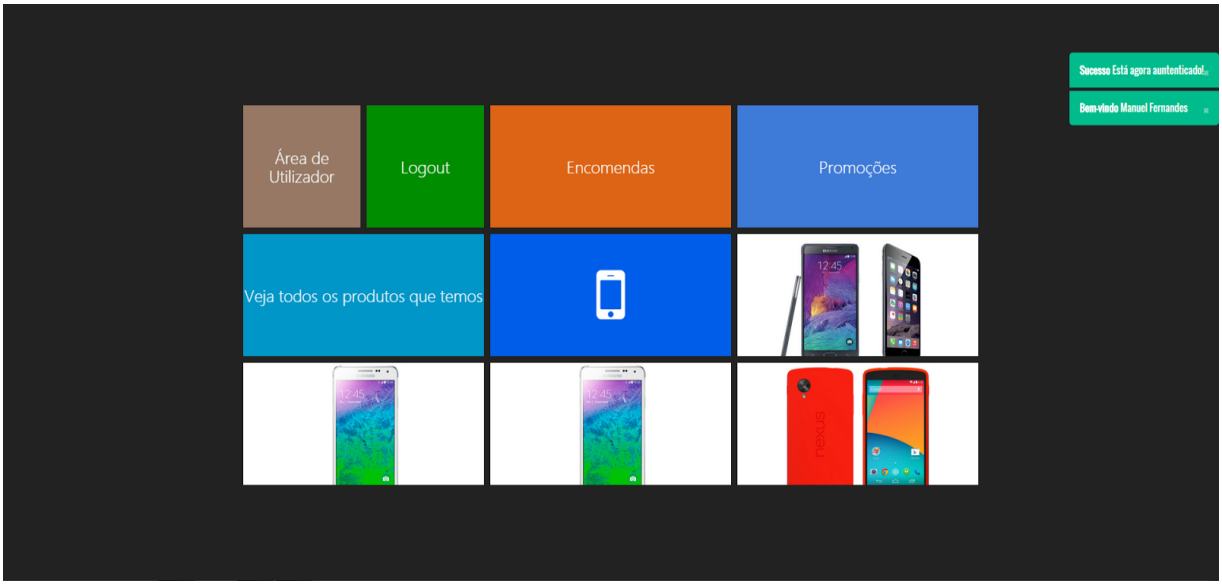


Fig. 2 - Homepage user logado

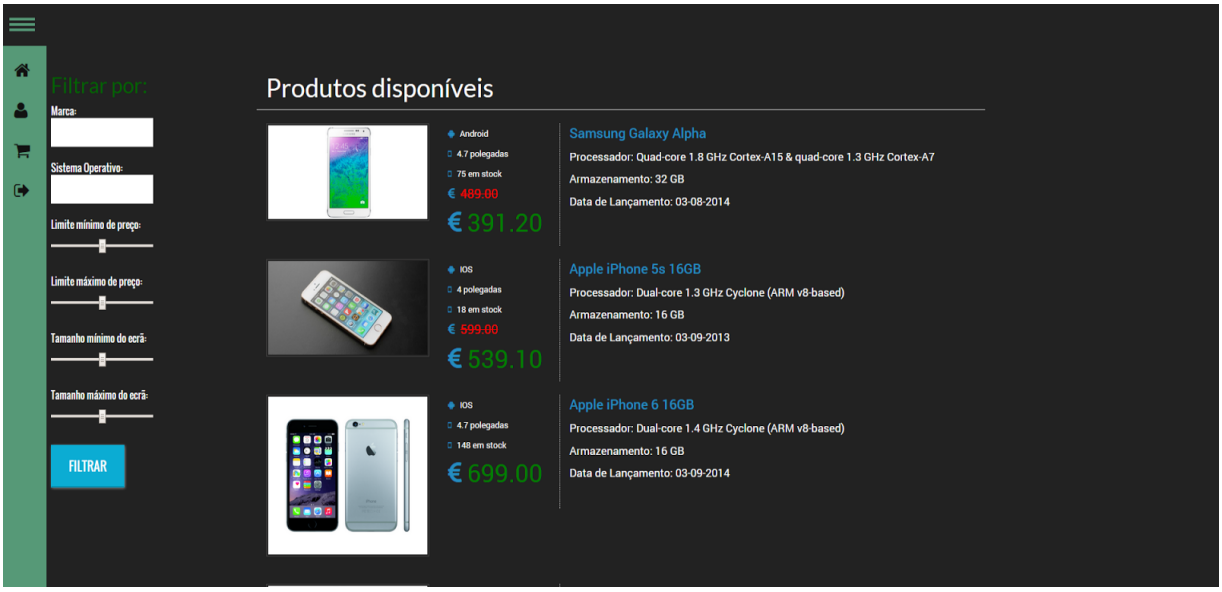


Fig. 3 - Página produtos

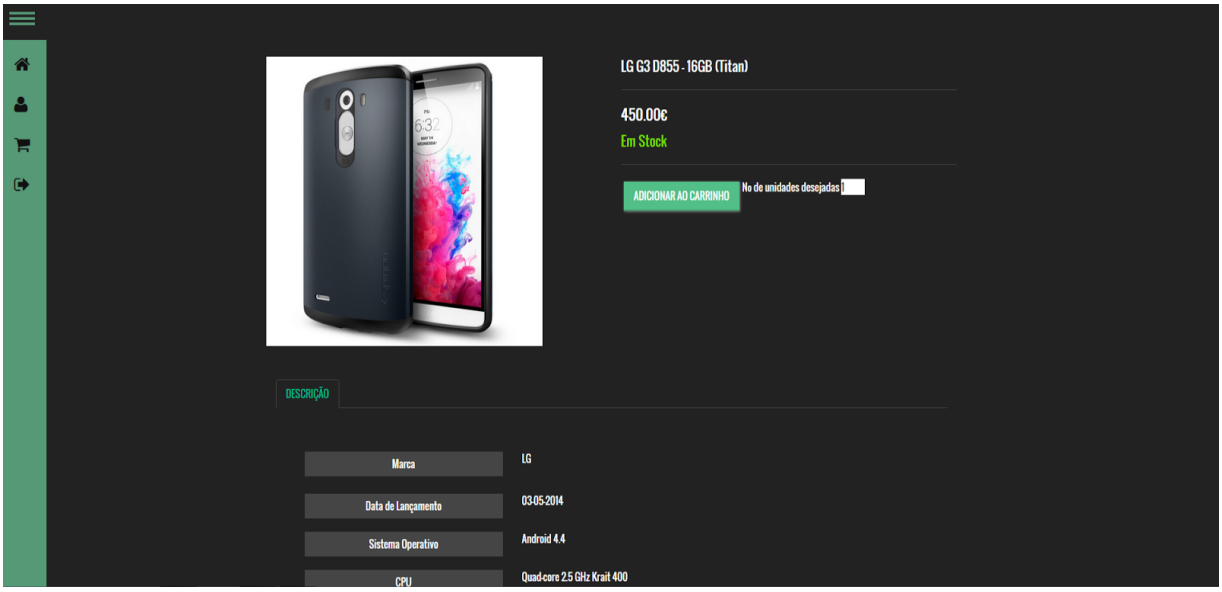


Fig. 4 - Página produto único

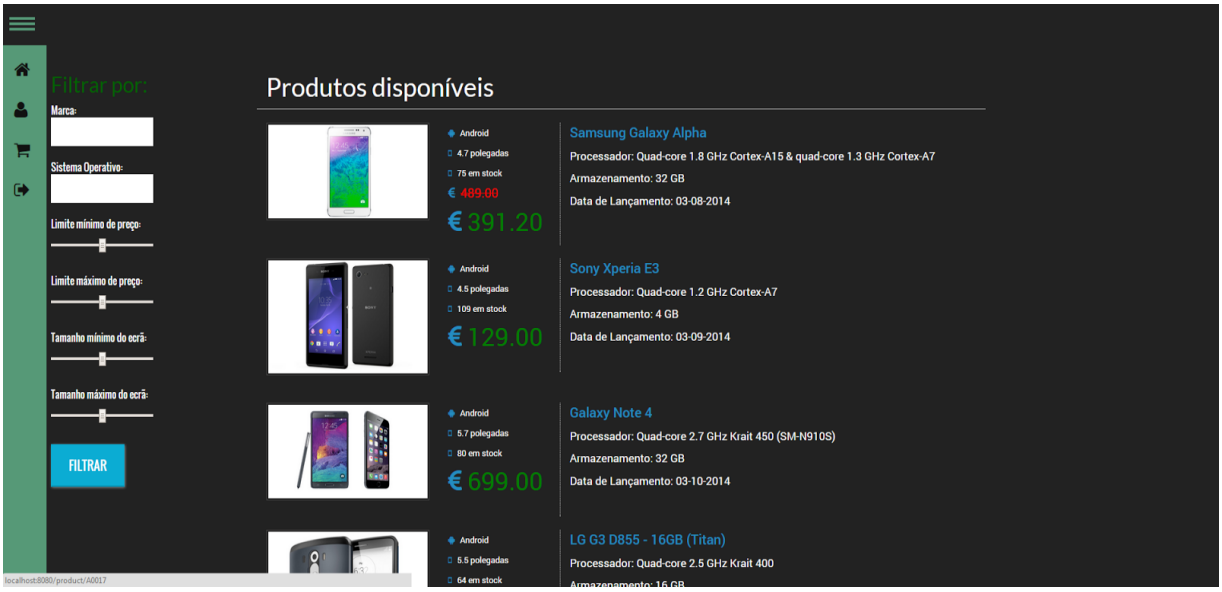


Fig. 5 - Página produtos filtrados

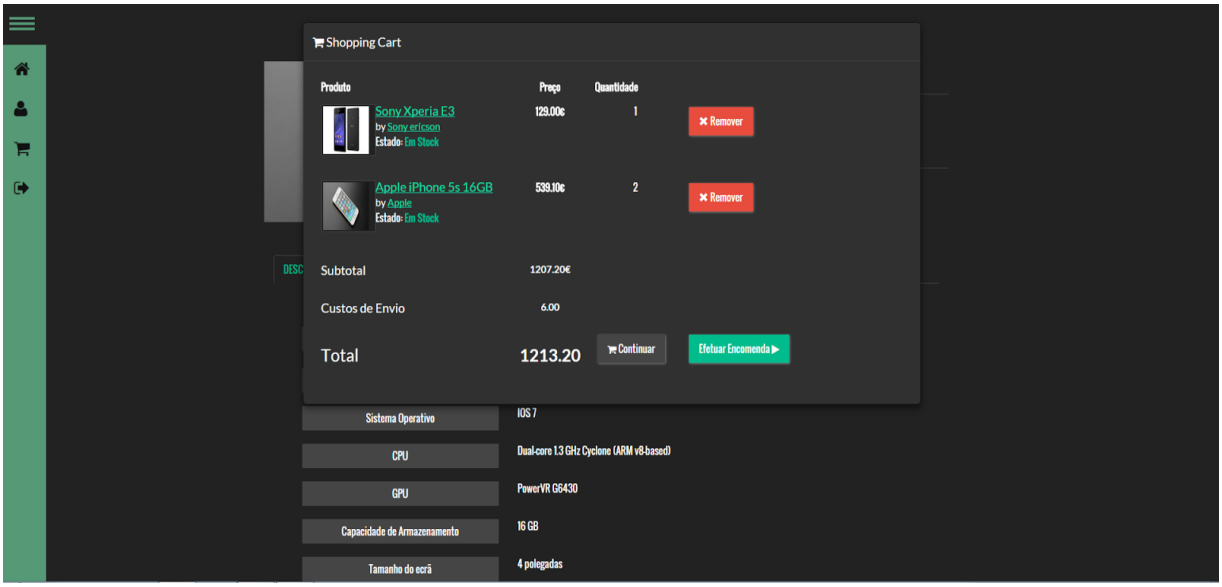


Fig. 6 - Carrinho de compras

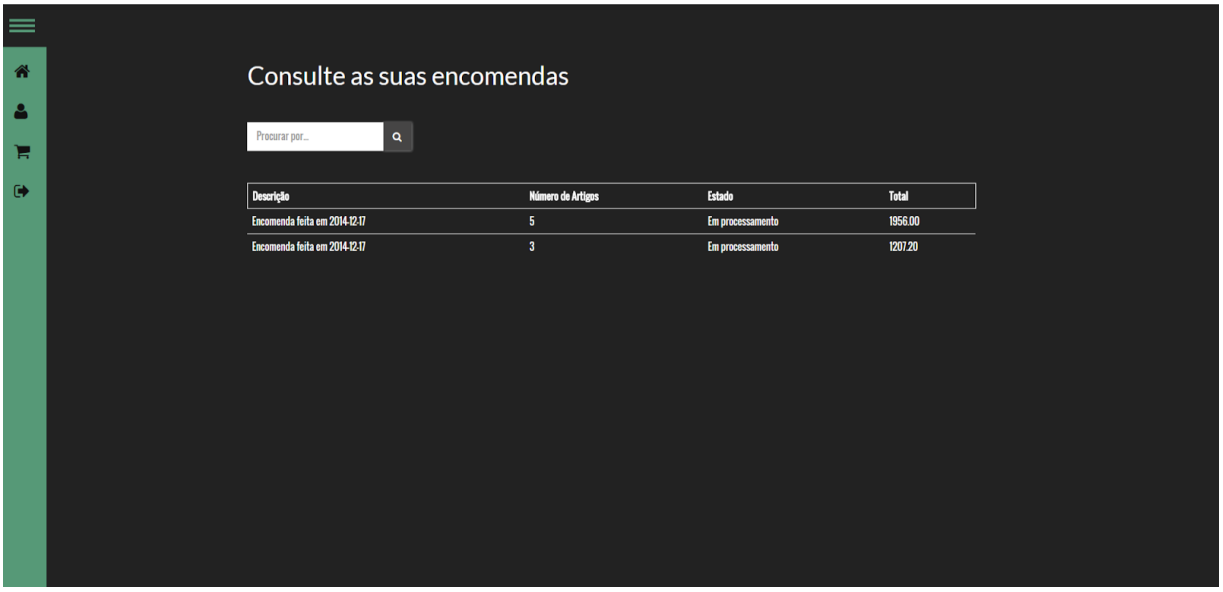


Fig. 7 - Página encomendas

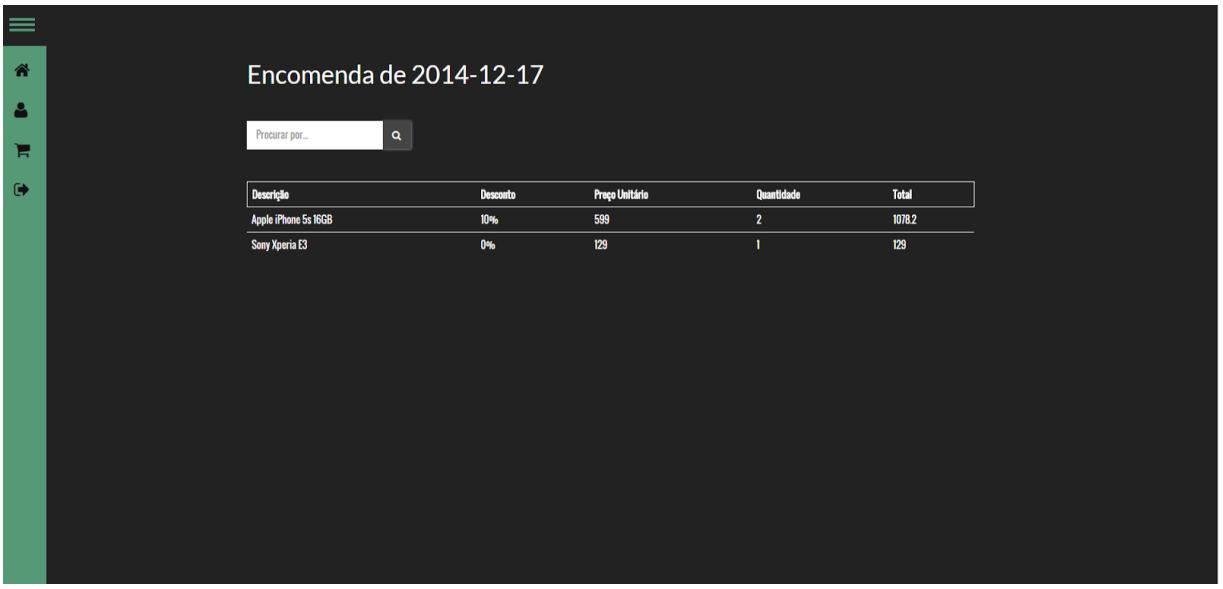
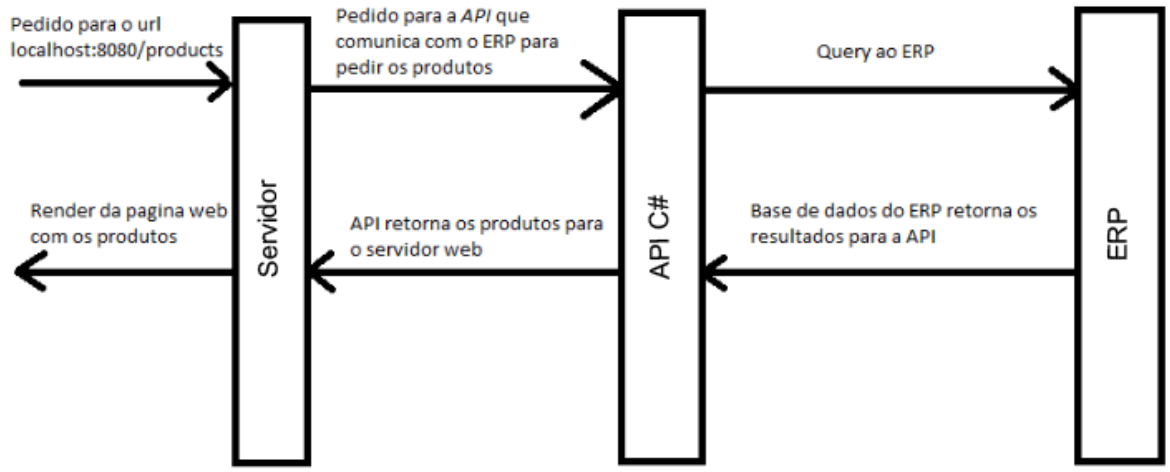


Fig. 8 - Página encomenda única

Arquitetura Tecnológica

Em termos concretos, quando um utilizador quer aceder, por exemplo, à página dos produtos, o servidor web recebe o pedido, faz o pedido à *API Restful* para ela lhe retornar os produtos (neste caso apenas os produtos da "primeira página") e, se houver sucesso no retorno da *API*, então passa os resultados retornados para a vista e, de seguida, é apresentada a página *web* ao utilizador.



De seguida, apresentamos as chamadas à *API* por nós construída em C# (que utiliza a *API* do *ERP*):

PUT Cliente
URL: /api/Clients
Type: PUT
Parâmetros: Morada, Localidade, CodPostal, Nome, NumContribuinte, Email, Telefone, Password
Retorno:

- **Sucesso**: Codigo retorno (HttpStatusCode - Created): 201
- **Erro**: Codigo retorno (HttpStatusCode - Bad request): 400

GET Cliente
URL: /api/clients/CodigoCliente (exemplo: CL000004)
Type: GET
Retorno:

- **Sucesso**: Codigo retorno (HttpStatusCode - OK): 200
- **Erro**: Codigo retorno (HttpStatusCode - Not found): 404


```
{
    CodigoCliente: "CL000004"
    Nome: "Manel"
    Telefone: "919191919"
    Email: "gmail@gmail.com"
    Password: null
    Morada: "Rua das ruas"
    Localidade: "Ali"
    CodPostal: "4995-564"
    NumContribuinte: "505456010"
}
```


POST Session
URL: /api/Sessions
Type: POST
Parâmetros: Email, Password
Retorno:

- **Sucesso**:
  - Codigo retorno (HttpStatusCode - OK): 200
  - {
    - Email: gmail@gmail.com,
    - Password: password,
    - CodigoCliente: CL000004
  - }
- **Erro**: Codigo retorno (HttpStatusCode - Bad request): 400

GET Todos os Artigos
URL: /api/artigos
Type: GET
Parâmetros opcionais:

- page
- promocao (só mostra artigos que estejam em promocao).
- precoLimiteSuperior
- precoLimiteInferior
- ecraLimiteInferior
- ecraLimiteSuperior
- marca
- so (sistema operativo)

Retorno:


```
-1: {
    avaliacao: 435
    CodigoArtigo: "A0010"
    Nome: "Huawei Ascend P7"
    StockAtual: 35
    PVP: 289
    Desconto: 0
    Marca: "Huawei"
    NomeSistemaOperativo: "Android"
    TamanhoEcra: 5
    CPU: "Quad-core 1.8 GHz Cortex-A9"
    Armazenamento: 16
    fotoURL: ""
    Lancamento: "03-06-2014"
}
```

- Objecto JSON

GET Um Artigo
URL: /api/artigo/CodigoArtigo
Exemplo de URL: /api/artigo/A0001
Type: GET
Retorno:

- **Sucesso**: 200 OK e JSON, em que
  - RAM em megas
  - Peso em gramas
  - Camara em megapixeis
  - Armazenamento em GB
  - Autonomia em horas
  - PVP em euros
  - Desconto é a percentagem a aplicar ao PVP
  - TamanhoEcra em polegadas
  - data de lançamento, no formato dd-mm-aaaa
- **Erro**: 404 Not Found (quando o CodigoArtigo é inválido)


```
{
    Marca: "LG"
    VersaoSistemaOperativo: "4.4"
    RAM: 2048
    Peso: 130
    CamaraTraseira: 8
    Armazenamento: 16
    Autonomia: 17
    GPU: "Adreno 330"
    CodigoArtigo: "A0001"
    Nome: "LG Nexus 5 16GB"
    StockAtual: 50
    PVP: 250
    Desconto: 0
    NomeSistemaOperativo: "Android"
    TamanhoEcra: 4.95
    CPU: "Quad-core 2.3 GHz Krait 400"
    fotoURL: ""
    Lancamento: "03-10-2013"
}
```


GET Artigo para Homepage
URL: /api/artigos/homepage
Type: GET
Retorno:

- **Sucesso**: JSON


```
{
  -novidade: {
       CodigoArtigo: "A0009"
       fotoURL: ""
    }
  -oportunidade: {
       CodigoArtigo: "A0002"
       fotoURL: ""
    }
  -maisvendido: {
       CodigoArtigo: "A0001"
       fotoURL: ""
    }
  -stock: {
       CodigoArtigo: "A0014"
       fotoURL: ""
    }
}
```

PUT Encomenda
URL: /api/encomendas
Type: PUT
Parametros: Entidade, LinhasEncomendas (tendo cada linha encomenda o CodigoArtigo e Quantidade)

- **Exemplo**:

```
{ "Entidade": "CL000004",
  "LinhasEncomenda" : [
        {"CodigoArtigo":"A0001", "Quantidade":1},
        {"CodigoArtigo":"A0002", "Quantidade":2}
]}
```

Retorno:

- **Sucesso**: Codigo retorno (HttpStatusCode - Created): 201
- **Erro**: Codigo retorno (HttpStatusCode - Bad request): 400

GET Encomendas do Cliente
URL:
/api/encomendas?CodigoCliente=CL000004
Type: GET
Retorno:

- **Sucesso**: Codigo retorno (OK): 200

- **Erro**: Codigo retorno (Bad request): 400

GET Encomenda
URL: /api/encomendas/NumDoc (inteiro >= 1)
Type: GET
Parametros: -
Retorno:

- **Sucesso**: Codigo retorno (OK): 200
- **Erro**: Codigo retorno (Not found): 404


```
{
  -LinhasEncomendaExtended: [2]
     -0:  {
          DescricaoArtigo: "LG Nexus 5 16GB"
          Desconto: 0
          PrecoUnitario: 250
          TotalILiquido: 250
          TotalLiquido: 250
          CodigoArtigo: "A0001"
          Quantidade: 1
      }
     -1:  {
          DescricaoArtigo: "Apple iPhone 5s 16GB"
          Desconto: 0
          PrecoUnitario: 599
          TotalILiquido: 1198
          TotalLiquido: 1198
          CodigoArtigo: "A0002"
          Quantidade: 2
      }
    Entidade: "CL000004"
    Data: "2014-12-06T00:00:00"
    NumDoc: 14
    Estado: "Em processamento"
}
```

Manual de Utilizador

Para iniciar a aplicação é necessário efetuar alguns passos para iniciar tanto o servidor assim como a *API*:

1. Executar o Visual Studio express
2. Abrir o projeto que se encontra na pasta "C:\Users\user\Desktop\SINF_Loja_Online\API"
3. Executar o projeto (irá abrir uma janela do *browser Chrome*)
4. Executar a git bash
5. Executar o seguinte comando: cd Desktop\SINF_Loja_Online
6. Executar o seguinte comando: node app/server/app.js

Após efetuar estes passos, é possível aceder à aplicação no endereço http://localhost:8080 no *browser*.

Conclusão

A utilização do *ERP* em conjunto com a aplicação que criamos, permite criar um valor acrescido, tanto para a loja, como para os clientes.

Acreditamos que a criação da fidelidade de um cliente é feita através de pequenos pormenores, daí o foco que foi colocado na *UX* e na facilidade com que o cliente é introduzido aos produtos, assim como a facilidade com que este efetua, com facilidade e rapidez, uma encomenda. Desta forma, o cliente sente-se facilmente familiarizado no ambiente da loja e a sugestão, subtil e não-intrusiva, de produtos acrescenta o valor necessário e requirido para a situação em questão.

Analogamente, este cenário é do agrado dos lojistas, dado que potencia o escoamento de *stocks* e o aumento de vendas, dado que são sugeridos produtos que temos mais interesse (e dificuldade) em vender.

Desta forma, a satisfação dos atuais clientes irá trazer novos, possibilitando o aumento do número de vendas.

Com a execução deste trabalho, demostramos que, utilizando o ERP e uma aplicação *HTML5,* que o extende, dando uso da gestão de stocks, produtos e encomendas que este efetua, é possível criar um ambiente virtual de uma loja com bastantes benefícios.