UNIVERSIDADE
LUSÓFONA

Plataforma de gestão para uma lavandaria e engomadoria

Trabalho Final de Curso

Relatório Final

Luís Miguel Galo

Miguel Bastos

Prof. Pedro Perdigão

Trabalho Final de Curso | LEI | 08/09/2023

Direitos de cópia

Resumo

Neste trabalho final de curso é proposto o desenvolvimento de uma aplicação web para a cadeia de lavanderias, Lavanderia Tejo. Esta aplicação web tem como objetivo melhorar e simplificar a gestão de informação na lavandaria e engomadoria. A parte de engomadoria apresenta diversos problemas atuais na gestão de informação, tais como o facto de a base de dados ser em folhas de papel, assim como toda a documentação da lavandaria e engomadoria.

Queremos resolver esses problemas a partir de uma aplicação *web* em React.js com o suporte de uma API desenvolvida em .NET Web API complementado por uma base de dados em SQL server.

A partir dessa aplicação, queremos que os clientes da Lavanderia Tejo possam verificar se as máquinas estão em uso no local, quantas peças de roupa e camisas ainda têm disponíveis para lavar tal como a validade do seu cartão de cliente. Pensando também na parte administrativa, queremos desenvolver soluções que permitam gerir os gastos das máquinas e dos serviços como por exemplo: detergente, água, luz, etc.. e adicionalmente que os funcionários da lavanderia possam verificar as encomendas ainda pendentes para entrega.

Este projeto está protegido com um *non-disclosure agreement*, tendo assim uma impossibilidade de partilhar qualquer tipo de repositório, como código fonte, imagens, dados, ficheiros da própria empresa, bem como os dados da base de dados devido a segurança.

Abstract

In this final course work it is proposed the development of a web application for the laundry chain, Lavanderia Tejo. This web application aims to improve and simplify the information management in laundry and ironing. The ironing part presents several current problems in information management, such as the fact that the database is on paper sheets, as well as all the laundry and ironing documentation. .

We want to solve these problems with a *web* application in React.js supported by an API developed in .NET Web API that being complemented by a database in SQL server.

From this application, we want the Lavanderia Tejo customers to be able to check if the machines are in use on site, how many clothes and shirts they still have available for washing as well as the validity of their customer card. Looking at the administrative part, we want to develop solutions that allow us to manage the costs of the machines and services, such as detergent, water, electricity and others, and additionally allow the laundry staff to check the orders that are still pending for delivery for example.

This project is protected with a non-disclosure agreement, making it impossible to share any kind of repository, such as source code, images, data, the company's own files, as well as the database data due to security reasons.

Índice

Lista de Figuras

Lista de Tabelas

1 Identificação do Problema

A lavandaria Tejo é uma empresa que atualmente tem três lojas em funcionamento, A primeira loja foi inaugurada em 2016 na Póvoa de Santa Iria e à data a mesma possui três máquinas de lavar, seis máquinas de secar e onde já passaram mais de 10.000 clientes e onde mais 1000 clientes tem sua ficha na engomadoria. Desde o início da operação da loja toda a administração é realizada de forma manual (em papel) e sem sistemas de informação a apoiar a gestão da mesma.

Tendo em conta o crescimento das lojas face à elevada procura pelos seus clientes é imperativo a criação de uma software orientado à *Web* que facilite e apoie a gestão da mesma e envolva mais os seus clientes numa jornada de proximidade digital.

Face ao:

- pouco controle atual;
- à falta de dados para analisar e agir sobre;
- à difícil gestão e proximidade com os clientes;
- à falta de página *web* com informação sobre as lojas e os seus serviços;
- à falta de informação sobre as máquinas disponíveis e tabela de preços atualizados.

Propomos desta forma, ajudar esta empresa numa jornada de transformação digital ao desenvolver uma aplicação *web* e *híbrida* que visa resolver muitos problemas supramencionados, facilitar na gestão de todas as lavandarias e aumentar a fidelização dos atuais clientes e o crescimento de novos clientes.

Adicionalmente, a nossa solução visa centralizar todos os dados num único repositório para análise posterior e experiências de utilizador distintas face às necessidades dos nossos clientes.

2 Viabilidade e Pertinência

Este projeto está a ser desenvolvido a pedido do destinatário, Lavandaria Tejo.

O desenvolvimento é de interesse para o aumento da eficiência e da possibilidade de automatizar e gerir algumas das tarefas executadas manualmente, para um melhor desempenho.

Como descrito no Benchmarking figure 6, 94,7% das pessoas questionadas nunca usaram um serviço como este. Acreditamos então que existe um espaço no mercado português para uma plataforma como esta.

O software *Web* é desenvolvido com tecnologias modernas como React que é uma biblioteca JavaScript em constante desenvolvimento e evolução o que torna o futuro do nosso serviço claro e seguro.

Adicionalmente, vai ser usada uma API do tipo REST que um dos seus pontos fortes é ser extremamente escalável, o que significa que se o negócio expandir, essa expansão não impactará de modo negativo a aplicação e torna fácil implementar mais lavandarias por exemplo.

3 Benchmarking

Após uma análise do mercado, não encontramos atualmente nenhum software que tenha o mesmo funcionamento, existe sim uma aplicação que faz a gestão da parte da lavandaria em si.

Esta aplicação tem 3 funcionalidades, uma funcionalidade que dá para ver se a máquina está disponível, dá para fazer o pagamento pela aplicação (disponível somente em algumas máquinas) e por fim tem uma área de administrador que dá para ver a quantidade de lavagens. Atualmente esta aplicação é pouco usada, tem um custo elevado (3500€ + Mensalidade) e tem pouco desenvolvimento.

Na nossa solução abrange estas funcionalidades e muito mais, a mesma consegue fazer a gestão da engomadoria que atualmente é o ponto mais importante a ser resolvido.

Com a finalidade de conseguirmos provar que este projeto será viável, decidimos fazer um questionário, onde obtivemos os seguintes resultados:

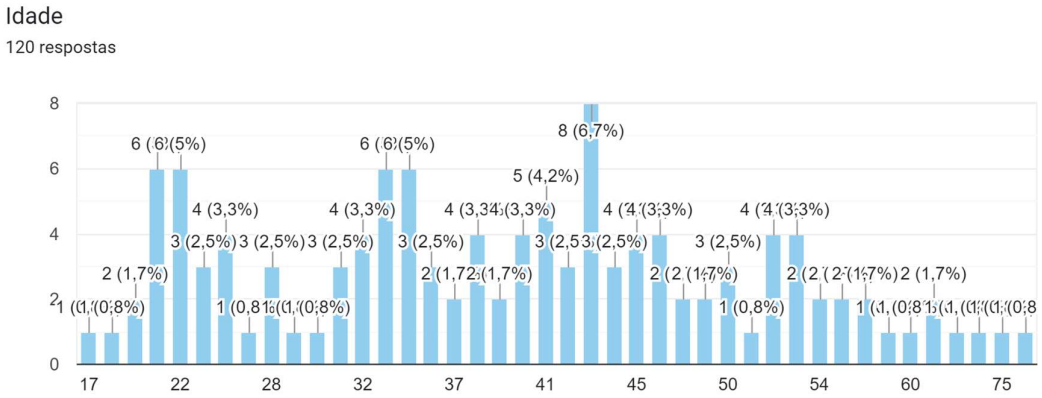


Figure 1 Questionário – Idade

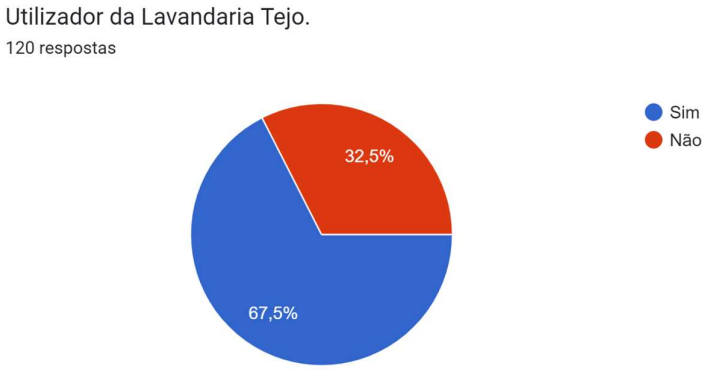


Figure 2 Questionário – Utilizador Lavandaria Tejo

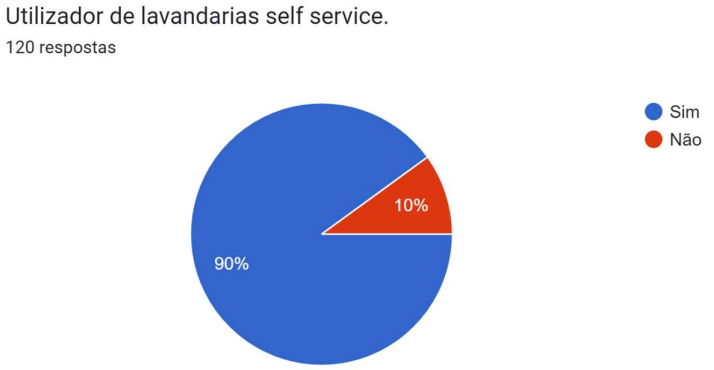


Figure 3 Questionário – Utilizador Self Service

Utilizaria um serviço Web/Aplicação onde poderia ver se as maquinas estavam disponíveis ou em utilização (Tempo para estar disponível).

120 respostas

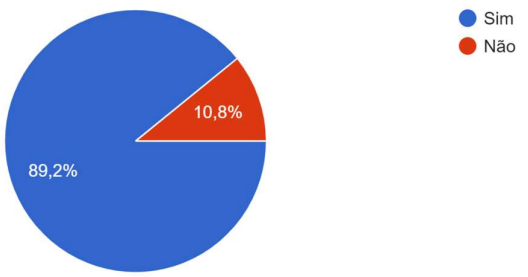


Figure 4 Questionário – Utilizaria Serviço Web

Qual seria o nível de utilização deste serviço.

120 respostas

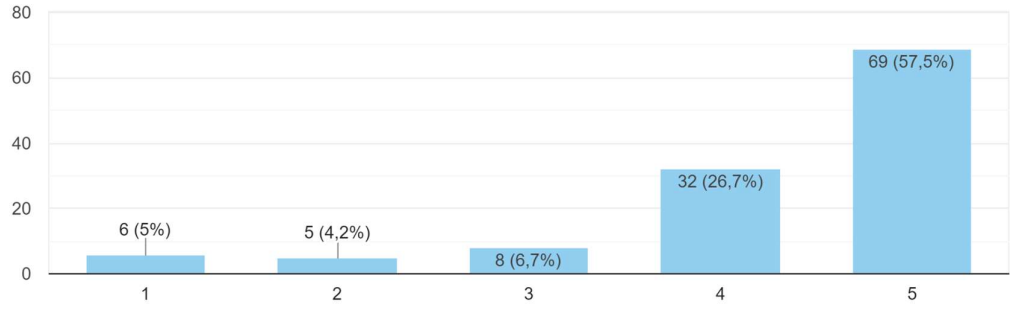


Figure 5 Questionário – Utilidade do serviço

Se é utilizador de lavandarias self service alguma vez utilizou um serviço como este.

114 respostas

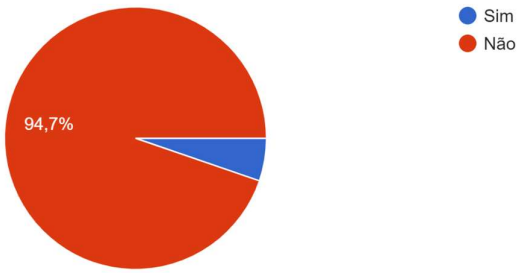


Figure 6 Questionário – Utilização de um serviço parecido

4 Engenharia

4.1. Levantamento e análise dos Requisitos

Para entendermos melhor a estrutura e as características da solução que queremos desenvolver, precisamos de fazer um levantamento de requisitos.

Trata-se do processo de compreensão e identificação das necessidades que o cliente espera ser resolvido com a aplicação que será desenvolvida.

Para cada requisito atribuímos uma prioridade, decidida através da sua relevância para o cliente:

- Essencial
    - Requisito é imperativo para o sistema entrar em funcionamento.
- Importante
    - Requisito não é imperativo para o sistema entrar em funcionamento, mas de forma não satisfatória. Deve ser implementado, mas se não for, o sistema poderá ser implementado e usado mesmo assim.
- Desejável
    - O Requisito não compromete as funcionalidades básicas do sistema. Significa que o sistema funciona de modo satisfatório sem ele. Significa também que este requisito pode ficar para versões posteriores da aplicação caso não exista tempo hábil para o implementar.

No âmbito de entrega final, atribuímos a cada requisito o seu grau de cumprimento:

- Completo
    - Requesito foi comprido e testado.
- Parcial
    - O requisito foi parcialmente feito, não foi testado.
- Não completo
    - O requisito não foi desenvolvido.

Requisitos não funcionais:

ID	Nome requisito	Descrição	Cargo	Prioridade	Cumprimento
RF01	Login	O Cliente/Funcionario/Administrador pode fazer login.	Cliente/ Funcionário/ Administrador	Essencial	Parcial
RF02	Localização das lojas	O Cliente pode pesquisar por lojas e obter também a sua localização.	Cliente	Desejável	Completo
RF03	Máquinas disponíveis por loja	O cliente pode ver quais as máquinas disponíveis por loja para iniciar uma lavagem ou uma secagem.	Cliente	Essencial	Parcial
RF04	Criação do cliente	O Funcionário/ Administrador cria o cliente.	Funcionário/ Administrador	Essencial	Parcial
RF05	Atualização da ficha de um cliente	O Funcionário/ Administrador pode atualizar uma ficha de cliente a partir do seu número de cliente.	Funcionário/ Administrador	Essencial	Parcial

RF06	Pesquisa de cliente	O Funcionário / Administrador pesquisa o cliente por diversos parâmetros como Número de Cliente, Nome, Localidade ect.. para poder ter acesso ao histórico, adicionar cartões e/ou peças.	Funcionário/ Administrador	Essencial	Completo
RF07	Atualização dos dados do cliente	O Funcionário / Administrador pode atualizar os dados do cliente após a sua pesquisa.	Funcionário/ Administrador	Importante	Não completo
RF08	Verificação de cartão válido	O Cliente por verificar se o seu cartão de cliente ainda é válido	Cliente	Desejável	Não completo
RF09	Verificação dos diferentes tipos de serviços	O cliente pode verificar os diferentes tipos de serviços para o seu tipo de cartão.	Cliente	Essencial	Não completo
RF10	Criação de encomendas	O Funcionário pode criar encomendas	Funcionário	Essencial	Não completo
RF10	Verificar as encomendas	O Funcionário pode verificar as encomendas completas, pendentes ou atrasadas.	Funcionário	Essencial	Parcial
RF1	Atualizar	O Funcionário pode	Funcionário	Essencial	Parcial

1	encomendas	atualizar uma encomenda. Seja isso, mudar o seu estado, adicionar peças ect.			
RF12	Venda de cartões	O Funcionário pode registar uma venda de um cartão.	Funcionário	Importante	Parcial
RF13	Ativar/Desativar serviço	Um Administrador consegue tirar a visibilidade de um certo serviço.	Administrador	Essencial	Parcial
RF14	Criar um funcionário	Um administrador consegue criar um funcionário novo	Administrador	Essencial	Parcial
RF15	Visualizar funcionários	Um administrador consegue ver o total de funcionários e os seus dados.	Administrador	Essencial	Parcial
RF16	Atualizar funcionário	Um administrador pode atualizar um funcionário.	Administrador	Importante	Parcial
RF17	Apagar Funcionário	Um administrador pode apagar um funcionário.	Administrador	Importante	Parcial
RF18	Adicionar despesas	Um administrador pode adicionar uma ou várias despesas.	Administrador	Essencial	Parcial
RF19	Atualizar despesas	Um administrador pode atualizar as despesas já adicionadas.	Administrador	Essencial	Parcial

RF20	Visualizar Despesas	Um administrador pode visualizar as despesas.	Administrador	Essencial	Parcial

Table 1 Requisitos funcionais

Como podemos verificar pelos requisitos funcionais, alguns não ficaram completos.

Requisitos não funcionais:

ID	Nome requisito	Descrição	Categoria	Prioridade	Cumprimento
RNF01	99.6% de uptime aplicacional	A aplicação tem de ter uptime de 99.6% tal como as máquinas de lavar	Disponibilidade	Importante	Parcial
RNF02	Criação de Passwords para utilizadores	Quando é criado um cliente, é lhe dado uma password default, essas passwords têm de ser seguras com as seguintes limitações: Mínimo 8 characters 1 Letra Maiúscula 1 Número 1 character	Segurança	Importante	Não completo

		especial (!, @, $)			
RNF 03	Facilidade de compreensão	Como queremos ter clientes de todas as idades a usar a aplicação, precisa de ser fácil de compreender até para os clientes mais idosos	Usabilidade	Desejável	Feito
RNF 04	Facilidade de aprendizagem	Queremos o tempo e esforço mínimo para alcançar um determinado nível de desempenho no uso do sistema.	Usabilidade	Desejável	Não completo
RNF 05	Protocolo SMTP	Uso do protocolo SMTP para o envio de e-mails a partir da aplicação	Implementação	Desejável	Não completo
RNF 06	API em C#	Desenvolvimento da API em C#	Implementação	Essencial	Feito
RNF 07	Front-end em React.JS	Desenvolvimento do front-end da aplicação em React.JS.	Implementação	Essencial	Parcial
RNF 08	Comunicação com BD em SQL Server	API tem de comunicar com a base de dados em SQL Server	Interoperabilidade	Essencial	Feito

RNF 09	Uso do HTTPS	Use do protocolo HTTPS na aplicação	Segurança	Importante	Parcial

Table 2 Requisitos não funcionais

4.2. Diagramas de Casos de Uso

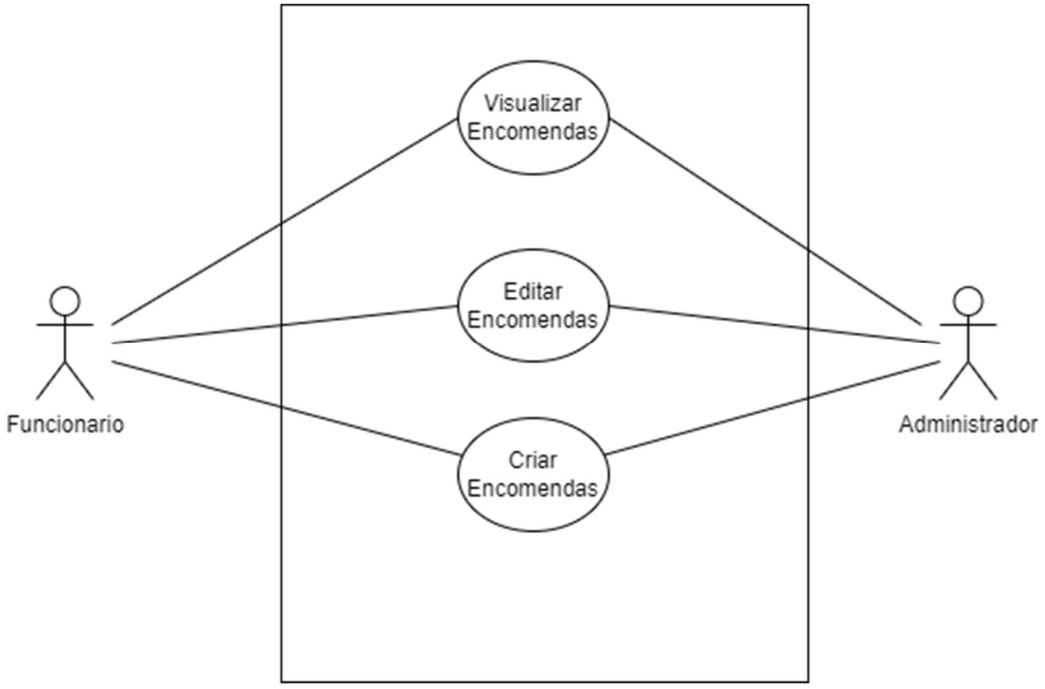


Figure 1 Visualização de Encomendas

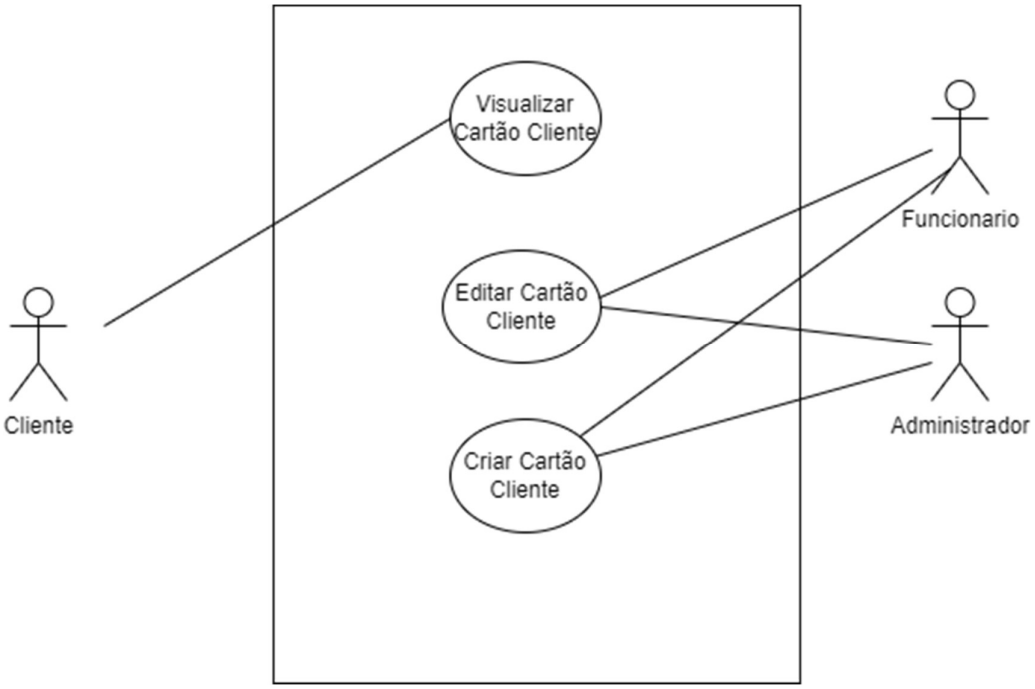


Figure 2 Funcionamento do Cartão Cliente

4.3. Diagramas de Actividades ou equivalente

4.4. Modelos relevantes

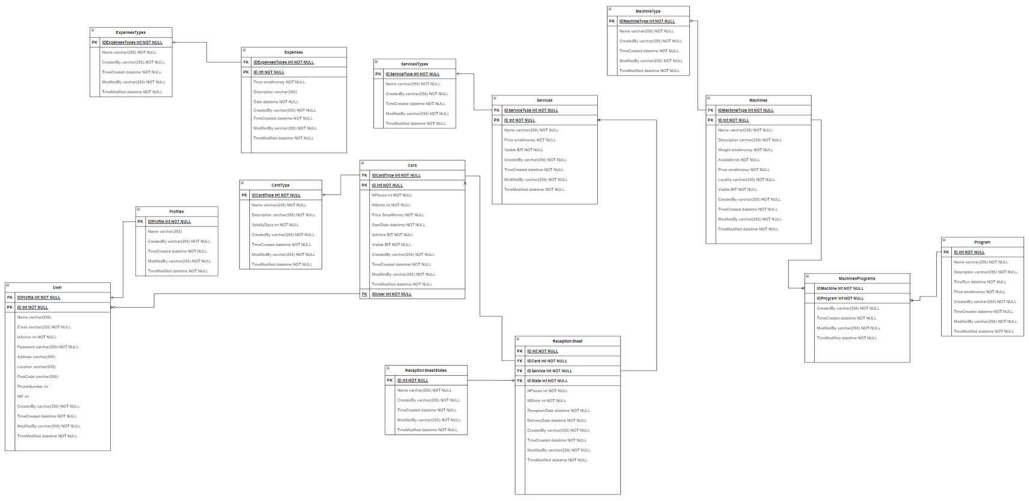


Figure 9 Modelo Entidade-Relação

4.5. Estrutura

4.6. Mockups, story boards

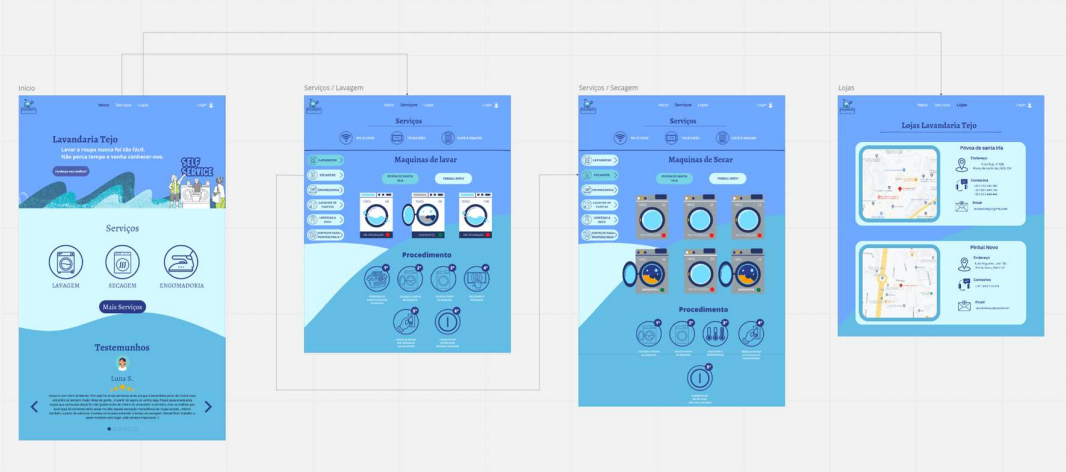


Figure 10 Story boards área de cliente

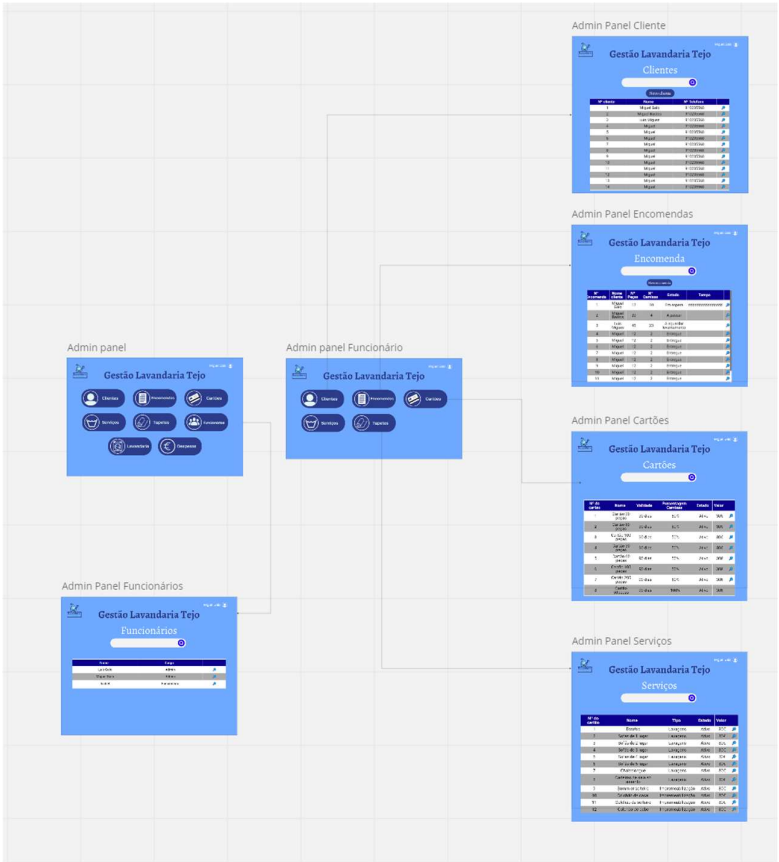


Figure 11 Story boards área administrativa

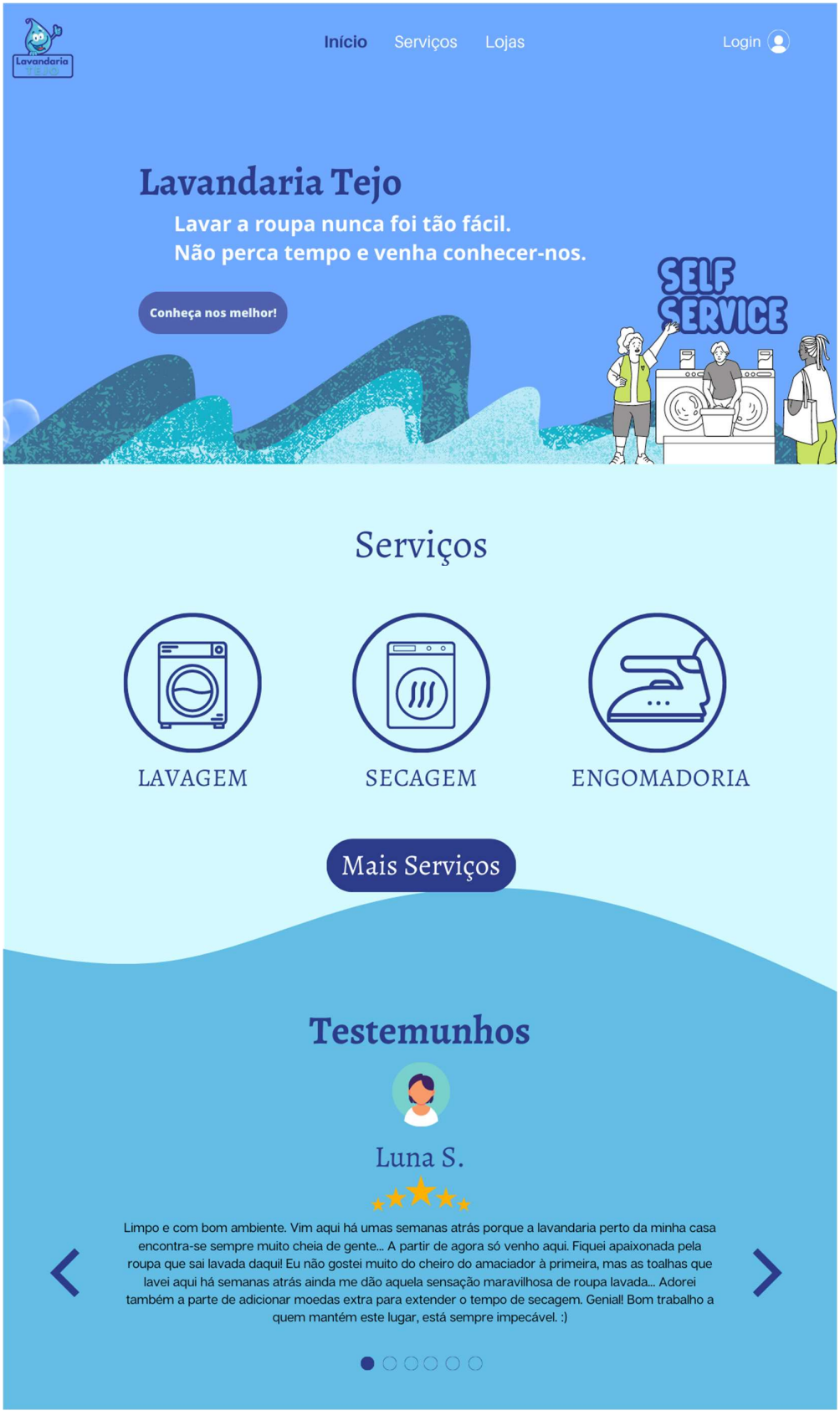


Figure 12 Mockup - Página Inicial

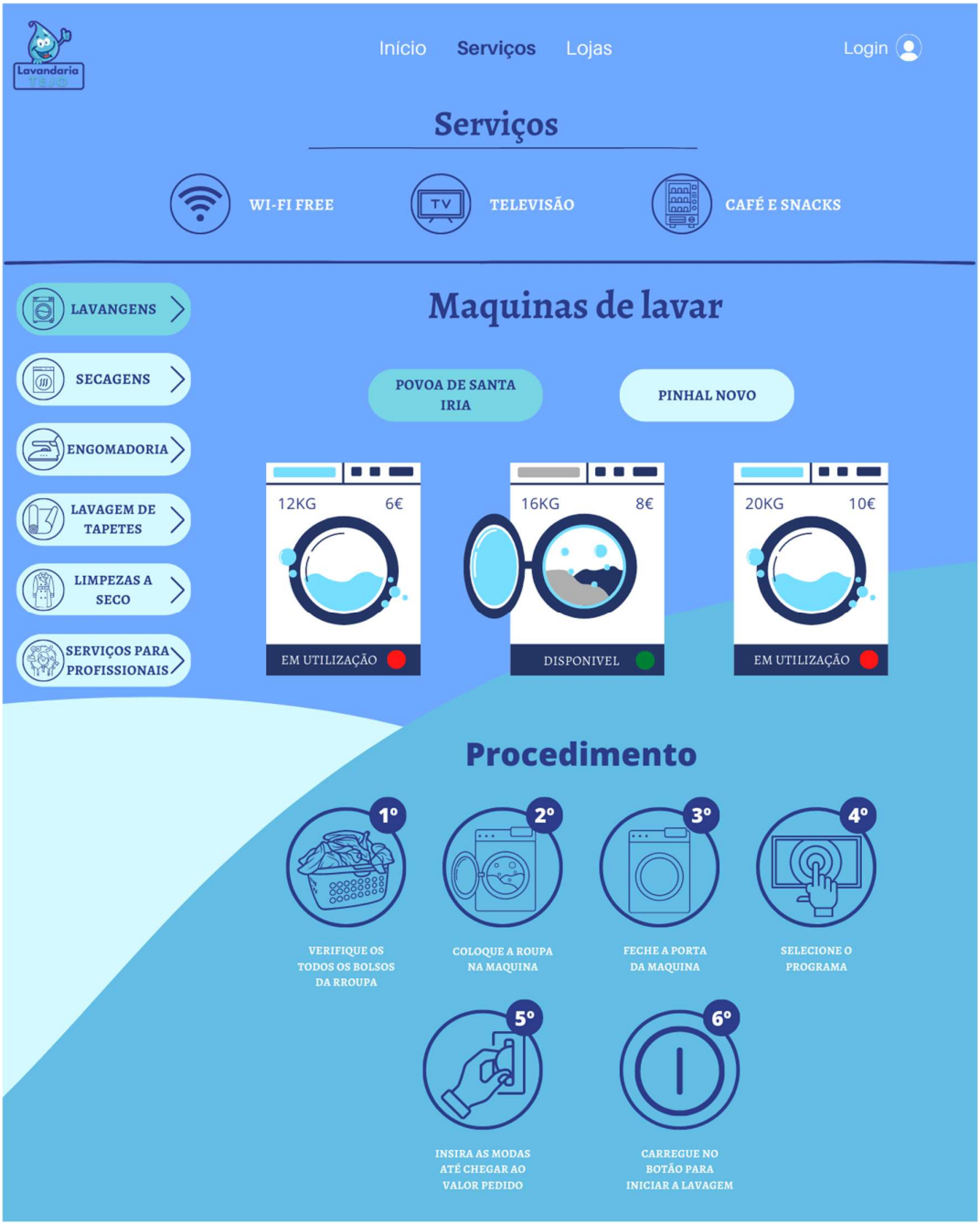


Figure 13 Mockup - Página Serviços lavagens

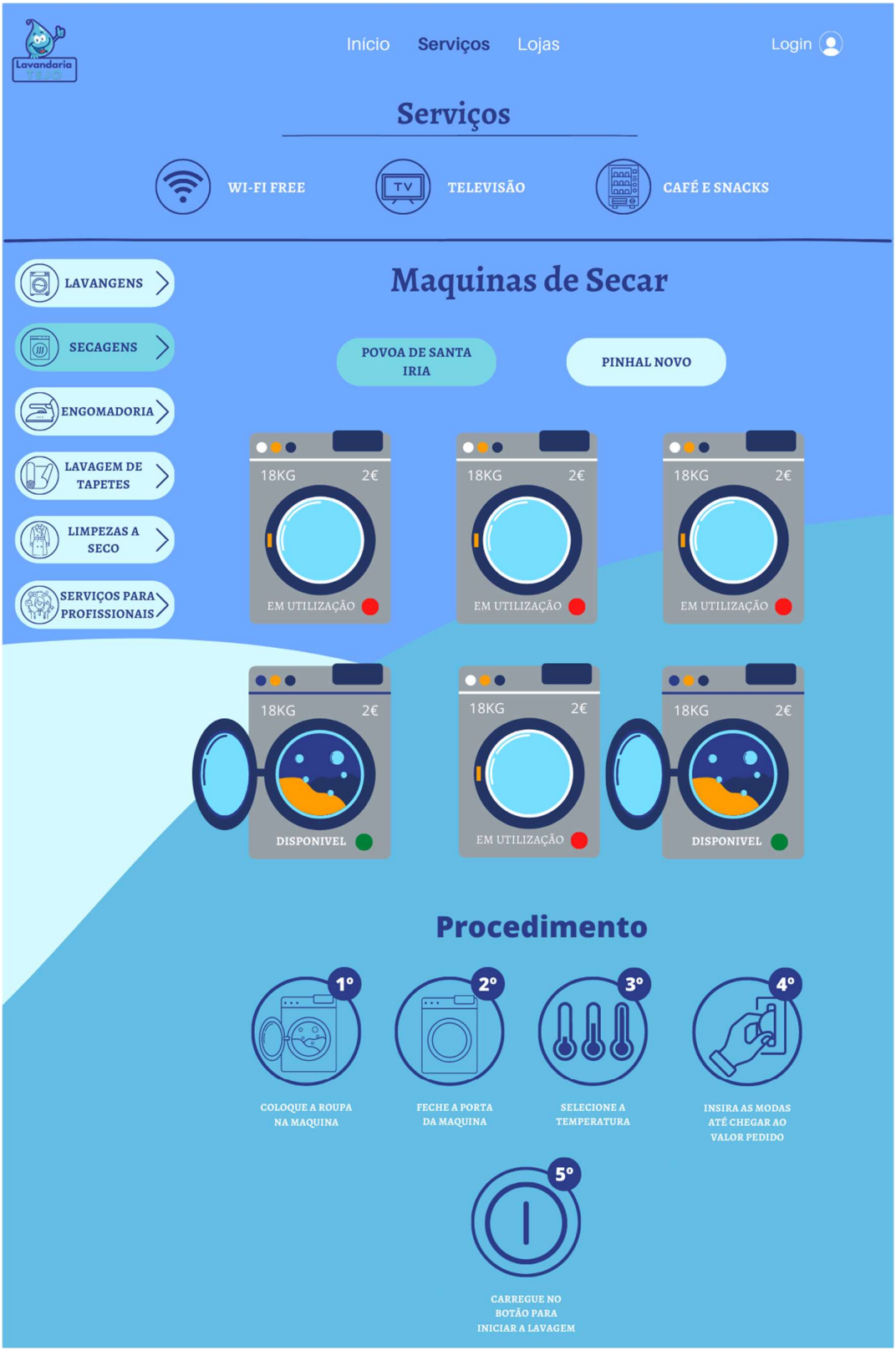


Figure 14 Mockup - Página Serviços secagem

Figure 15 Mockup - Página lojas

Figure 16 Mockup - Admin panel funcionário

Figure 17 Mockup - Admin panel

Figure 18 Mockup - Admin panel clientes

Figure 19 Mockup - Admin panel encomendas

Nº do cartão	Nome	Validade	Porcentagem Camisas	Estado	Valor	
1	Cartão 30 peças	30 dias	50%	Ativo	30€	
2	Cartão 60 peças	30 dias	50%	Ativo	30€	
3	Cartão 100 peças	90 dias	50%	Ativo	30€	
4	Cartão 30 peças	90 dias	50%	Ativo	30€	
5	Cartão 60 peças	90 dias	50%	Ativo	30€	
6	Cartão 100 peças	90 dias	50%	Ativo	30€	
7	Cartão 200 peças	90 dias	50%	Ativo	30€	
8	Cartão 50peças	90 dias	100%	Ativo	30€	

Figure 20 Mockup - Admin panel cartões

Figure 21 Mockup - Admin panel Serviços

Figure 22 Mockup - Admin panel funcionários

5 Solução Desenvolvida

5.1. Introdução

Para a criação de um software orientado à *Web,* precisamos de um backend estável, fluido e rápido. Decidimos usar o ASP .NET Web API como API com uma base de dados em SQL Server para este desafio. Para o front-end, decidimos ligar a API ao React para ser simples e rápido. Esta biblioteca ajuda os nossos UIs a serem rápidos e fáceis de desenvolver.

Link Vídeo demonstrativo:

Frontend: https://github.com/Miguel-22004314/Lavandaria_Tejo_Front_End

Backend: https://github.com/Miguel-21904220/Lavandaria_Tejo_Back_End

5.2. Arquitetura

Existem 3 componentes para a nossa arquitetura:

- A base de dados em SQL Server.
- API em .NET Web API.
- Uma aplicação usando a framework React.

A API é a peça central da aplicação. Quando recebe uma chamada HTTP a partir do React, faz uma query à base de dados para encontrar a informação desejada pelo utilizador.

Essa informação é depois enviada pela base de dados para a API, a qual faz o tratamento de dados e envia esses dados em formato JSON ou XML para o front-end em React para ser exibido.

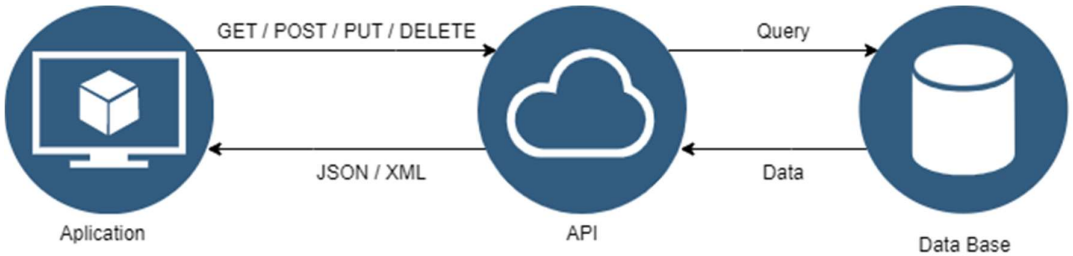


Figure 23 Legenda

5.3. Tecnologias e Ferramentas Utilizadas

Para este projeto, vamos usar .NET Web API para o back-end da aplicação Web e React para o front-end da aplicação.

Para entender o que é .NET Web API, precisamos de perceber o que é uma API. Uma API é uma software que liga uma aplicação diretamente aos dados e serviços de outro, cedendo-lhe acesso a partes específicas de um servidor.

Em termos simples, as APIs permitem que dois softwares comuniquem entre si e são a base para a maioria das aplicações modernas.

Neste caso .NET Web API é uma API do tipo REST.

O que é um REST API?

REST é na verdade, uma API de "*web services*". As APIs REST baseiam-se em URIs (Uniform Resource Identifier onde um URL é um tipo específico) e no protocolo HTTP. Para além disso, usam JSON para formatar os dados, o que é super compatível com qualquer browser.

Porque usar um REST API?

- REST foi construído para ser simples, graças aos protocolos HTTP.
- REST é conhecido pelo seu excelente desempenho e escalabilidade.
- O REST está otimizado para a Web. A utilização do JSON como o seu formato de dados torna-o compatível com basicamente todos os browsers.

O que é .NET Web API?

Web API é uma framework para construir serviços HTTP. Desta forma conseguimos expor serviços via HTTP e possibilitar uma ampla gama de dispositivos (browser, mobile ect..) a acendê-los e consumi-los.

Porque usar .NET Web API?

Hoje em dia, uma aplicação baseada na web não é suficiente para chegar aos seus clientes. As pessoas são muito inteligentes, utilizam dispositivos iphone, móveis, tablets, etc. na sua vida diária. Estes dispositivos também têm muitas aplicações para tornar a vida fácil. Na verdade, estamos a passar da web para o mundo das aplicações.

Para expor os nossos dados de serviço aos navegadores e assim como todas estas aplicações de dispositivos modernos de forma rápida e simples, devemos ter uma API compatível com os navegadores e todos estes dispositivos.

Web API é uma boa *framework* para expor os nossos dados e serviços a dispositivos diferentes. Além disso, a Web API é uma plataforma de código aberto ideal para construir serviços REST-ful sobre a estrutura .NET. Neste caso, não vamos usar a estrutura .NET para visualizar esses dados, utilizamos em seguida a framework React.js.

Algumas features do Web API:

- Apoia ações CRUD uma vez que funciona com os verbos HTTP (GET, POST, PUT, DELETE)
- Pode aceitar e gerar conteúdo que pode não ser orientado para objetos como imagens, ficheiros, PDFs. Isto será útil para o nosso projeto, poderemos gerar ficheiros excel com os dados introduzidos num certo mês ou os gastos de um certo período.

O que é React?

React JS é uma biblioteca JavaScript para a criação de interfaces de utilizador.

Porque React?

Não só queríamos algo que fosse fácil e rápido de desenvolver, mas que fosse fácil de aprender. React em comparação com outras bibliotecas como Angular ou Vue, é muito mais simples de aprender. De facto, é uma das principais razões pelas quais a React ganhou tanta tração em pouco tempo. Ajuda as empresas a construir rapidamente os seus projetos.

Um dos principais benefícios da utilização do React JS é o seu potencial para reutilizar componentes.

Podemos criar componentes para um site, mas reutilizar esses mesmos componentes numa aplicação móvel, por exemplo, o que nos abre a porta a também criar uma *App* mais tarde.

O que é JWT?

JWT é um padrão aberto utilizado para partilhar informação de segurança entre duas partes, um cliente e um servidor.

Cada JWT contém objetos JSON codificados. Os JWT são assinados utilizando um algoritmo criptográfico para assegurar que as reclamações não podem ser alteradas após a emissão do token.

JWT não foi utilizado na versão final do projeto por falta de tempo para a sua implementação.

O que é Postman?

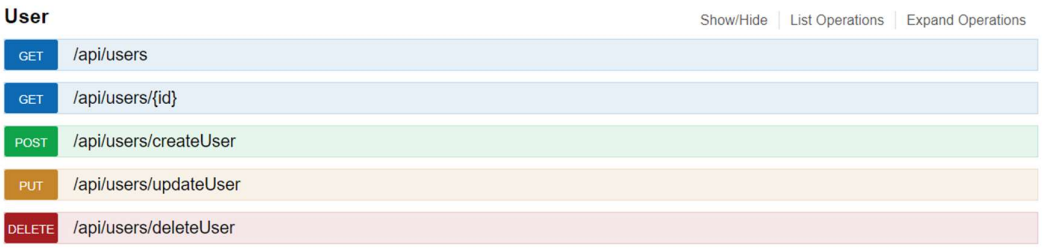
Postman é uma plataforma autônoma de teste de software API para construir, testar, projetar e documentar APIs. É uma interface gráfica de utilizar para enviar e visualizar pedidos e respostas HTTP.

Esta ferramenta tem a capacidade de fazer vários tipos de pedidos HTTP como GET, POST, PUT, PATCH, e converter a API para código para linguagens como JavaScript e Python. O que é extremamente importante para o nosso projeto, para testar pedidos à base de dados e podermos construir testes "unitários" em caso de mudanças no código.

O que é Swagger?

Swagger é muito parecido com Postman, é uma aplicação open source que nos auxilia para construir, testar, projetar, documentar APIs. A diferença entre os dois é que o Swagger está implementado diretamente na nossa API, o que nos deixa correr o projeto e abrir diretamente o swagger onde podemos encontrar respostas a pedidos GET, fazer pedidos POST, PUT e DELETE.

Na imagem abaixo vemos um exemplo do Swagger a trabalhar com a nossa API para mostrar os pedidos desenvolvidos.



GET - /api/users -> faz um pedido de todos os utilizadores.

Response Body


```
[
  {
    "ID": 3,
    "IDProfile": 3,
    "Name": "Ana Silva",
    "Email": ,
    "IsActive": 1,
    "Password": "minhapassword123",
    "Address": "Rua das Flores 123",
    "Location": "Lisboa, Portugal",
    "PostCode": "1000-001",
    "PhoneNumber": 123456789,
    "NIF": 123456789,
    "CreatedBy": "1",
    "TimeCreated": "2023-09-08T10:00:00",
    "ModifiedBy": "1",
    "TimeModified": "2023-09-08T10:00:00"
  },
  {
    "ID": 4
```


GET - /api/users/{id} -> faz um pedido para receber um utilizador com um ID específico. (Exemplo com ID = 5)

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	5		path	integer

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'https://localhost:44386/api/users/5'
```

Request URL

```
https://localhost:44386/api/users/5
```

Response Body


```
{
  "ID": 5,
  "IDProfile": 3,
  "Name": "Marta Fernandes",
  "Email": "marta.fernandes@example.com",
  "IsActive": 1,
  "Password": ,
  "Address": "Rua da Praia 101",
  "Location": "Faro, Portugal",
  "PostCode": "8000-003",
  "PhoneNumber": 567890123,
  "NIF": 567890123,
  "CreatedBy": "1",
  "TimeCreated": "2023-09-08T12:00:00",
  "ModifiedBy": "1",
  "TimeModified": "2023-09-08T12:00:00"
}
```

Estes são dois exemplos de como o Swagger funcionou para nos ajudar no desenvolvimento da API.

5.4. Implementação

Para o ambiente produtivo, continuamos em debate sobre a melhor solução. A primeira ideia do ambiente seria meter no Azure, o que oferece rapidez, segurança e facilidade na escalabilidade.

Apesar de ser uma plataforma economicamente eficiente, os preços não condizem com a nossa realidade pessoal.

5.5. Abrangência

Existem várias disciplinas que vamos aplicar na solução proposta como:

- Fundamentos da Programação
    - Com esta disciplina, foi nos fornecida as bases para que possamos iniciar a atividade de programação.
- Linguagens de Programação 1
    - O aprofundamento dos conhecimentos sobre a programação é feito nesta unidade curricular. Aprendemos um conjunto de conhecimentos gerais sobre os diversos paradigmas da programação o que nos ajuda a ter uma mente aberta para outras possíveis soluções para os problemas da nossa aplicação.
- Linguagens de Programação 2
    - O objetivo desta disciplina é apresentar os conceitos fundamentos da programação orientada a objetos para que possamos aplicá-los em projetos de desenvolvimento de software usando linguagens de programação modernas como C#, o qual a nossa API é desenvolvida.

- Programação Web
    - O objetivo desta unidade curricular é introduzir o aluno a um conjunto de linguagens e tecnologias web. Isto será útil para o nosso desenvolvimento, já que nos ajudou a perceber a arquitetura de uma aplicação web de grande escala, a importância da segurança nas aplicações web e como uma aplicação interage com os vários níveis da arquitetura.
- Base de Dados
    - A cadeira mais importante para o desenho e criação do nosso modelo de dados. Aprendemos a criar e alterar uma estrutura de dados, manipular dados, desenvolver e implementar uma política de segurança com base no SQL.

6 Método e Planeamento

O planeamento do desenvolvimento da segunda parte do projeto:

Name	Start Date	End Date	Duration	Dependency
Desenvolvimento do relatório	Oct 20, 2022	Nov 25, 2022	27 days	
Desenvolvimento relatório, Parte 2	Nov 28, 2022	Jan 27, 2023	45 days	1FS
Desenvolvimento relatório, Parte 3	Jan 30, 2023	Apr 21, 2023	60 days	4FS
Desenvolvimento relatório, Parte 4	Apr 24, 2023	Sep 08, 2023	100 days	13FS
Criação do Questionário	Oct 31, 2022	Nov 01, 2022	2 days	
Recolha de dados	Nov 03, 2022	Apr 03, 2023	108 days	3FS+1 day
Criação das mockups	Nov 28, 2022	Feb 06, 2023	51 days	
Criação da DB	Jan 13, 2023	Jan 18, 2023	4 days	
Desenvolvimento Front End	Jan 19, 2023	Sep 08, 2023	167 days	6FS
Desenvolvimento API	Jan 19, 2023	Sep 08, 2023	167 days	6FS
Testes do Software	Jul 17, 2023	Sep 08, 2023	40 days	8SS+127 days,7...
Instalação no cliente	Sep 11, 2023	Sep 12, 2023	2 days	9FS
Feedback do cliente	Sep 12, 2023	Sep 29, 2023	14 days	10SS+1 day
Alterações / Correção de erros	Sep 12, 2023	Sep 29, 2023	14 days	11SS

Figure 24 Calendário - Dados e tempos

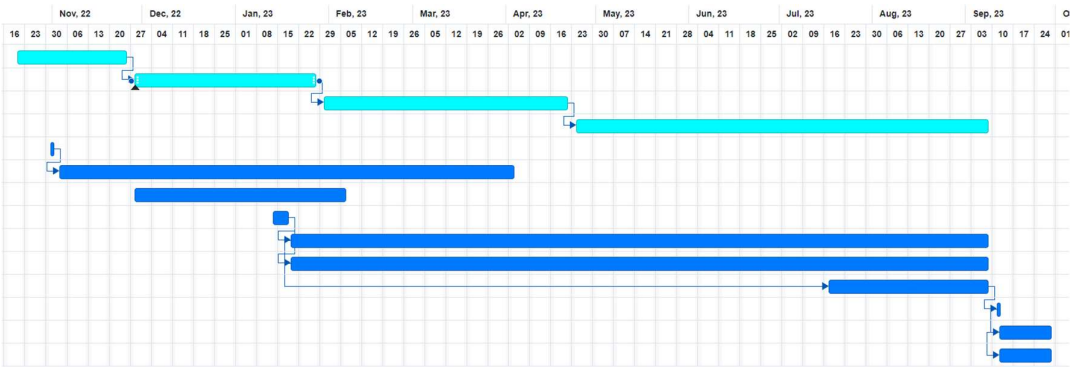


Figure 25 Calendário - Visão geral

Tendo como referência o calendário desenvolvido para a segunda parte do projeto, de um modo geral, estamos satisfeitos com o progresso do projeto.

Não conseguimos seguir o planeamento do projeto tal como foi desenvolvido na altura devido a vários problemas na nossa vida pessoal. No entanto, conseguimos desenvolver a maior parte do front-end e a API dentro do tempo estipulado.

Como o projeto não se encontra terminado, não foi possível fazer a instalação no cliente, no entanto, como cliente é um dos developers do projeto, achamos que é aceitável o atraso.

Em geral, o planeamento foi bem pensado e dividido para o tamanho do projeto e ainda deixamos alguma margem para problemas no desenvolvimento.

7 Resultados

Foram cumpridos alguns dos requisitos previstos na tabela 1 e tabela 2. A aplicação está a funcionar, mas ainda não está completa.

Requisitos Funcionais:

Requisitos	Estado
O Cliente/Funcionario/Administrador pode fazer login.	Parcial
O Cliente pode pesquisar por lojas e obter também a sua localização.	Completo
O cliente pode ver quais as máquinas disponíveis por loja para iniciar uma lavagem ou uma secagem.	Parcial
O Funcionário/ Administrador cria o cliente.	Parcial
O Funcionário/ Administrador pode atualizar uma ficha de cliente a partir do seu número de cliente.	Parcial
O Funcionário / Administrador pesquisa o cliente por diversos parâmetros como Número de Cliente, Nome, Localidade ect.. para poder ter acesso ao histórico, adicionar cartões e/ou peças.	Completo
O Funcionário / Administrador pode	Não completo

atualizar os dados do cliente após a sua pesquisa.	
O Cliente por verificar se o seu cartão de cliente ainda é válido	Não completo
O cliente pode verificar os diferentes tipos de serviços para o seu tipo de cartão.	Não completo
O Funcionário pode criar encomendas	Não completo
O Funcionário pode verificar as encomendas completas, pendentes ou atrasadas.	Parcial
O Funcionário pode atualizar uma encomenda. Seja isso, mudar o seu estado, adicionar peças ect.	Parcial
O Funcionário pode registar uma venda de um cartão.	Parcial
Um Administrador consegue tirar a visibilidade de um certo serviço.	Parcial
Um administrador consegue criar um funcionário novo	Parcial
Um administrador consegue ver o total de funcionários e os seus dados.	Parcial
Um administrador pode atualizar um funcionário.	Parcial
Um administrador pode apagar um funcionário.	Parcial

Um administrador pode adicionar uma ou várias despesas.	Parcial
Um administrador pode atualizar as despesas já adicionadas.	Parcial
Um administrador pode visualizar as despesas.	Parcial

Table 3 - Resultado requisitos funcionais

Requisitos Não Funcionais:

Requisitos	Estado
A aplicação tem de ter uptime de 99.6% tal como as máquinas de lavar	Parcial
Quando é criado um cliente, é lhe dado uma password default, essas passwords têm de ser seguras com as seguintes limitações: Mínimo 8 characters 1 Letra Maiúscula 1 Número 1 character especial (!, @, $)	Não completo
Como queremos ter clientes de todas as idades a usar a aplicação, precisa de ser fácil de compreender até para os clientes mais idosos	Feito

Queremos o tempo e esforço mínimo para alcançar um determinado nível de desempenho no uso do sistema.	Não completo
Uso do protocolo SMTP para o envio de e-mails a partir da aplicação	Não completo
Desenvolvimento da API em C#	Feito
Desenvolvimento do front-end da aplicação em React.JS.	Parcial
API tem de comunicar com a base de dados em SQL Server	Feito
Use do protocolo HTTPS na aplicação	Parcial

Table 4 - Resultado requisitos não funcionais

8 Conclusão e trabalhos futuros

Este projeto não se encontra completo apesar de a aplicação funcionar, um dos grandes motivos de não termos conseguido a sua conclusão foi a decisão de usar React.JS para o nosso front-end. Ambos os elementos do grupo nunca tinham tido qualquer contacto ou experiência em JavaScript. Com isto, houve inúmeras dificuldades em criar o front-end para o projeto, não só pela nossa falta de tempo, mas também pela falta de experiência.

A proposta de desenvolver uma aplicação web em React.js com o suporte de uma API em .NET Web API e uma base de dados em SQL Server parece ser uma abordagem sólida para resolver os problemas existentes na gestão de informações, mas a escolha de usar React.Js para o nosso front-end impactou imenso a nossa habilidade de completar o projeto.

A transição efetuada de uma base de dados em papel para um sistema digital permite uma organização mais eficiente dos dados, reduzindo a possibilidade de erros e melhorando a acessibilidade às informações críticas.

Existem ainda alguns trabalhos futuros a fim de a aplicação estar 100% funcional, para que a utilização da aplicação web proporcione uma maior flexibilidade e acessibilidade aos funcionários, tornando a gestão de tarefas mais simples e eficaz.

Futuramente se a implementação for bem sucedida na Lavandaria Tejo gostaríamos de vender o projeto a outras lavandarias para auxiliar a sua gestão, flexibilidade e acessibilidade, uma vez que achamos que o projeto tem potencial para ocupar um espaço no mercado Português, tal como descrito no capítulo 2.

Bibliografia

[DEISI 22] DEISI, Regulamento de Trabalho Final de Curso, Set. 2022.

[ULHT 23] Universidade Lusófona de Humanidades e Tecnologia, www.ulusofona.pt, acedido em jan. 2023.

[DEISI 23] Lusófona Informática , https://informatica.ulusofona.pt, acedido em jan. 2023.

[Wikipedia 23] Wikipedia , https://pt.wikipedia.org/wiki/Requisito_n%C3%A3o_funcional, acedido em jan.2023.

[Wikipedia 23] Wikipedia , https://pt.wikipedia.org/wiki/Microsoft_SQL_Server , acedido em jan.2023.

[Miro 23] Miro, www.miro.com, acedido Jan. 2023.

[Microsoft 23] Microsoft, https://dotnet.microsoft.com/en-us/apps/aspnet/apis, acedido em jan. 2023.

[React.JS 23] ReactJs, https://reactjs.org/, acedido set. 2023.

[Canva 23] Canva, https://www.canva.com/, acedido Jan. 2023.

[Diagrams 23] Diagrams, https://app.diagrams.net, acedido Jan. 2023.

[Chat GPT] https://chat.openai.com/, acedido set. 2023.

[React.JS] https://react.dev/, acedido set 2023.

Anexos 1 – Mockups, Modelos

Miro Mockups -

https://miro.com/welcomeonboard/c0xwckNSV3c0VlQ4YkNOSDBhSzRaa2R2dzVkclZDamhXaTNCVlFtMUNuakN0UVhnSG5udzVKWXBMUndCc3RZR3wzMDc0NDU3MzY2Njc3NTI3NDk3fDI=?share_link_id=624489619951

Diagrams Modelo Entidade-Relação –

https://drive.google.com/file/d/13-n60rFg_N4dLeafHnh8jNxlVQRRkDVv/view?usp=sharing

Anexos 2 - Progresso de trabalho

Em termos de progresso do trabalho, subestimamos a quantidade de tempo que precisávamos para criar os mockups de todas as páginas. A aplicação em tamanho foi maior do que pensávamos, daí necessitamos de mais tempo para criar os mockups. Em contraste, criamos a base de dados em apenas 2 dias e utilizamos mais 2 dias para a normalizar.

Aumentamos substancialmente o desenvolvimento do API e do front-end. Achamos que estas duas partes são as mais importantes do projeto e que precisamos de dedicar mais tempo para estas duas tarefas.

Queremos também começar os testes antes de estar completa a aplicação, por exemplo, queremos poder testar a área do cliente enquanto desenvolvemos a área do administrador e do funcionário.

Adicionamos o feedback do cliente e alterações pequenas ao nosso planeamento. Após recebermos o feedback, decidimos que precisamos de tempo para fazer essas alterações dependendo da sua prioridade.

Name	Start Date	End Date	Duration	Dependency
Desenvolvimento do relatório	Oct 20, 2022	Nov 25, 2022	27 days	
Criação do Questionário	Oct 31, 2022	Oct 31, 2022	1 day	
Recolha de dados	Nov 07, 2022	Nov 23, 2022	13 days	3FS+4 days
Desenvolvimento relatório, Parte 2	Nov 28, 2022	Jan 27, 2023	45 days	1FS
Criação das mockups	Nov 28, 2022	Dec 05, 2022	6 days	
Criação da DB	Dec 05, 2022	Dec 22, 2022	14 days	
Desenvolvimento Back End	Dec 07, 2022	Feb 24, 2023	58 days	
Desenvolvimento Front End	Dec 26, 2022	Mar 03, 2023	50 days	
Testes do Software	Mar 06, 2023	Apr 03, 2023	21 days	6FS,5FS,4FS
Instalação no cliente	Apr 03, 2023	Apr 14, 2023	10 days	9FS-1 days
Desenvolvimento relatório, Parte 3	Jan 27, 2023	Apr 21, 2023	61 days	
Desenvolvimento relatório, Parte 4	Apr 24, 2023	Jun 30, 2023	50 days	

Figure 26 Calendário - Dados e tempos - Versão inicial

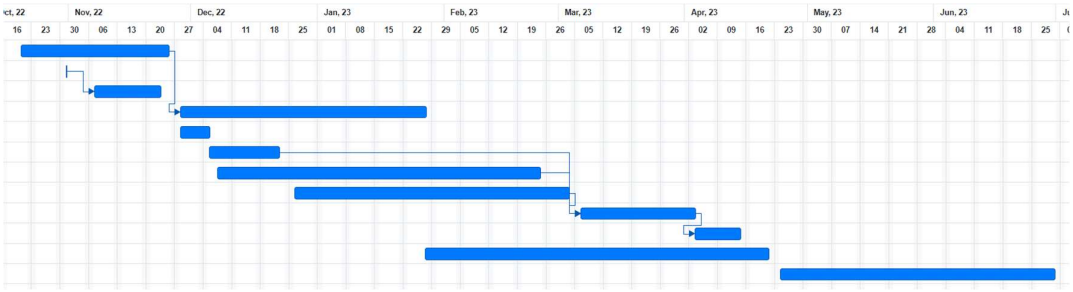


Figure 27 Calendário - Visão geral - Versão inicial

Glossário

LEI Licenciatura em Engenharia Informática

LIG Licenciatura em Informática de Gestão

TFC Trabalho Final de Curso

UI User Interface

REST Representational State Transfer

BD Base de Dados

CRUD Create, Read, Update, Delete

API Application Programming Interface

SQL Structured Query Language

HTTP Hyper Text Transfer Protocol

JWT JSON Web Token

RF Requisito Funcional

RNF Requisito Não Funcional

HTTPS Hyper Text Transfer Protocol Secure