



Licenciatura em Engenharia Informática

Desenvolvimento de Aplicações Web

Relatório do projeto – Gestão Empréstimos de livros

Paulo Luis

N.º 17359

Ano Letivo: 2020/2021 / 1º semestre / 3º Ano

Índice

Índice figuras.....	3
1. Introdução	4
2. Análise do problema.....	5
2.1 Caracterização dos Atores	5
2.2 Caracterização das Tarefas.....	5
2.3 Casos de uso (diagrama);	6
2.3.1 Especificações dos casos de usos	7
2.4 Requisitos Não Funcionais.....	13
3. Desenho do Sistema	13
3.1 Modelação de Interfaces.....	13
3.2 Storyboard de Navegação.....	18
3.2 Modelação das duas bases de dados.....	22
3.2.1 Diagrama ER.....	22
3.2.2 Modelo Físico.....	23
4. Definição da arquitetura do sistema na integração entre a aplicação e a API.....	24
4.1 API	24
4.1.1 endpoints da API.....	24
5. Implementação do sistema	26
5.1 Implementação API.....	26
5.2 Teste da API.....	28
5.3 Aplicação.....	28
5.4 Implementação dos processos críticos	29
Código para adicionar Livro no carrinho	30
Classe Funcionário	31
6. Testes de usabilidade.....	32
6.1 Questionário e o Resultado do teste	33
6.2 Resultado da avaliação	33
6.3 Métrica do teste de usabilidade	34
7. Conclusão	35

Índice figuras

Figura 1 Diagrama de caso de uso	7
Figura 2 wireframes históricos de empréstimos (funcionários)	13
Figura 3 wireframe login (geral)	14
Figura 4 wireframe Lista de funcionários	14
Figura 5 wireframe página estado da encomenda	15
Figura 6 wireframe lista de livros (funcionários)	16
Figura 7 wireframe criar livro	16
Figura 8 wireframe lista livros funcionários	17
Figura 9 Storyboard Login e Registo	18
Figura 10 Storyboard Livro e criar livro	18
Figura 11 Storyboard Aprovar empréstimo	19
Figura 12 Storyboard recusar empréstimos	19
Figura 13 Storyboard Multar	20
Figura 14 storyboard estado encomenda utente	20
Figura 15 storyboard adicionar livro ao cesto e confirmar encomenda	21
Figura 16 storyboard gestão de Autores	21
Figura 17 Diagrama E/R	22
Figura 18 Modelo Físico Base de dados - SQLServer	23
Figura 19 Diagrama de bloco API	24
Figura 20 endpoint api	24
Figura 21 endpoint empréstimos	25
Figura 22 endpoint utentes	25
Figura 23 endpoint funcionários	25
Figura 24 Solution Explorer API	26
Figura 25 código autorização API	27
Figura 26 controller Empréstimos	27
Figura 27 Resultado Livros – Postman	28
Figura 28 Solution explorer aplicação	28
Figura 29 classe Cart	29
Figura 30 implementação caso crítico - Adicionar livro ao cesto	30
Figura 31 implementação do processo crítico - remover item do cesto	31
Figura 32 implementação da classe funcionário	31
Figura 33 Resultado 1	33
Figura 34 Resultado 2	34
Figura 35 Métricas obtidas através dos testes realizados	34

1. Introdução

Para a realização do mini-projeto II solicitado pelo docente desta unidade curricular, foi proposto o desenvolvimento de duas aplicações que se interligam entre si, *uma Web API (Application Programming Interface) REST (Representational State Transfer)* e uma aplicação *Web ASPNET MVC (Model-View-Controller)* em *C Sharp*. Foram dados alguns possíveis temas de projeto em que as aplicações se deviam basear, tais como: desporto, comercio eletrónico, lazer, transportes, eventos, educação, etc.

O tema escolhido foi a realização de um local web de Empréstimos de Livros, por outras palavras, um sistema de requisição de livros. Sendo desenvolvida uma aplicação de gestão de empréstimos de livros disponibilizadas por uma biblioteca e posteriores requisitar livros ou revistas por parte dos utentes. Logo os casos de uso que foram definidos para completar este projeto foram os seguintes:

1. Consultar histórico de pedidos de empréstimos (para o utente e para funcionário);
2. Aprovar requisições;
3. Recusar requisições
4. Consultar o estado da requisição;
5. Ver a Lista de utente;
6. Adicionar livro;
7. Editar livro;
8. Apagar livro;
9. Aprovar
10. Ver a Lista de funcionários;
11. Inserir Novo funcionário;
12. Apagar Funcionário;
13. Ver a Lista de utente;
14. Inserir Novo utentes;
15. Apagar utentes;

Este projeto encontra-se dividido em cinco partes, onde a primeira foi a parte do registo da proposta de projeto. As restantes partes encontram-se documentadas neste relatório, sendo que cada capítulo retrata cada uma das partes restantes.

O Capítulo 2 – Análise do Problema, retrata a fase de análise do problema, onde são caracterizadas todas as tarefas e atores, apresentados os casos de uso e as suas respetivas especificações associadas, e por fim, os requisitos não funcionais do sistema.

O Capítulo 3 - Desenho da Solução, representa o desenho da solução, onde se encontra especificada toda a informação sobre os protótipos não funcionais das interfaces web, os modelos entidade/relação e físico das duas bases de dados, a definição da arquitetura do sistema na integração entre a aplicação e a API e a especificação da interface da API.

O Capítulo 4 - Implementação do Sistema, representa o processo seguido para a realização de ambas as aplicações.

O Capítulo 5 - Conclusão, fala sobre as reflexões sobre o trabalho realizado.

2. Análise do problema

Nesta secção mostra-se a análise do sistema e os principais elementos que a constituem.

2.1 Caracterização dos Atores

Os utilizadores deste sistema serão os potenciais clientes e um ou mais administradores. Os potenciais clientes poderão realizar tarefas como registarem-se, criando uma conta de utilizador; consultar os produtos disponíveis; efetuar requisição; contactar a equipa de suporte com dúvidas ou outros assuntos.

Ator utente

- **Utente (Cliente Final):** Clientes da biblioteca com 13 ou mais anos que tenham no mínimo conhecimentos de tecnologias e aplicações simples na _ótica do utilizador e que tenham acesso a um dispositivo com internet;
- **Administrador:** Funcionários da biblioteca com idades variadas que tenham no mínimo conhecimentos de tecnologias e aplicações simples na ótica do utilizador e têm como escolaridade mínima o Ensino Secundário.

2.2 Caracterização das Tarefas

1. Tarefa 1: Pesquisar livro (Cliente)

O utente Carlos pretende ver os livros disponíveis na biblioteca de modos, utilizar o sistema. O utente abre então o browser e acede ao sítio web de empréstimos., segui os seguintes passo:

1. Selecciona o botão “Registar”, insere os seus dados;
2. No menu, clica sobre opção Livros;
3. Já na página dos livros é disponibilizado os livros, disponíveis na biblioteca.

2. Tarefa 2: Empréstar Livros (Cliente)

A Mária com sessão iniciada que pretende requisitar um livro. Abre o browser e inserir as credencias de e inicia sessão. Já na página inicial da sua conta, a Maria segue para o mapa de livros e clica sobre opção de adiciona ao carinho dos livros que pretende. Na barra de navegação clica sobre o botão cesto e é disponibilizado os itens que seleccionou, de seguida confirma o pedido. Deve aguardar os livros serem enviado á sua casa.

3. Tarefa 4: ver estado do empréstimo (Cliente)

A Mária quer ver o estado do empréstimo que efetuou. Abre o browser e inserir as credencias e inicia a sessão. Na barra de tarefa, segui para o botão historio, é direcionado para lista de empréstimos solicitado por ele. Clica sobre o botão detalhe do empréstimo na linha correta que pretende ver as informações. É apresentado toda informação da requisição.

4. Tarefa 4: Aprovar/recusar empréstimos (Funcionário)

O Paulo é funcionário da biblioteca *MyBooks* e pretende aprovar ou recusar um empréstimo feito pelo cliente (utente).

O Paulo abre o browser e acedo ao sítio web da biblioteca, insere as suas credenciais e inicia sessão. Já na página inicial da sua conta, Paulo segue para a área dos Empréstimos, onde tem uma lista dos empréstimos existentes, e selecciona a opção de aprovar/recusar uma requisição do cliente na linha que pretende, é apresentado um *popup* de confirmação.

5. Tarefa 4: Gestão livros (Funcionário)

O Paulo é funcionário da biblioteca *MyBooks* e pretende efetuar algumas operações de criar, de editar pesquisar e eliminar livro á ser disponibilizado aos utentes. O Paulo abre o browser e acedo ao sítio web da biblioteca, insere as suas credenciais e inicia sessão. Já na página inicial da sua conta, Paulo segue para a área de Livros, onde tem uma lista dos livros que estão em stock.

Para criar livro no site, clica sobre botão novo. Insere informações do livro e do autor, clica sobre o botão guardar.

Para editar informação do livro no site, clica sobre botão editar na linha do livro que pretende. Insere informações nova do livro e do autor, clica sobre o botão atualizar.

Para editar informação do livro no site, clica sobre botão editar na linha do livro que pretende. Insere informações nova do livro e do autor, clica sobre o botão atualizar.

2.3 Casos de uso (diagrama);

Os casos de uso que foram em posteriormente identificados de forma a satisfazer todas as tarefas de forma completa estão apresentados no diagrama de casos de uso na Figura 1 Diagrama de caso de uso, onde se pode ver os dois tipos de utilizadores identificados e o que cada um pode fazer, estando todos os casos de uso pendentes de login credenciado e registado na base de dados da aplicação.

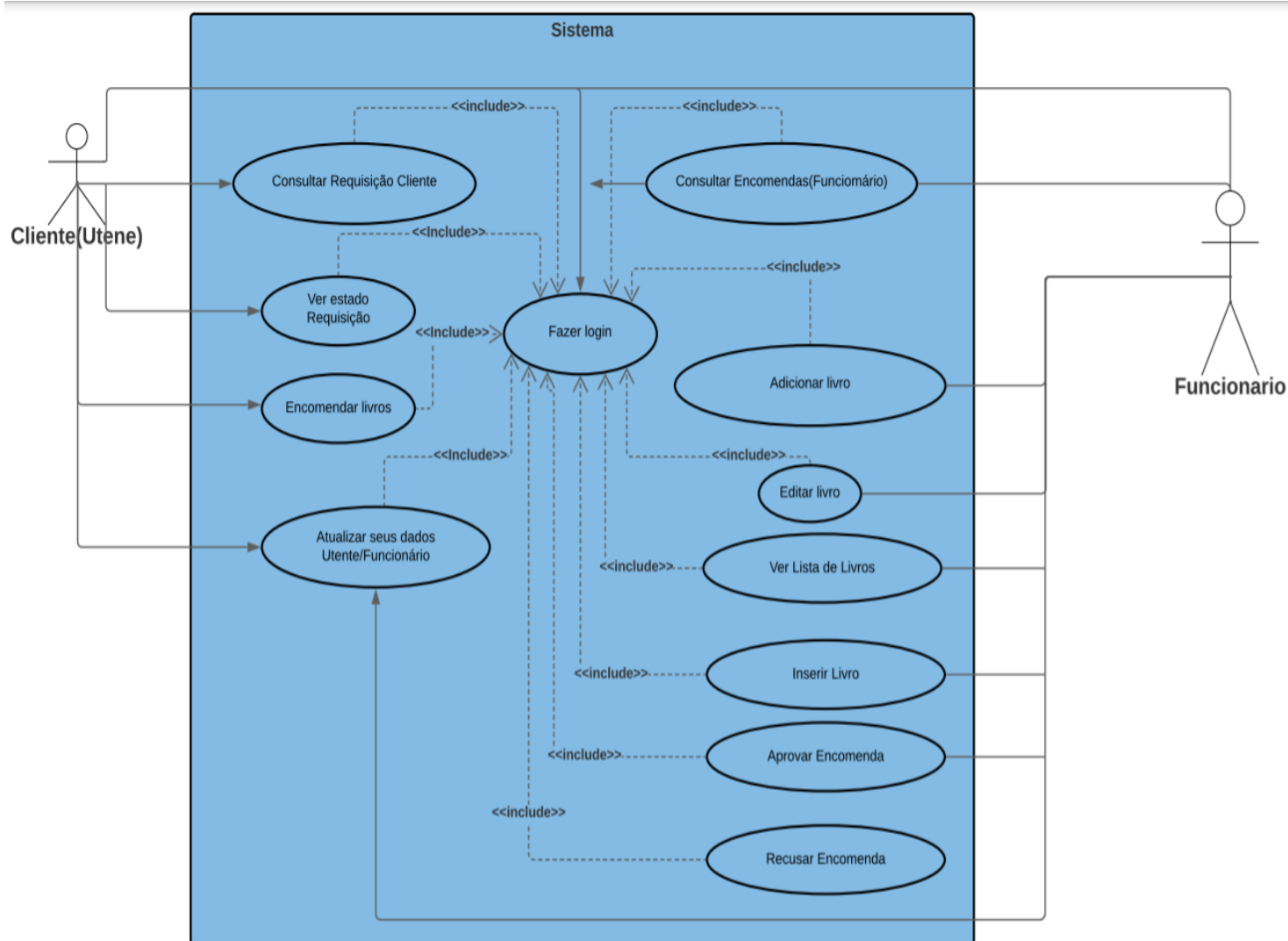


Figura 1 Diagrama de caso de uso

2.3.1 Especificações dos casos de usos

As especificações dos casos de uso estão separadas em três parte, o caso de uso comum e os afetos a cada ator.

- **Geral**

- **Fazer Login:**

- **Pré-condições:** O utilizador estar registado no sítio web.
 - **Pós-Condições:** Login efetuado com sucesso.
 - **Cenário Principal:**
 1. Utilizador insere as credenciais;
 2. O Sistema verifica autenticidade destes dados [Alternativa A: Dados incorretos]
 3. O sistema informa o Utilizador do login com sucesso.
 - **Alternativa A:** Dados incorretos: Sistema Informa Utilizador do login falhado.

- **Funcionário**

➤ **Consultar Encomendas (Empréstimos):**

- **Pré-condições:** Login efetuado.
- **Pós-Condições:** O funcionário visualiza o as encomendas atuais da biblioteca e pode ver os detalhes da encomenda.
- **Cenário Principal:**
 1. O funcionário seleciona no menu a área afeta dos empréstimos;
 2. O funcionário vê as encomendas atual [Alternativa A: Dados falhados].
- **Alternativa A:** Dados Falhados: Sistema Informa o Funcionário da falhado houve.

➤ **Aprovar empréstimos (Encomenda):**

- **Pré-condições:** Login efetuado.
- **Pós-Condições:** O funcionário visualiza o as encomendas para analise da biblioteca.
- **Cenário Principal:**
 1. O funcionário seleciona no menu a área afeta dos empréstimos;
 2. O funcionário vê as encomendas atual [Alternativa A: Dados falhados];
 3. O funcionário procurar na lista de empréstimos a linha que quer aprovar [Alternativa B: Falha a registar dados];
 4. O funcionário carrega no botão para aprovar o empréstimo do utente, na linha que deseja;
 5. O funcionário confirma que deseja aprovar.
 6. Funcionário volta à lista de empréstimos [Alternativa B: Falha a registar dados];
- **Alternativa A:** Dados Falhados: Sistema Informa o Funcionário da falhado houve.
- **Alternativa B:** Falha a registar dados: Sistema informa o Funcionário da falha que houve.

➤ **Recusar empréstimos (Encomenda):**

- **Pré-condições:** Login efetuado.
- **Pós-Condições:** O funcionário visualiza o as encomendas para analise da biblioteca.
- **Cenário Principal:**
 1. O funcionário seleciona no menu a área afeta dos empréstimos;
 2. O funcionário vê as encomendas atual [Alternativa A: Dados falhados];

3. O funcionário procurar na lista de empréstimos a linha que quer recusar [Alternativa B: Falha a registar dados];
 4. O funcionário carrega no botão para aprovar o empréstimo do utente, na linha que deseja;
 5. O funcionário confirma que deseja recusar.
 6. Funcionário volta à lista de empréstimos [Alternativa B: Falha a registar dados];
- **Alternativa A:** Dados Falhados: Sistema Informa o Funcionário da falhado houve.
 - **Alternativa B:** Falha a registar dados: Sistema informa o Funcionário da falha que houve.

➤ **Adicionar Livro:**

- **Pré-condições:** Login efetuado e estar á visualizar lista de livros
- **Pós-Condições:** O funcionário insere um novo livro na lista de livros disponibilizadas pela biblioteca.
- **Cenário Principal:**
 1. O funcionário seleciona no menu a área afeta dos Livros;
 2. O funcionário vê todas as informações dos livros existentes na biblioteca [Alternativa A: Dados falhados];
 3. O funcionário carrega no botão para inserir novo livro;
 4. O funcionário preencher o formulário do novo livro e escritor;
 5. O funcionário carrega no botão para inserir novo livro;
 6. A inserção é efetuada [Alternativa B: Falha registar dados].
 7. Funcionário volta à lista dos livros [Alternativa B: Falha a registar dados];
- **Alternativa A:** Dados Falhados: Sistema Informa o Funcionário da falhado houve.
- **Alternativa B:** Falha a registar dados: Sistema informa o Funcionário da falha que houve.

➤ **Editar Livro:**

- **Pré-condições:** Login efetuado e estar á visualizar lista de livros
- **Pós-Condições:** O funcionário atualiza um livro existente na lista de livros disponibilizadas pela biblioteca.
- **Cenário Principal:**
 1. O funcionário seleciona no menu a área afeta dos Livros;

2. O funcionário vê todas as informações dos livros existentes na biblioteca [Alternativa A: Dados falhados];
 3. O funcionário carrega no botão para alterar os dados do livro;
 4. O funcionário vê o formulário de editar livro e escritor com informações do livro;
 5. O funcionário preenche o formulário de editar livro, inserido os dados que pretende alterar;
 6. O funcionário carrega sobre o botão alterar;
 7. A alteração é efetuada [Alternativa B: Falha registrar dados].
 8. Funcionário volta à lista dos livros [Alternativa B: Falha a registrar dados];
- **Alternativa A:** Dados Falhados: Sistema Informa o Funcionário da falhado houve.
 - **Alternativa B:** Falha a registrar dados: Sistema informa o Funcionário da falha que houve.

➤ **Ver Livro:**

- **Pré-condições:** Login efetuado.
- **Pós-Condições:** O funcionário visualiza a lista de livros existente na lista de livros disponibilizadas pela biblioteca.
- **Cenário Principal:**
 1. O funcionário seleciona no menu a área afeta dos Livros;
 2. O funcionário vê todas as informações dos livros existentes na biblioteca [Alternativa A: Dados falhados];
 3. Na linha do livro que quer ver os dados, o funcionário carrega no botão detalhe;
 4. O funcionário vê o formulário de detalhe do livro, com informação do livro e do escritor;
- **Alternativa A:** Dados Falhados: Sistema Informa o Funcionário da falhado houve.

➤ **Apagar Livro**

- **Pré-condições:** Login efectuado.
- **Pós-Condições:** O funcionário remove um livro da lista de livros de aulas, sendo removida da lista.
- **Cenário Principal:**
 1. O funcionário seleciona no menu a área afeta aos mapas de aulas;
 2. O funcionário vê todas as informações dos livros de aulas existentes na biblioteca [Alternativa A: Dados falhados];
 3. O funcionário carrega no botão para apagar o livro que pretende;
 4. O funcionário confirma a remoção do livro;
 5. A remoção é efetuada [Alternativa B: Falha ao registar dados];
 6. Funcionário volta à lista dos mapas de aulas [Alternativa A: Dados falhados].
- **Alternativa A:** Dados falhados: Sistema informa o Funcionário da falha que houve.
- **Alternativa B:** Falha a registar dados: Sistema informa o Funcionário da falha que houve.

• **Cliente (Utentes)**

➤ **Consultar Livros:**

- **Pré-condições:** Login efetuado.
- **Pós-Condições:** O utente visualiza livro existente na lista de livros disponibilizadas pela biblioteca e pode adicionar ao cesto.
- **Cenário Principal:**
 1. O cliente seleciona no menu a área afeta dos Livros;
 2. O cliente vê todos os livros existentes na biblioteca [Alternativa A: Dados falhados];
- **Alternativa A:** Dados Falhados: Sistema Informa o Funcionário da falhado houve.

➤ **Adicionar livros ao cesto:**

- **Pré-condições:** Login efetuado e consultar livros
- **Pós-Condições:** O utente adicionar o livro no cesto e verifica os itens do cesto.
- **Cenário Principal:**
 1. O cliente seleciona no menu a área afeta dos Livros;
 2. O cliente vê todos os livros existentes na biblioteca [Alternativa A: Dados falhados];
 3. Na linha do livro que quer, carrega sobre o botão do carrinho [Alternativa A];
 4. O cliente vai carrega no botão de cesto “localizado na barra de menu” [Alternativa B: Cesto Vazio]
 - 5.
- **Alternativa A:** Dados Falhados: Sistema Informa o Utente da falhado houve.

- **Alternativa B:** Cesto Vazio: Sistema deteta o cesto vazio e informa o utente.

➤ **Encomendar de livros:**

- **Pré-condições:** Login efetuado e adicionar livros ao cesto
- **Pós-Condições:** O utente efetua encomenda
- **Cenário Principal:**
 1. Na área de verifica os itens do cesto.
 2. O cliente carrega sobre o botão de confirmar a encomenda;
 3. Os dados são guardados [**Alternativa A:** dados falhados]
 4. O cliente é direcionado para página de estado da encomenda;
- **Alternativa A:** Dados Falhados: Sistema Informa o Utente da falhado houve.
- **Alternativa B:** Cesto Vazio: Sistema deteta o cesto vazio e informa o utente.

➤ **Ver histórico da Encomendas:**

- **Pré-condições:** Login efetuado
- **Pós-Condições:** O utente efetua encomenda e ver item da encomenda
- **Cenário Principal:**
 1. O cliente seleciona no menu a área afeta a Empréstimos
 2. O cliente vê a lista das encomendas efetuadas encomendas [**Alternativa A:** dados falhados];
 3. O cliente pode gerir as suas encomendas [**Alternativa A:** dados falhados]
- **Alternativa A:** Dados Falhados: Sistema Informa o Utente da falhado houve.

2.4 Requisitos Não Funcionais

Por outro lado, foram identificados os seguintes requisitos não funcionais:

- Confidencialidade dos dados dos utilizadores;
- Ligação a uma base de dados "local" e à da API;
- Atualização da base de dados apenas por utilizadores autenticados;
- O sistema deverá ter disponibilidade total durante toda a sua utilização, especialmente durante a requisição do utilizador, isto permite a confiabilidade do sistema.

3. Desenho do Sistema

Neste capítulo será apresentado todo o processo de idealização do sítio web, onde serão apresentados todos os esboços iniciais das páginas idealizadas para este sistema, ou seja, os protótipos não funcionais, os modelos físicos e entidade/relação de ambas as bases de dados pedidos, a definição da arquitetura do sistema na integração entre a aplicação e a API e as especificações da interface da API.

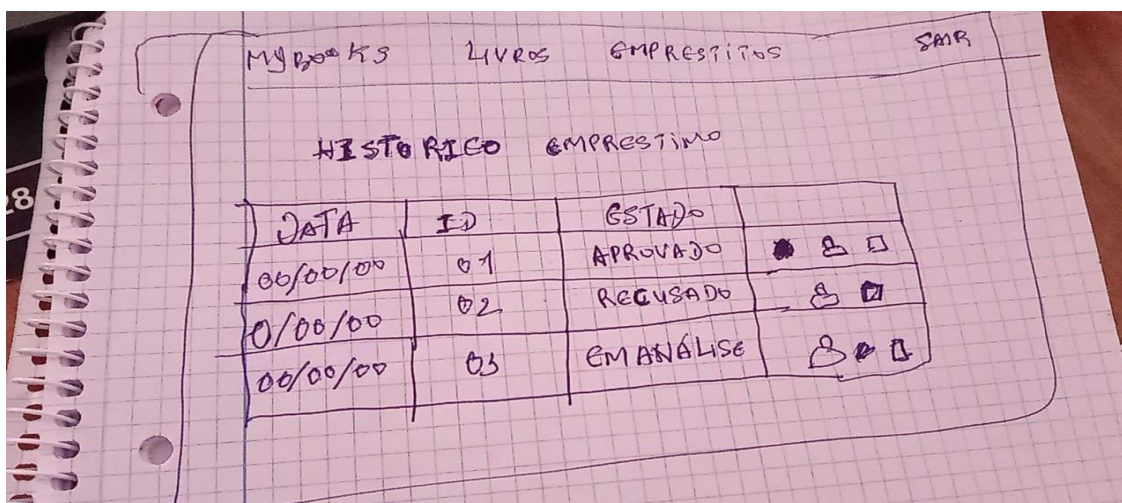
3.1 Modelação de Interfaces

A modelação dos protótipos não funcionais está dividida em três subsecções, (1) protótipos da parte do funcionário, (2) protótipos da parte do cliente e (3) diagrama de navegação. Tanto as wireframes [1] como o diagrama de navegação foram desenhados em papel sendo então protótipos de baixo nível. A Figura 3.1 que se encontra abaixo representa o formulário de início de sessão que é comum aos dois tipos de utilizadores.

3.1.1 Wireframe Funcionário

As figuras seguintes representam todas as interfaces com que o funcionário se depara aquando da utilização e que suportam todos os casos de uso descritos no capítulo anterior.

A Figura 2 wireframes históricos de empréstimos (funcionários) representa a página de Histórico de empréstimos, onde se encontra a informação sobre os empréstimos efetuados pelos utentes.











DATA	ID	ESTADO	
06/00/00	01	APROVADO	  
0/00/00	02	RECUSADO	 
00/00/00	03	EM ANÁLISE	  

Figura 2 wireframes históricos de empréstimos (funcionários)

A Figura 3 Wireframe login (geral) representa página de login é apresentado aos utilizadores para se autenticarem na página.

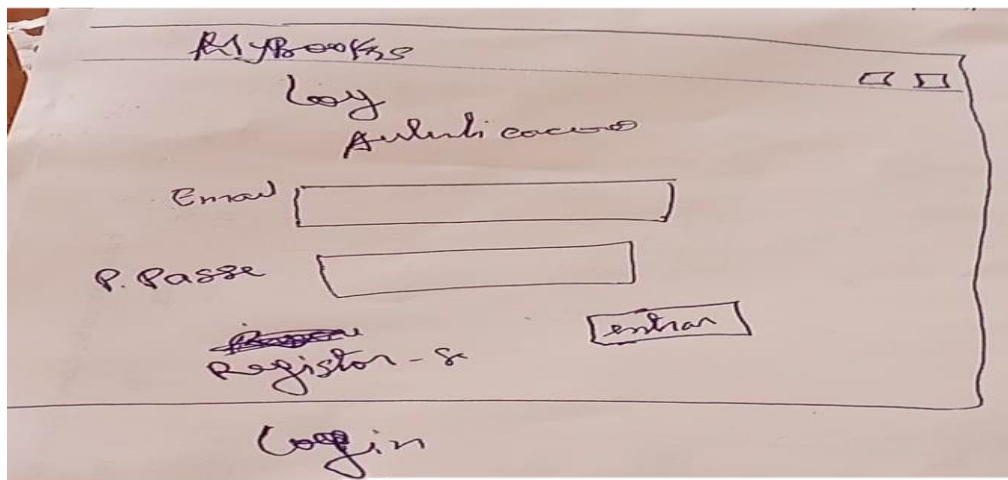


Figura 3 Wireframe login (geral)

A Figura 4 Wireframe Lista de funcionários representa uma lista de funcionários, só parecerá ao gestor da página web com credencias corretas.

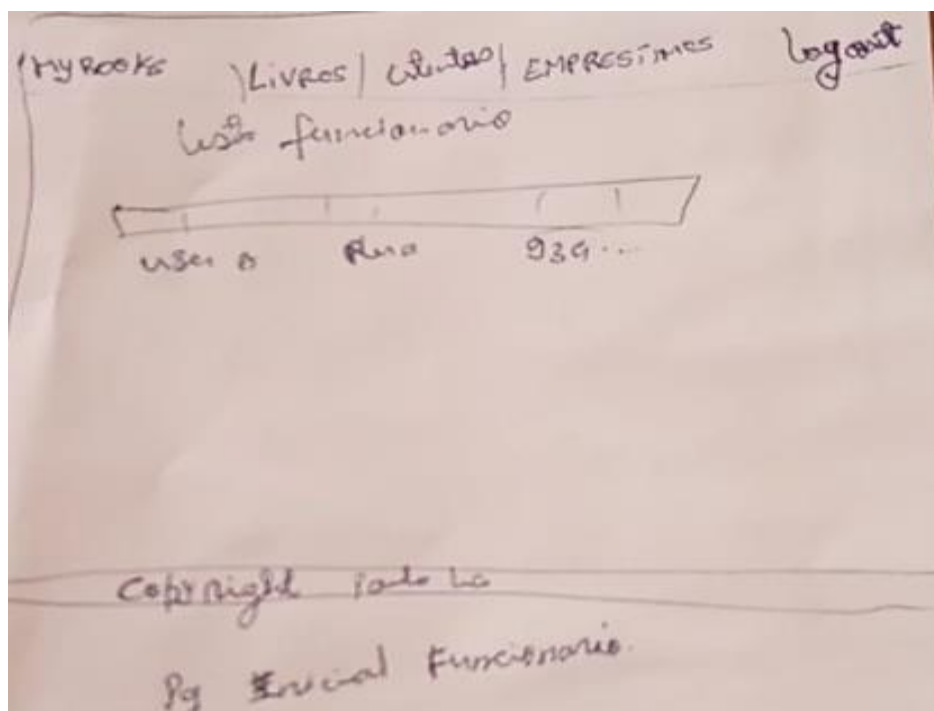


Figura 4 Wireframe Lista de funcionários

Na Figura 5 Wireframe página estado da encomenda está representado á página que detalhe o estado da encomenda feita pelo cliente. Uma página importante, por que informa ao utilizador detalhes da encomenda que efetuou.

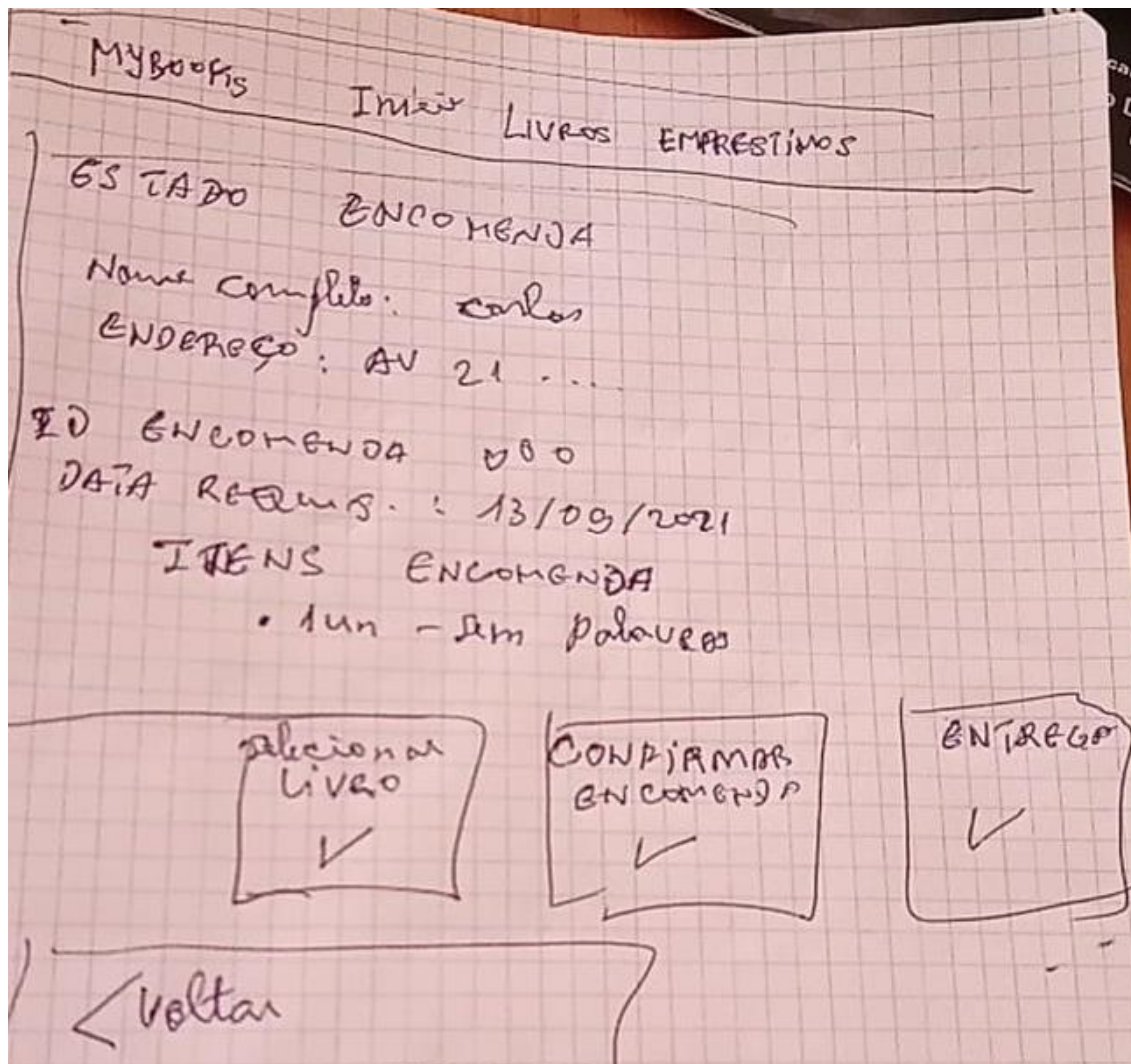


Figura 5 Wireframe página estado da encomenda

Na Figura 6 Wireframe lista de livros (funcionários) mostra representa lista de livros, disponíveis no sistema para utentes. Onde o funcionário poderá gerir os dados dos livros. Já na figura posterior mostra como adicionar um novo livro com preenchimento do formulário, neste nado também devem ser inserido informação dos escritor do livro.

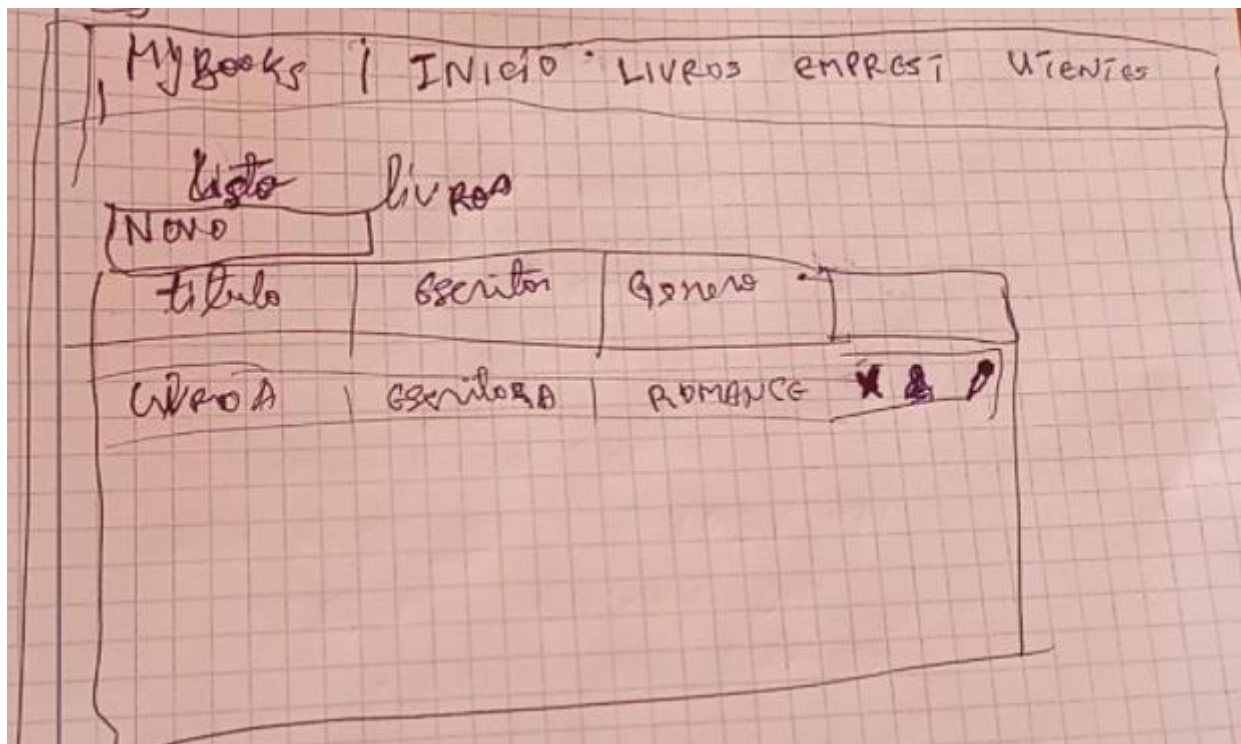


Figura 6 Wireframe lista de livros (funcionários)

A Figura 7 Wireframe criar livro representa á página de para adicionar livro no sistema.

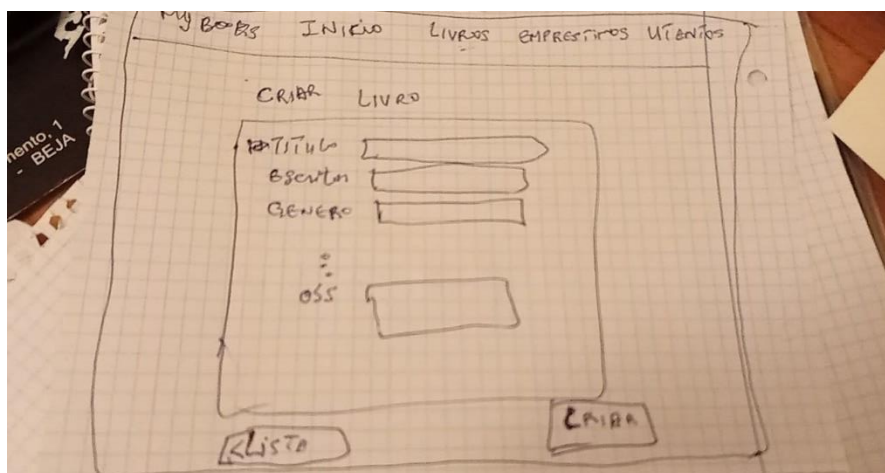


Figura 7 Wireframe criar livro

A Figura 8 Wireframe lista livros funcionários os mapas de livros da biblioteca e com o botão para adicionar um mapa novo, que representa o caso de uso relativo à consulta de livros.

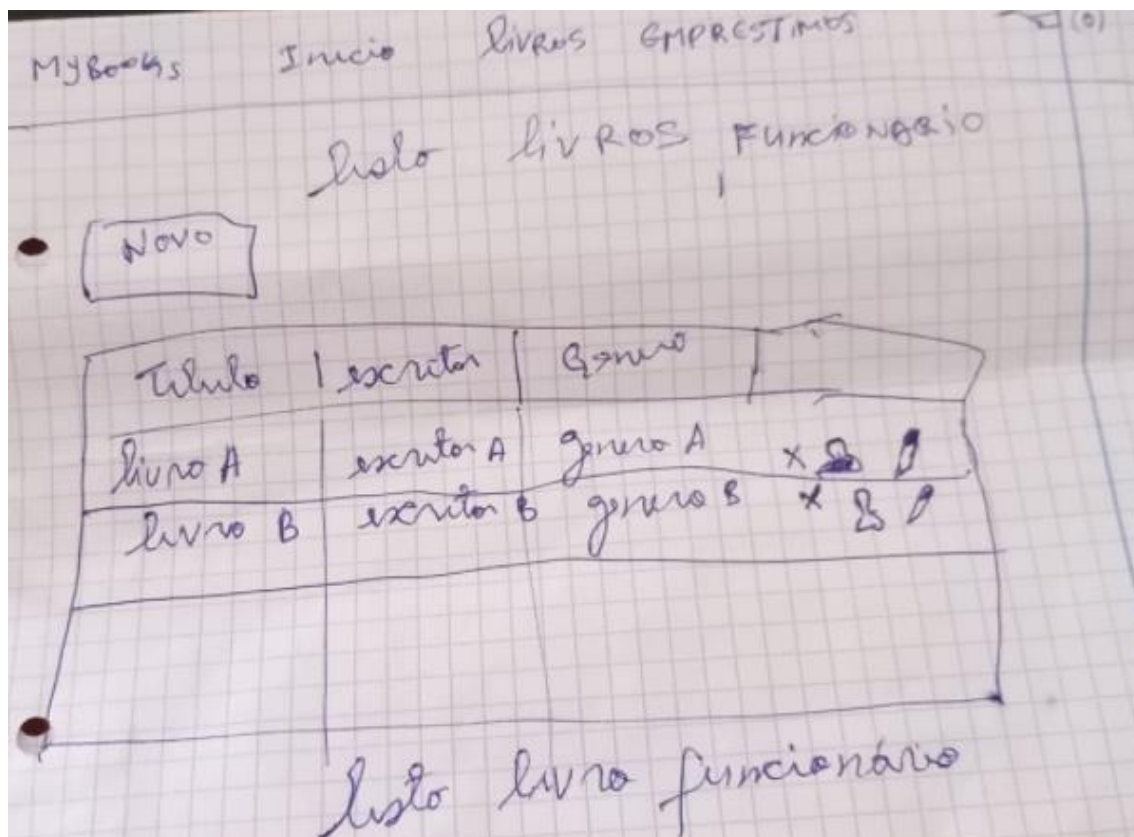


Figura 8 Wireframe lista livros funcionários

3.2 Storyboard de Navegação

As interfaces deste protótipo não funcional e navegação do sistema foram pensados e desenvolvidos de modo a serem o mais simples possível e ao mesmo tempo fiéis aos casos de uso. De um modo geral todos os storyboards desenvolvidos seguem a mesma sequência de ações do caso de uso respetivo.

1. Storyboard Criar conta e login

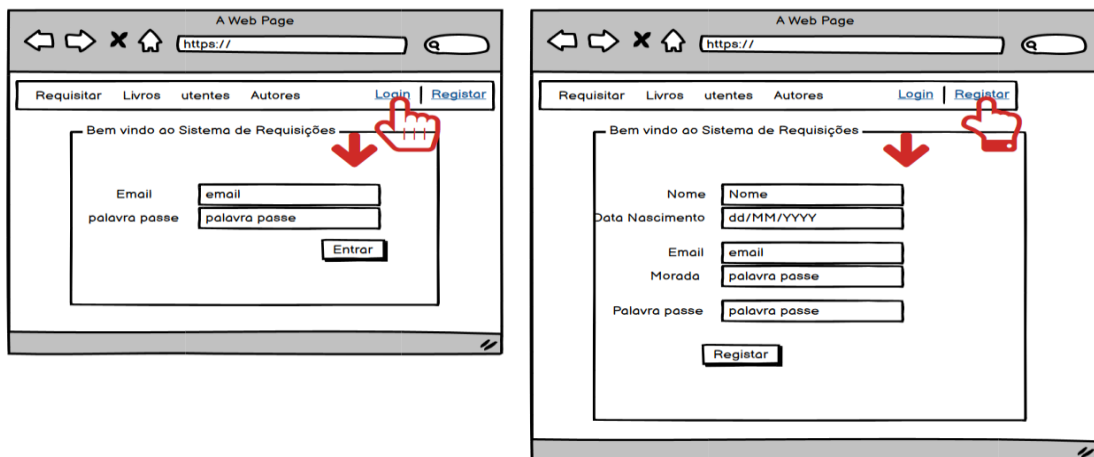


Figura 9 Storyboard Login e Registo

2. Storyboard Livro e criar livro

Na Figura 10 representa-se o storyboard do caso de uso criar livro.

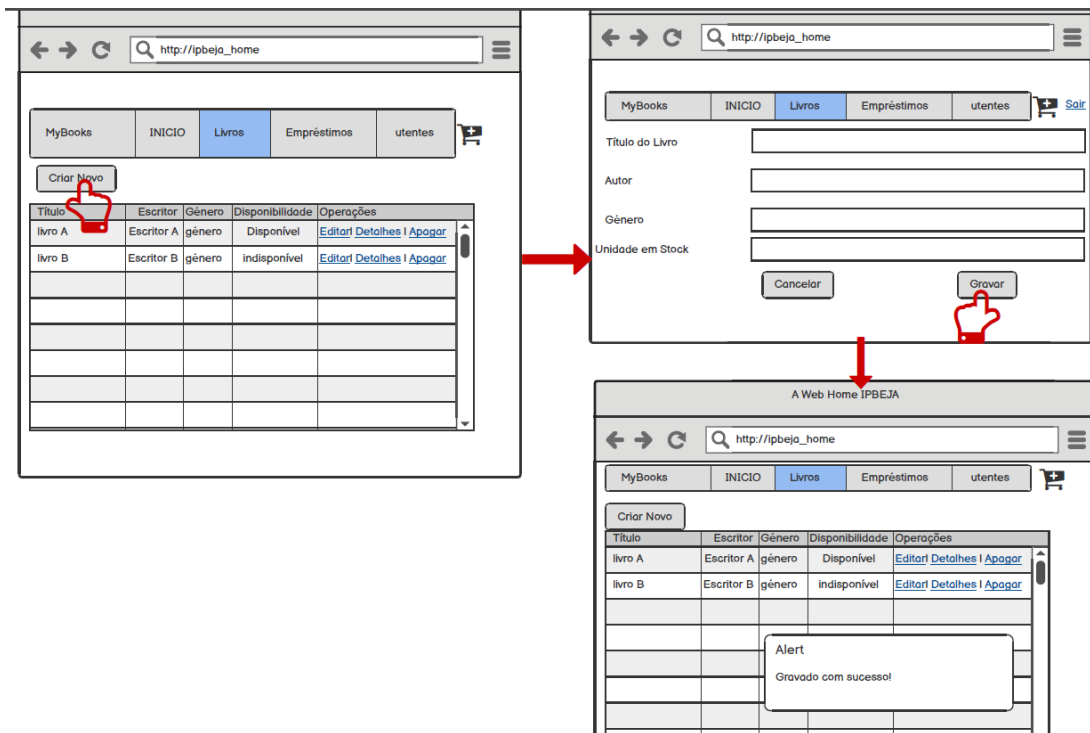


Figura 10 Storyboard Livro e criar livro

3. Storyboard caso de uso aprovar encomenda (empréstimo)

Na Figura 11 está presente o storyboard do caso de uso aprovar empréstimos do lado do funcionário no sistema.

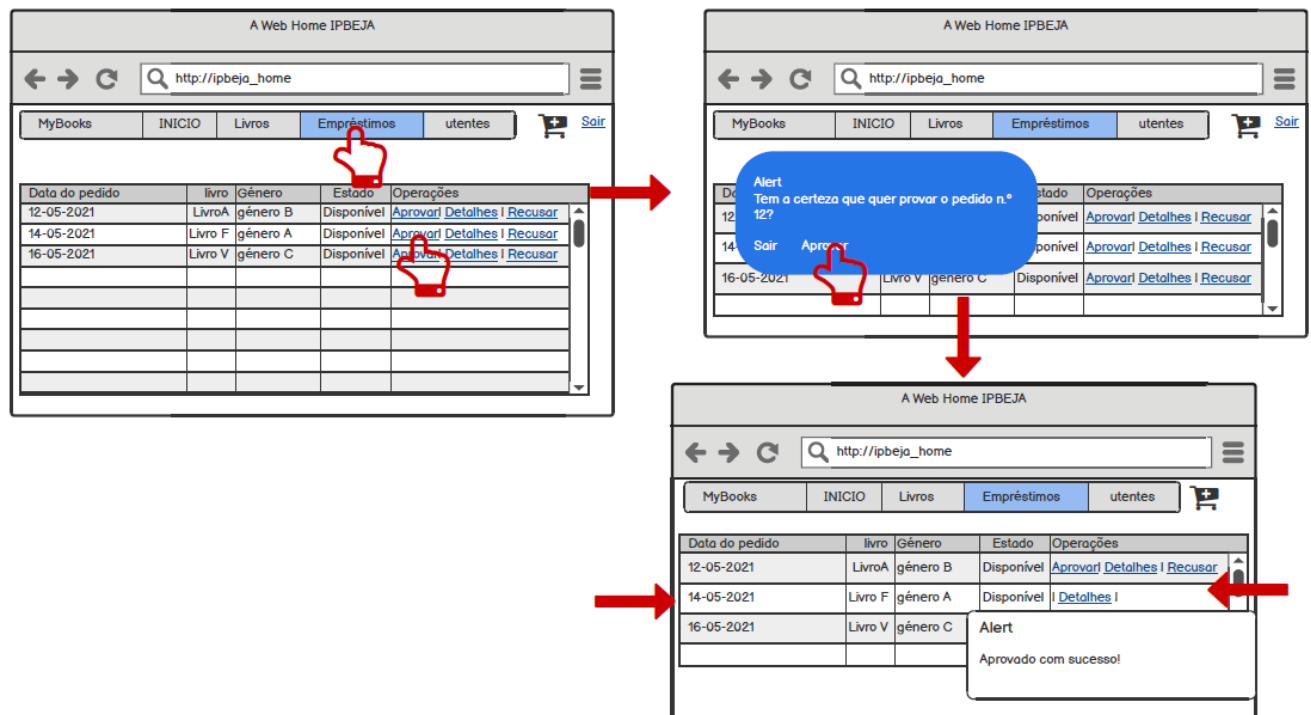


Figura 11 Storyboard Aprovar empréstimo

4. Storyboard caso de uso recusar encomenda (empréstimo)

Na está presente o storyboard do caso de uso aprovar empréstimos do lado do funcionário no sistema.

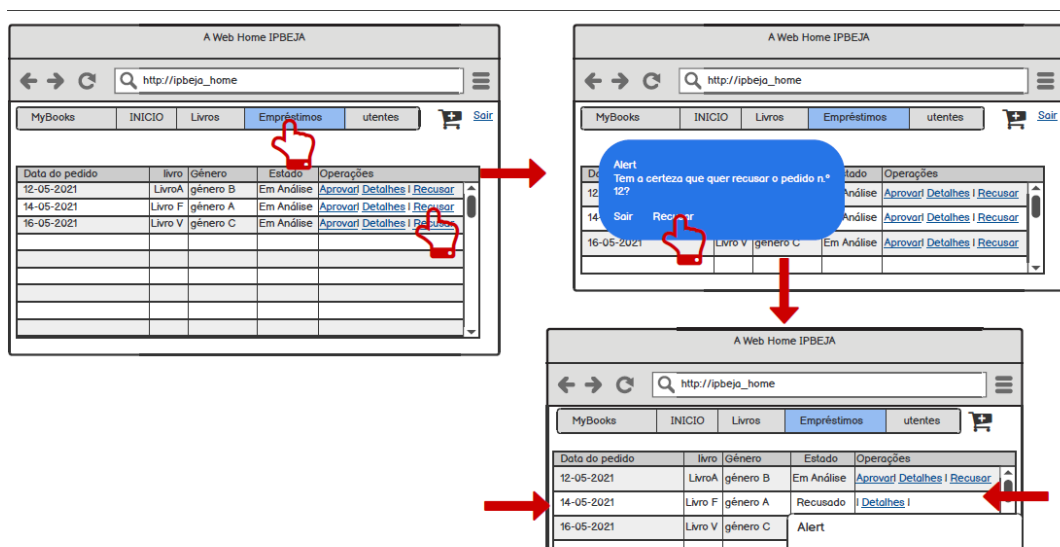


Figura 12 Storyboard recusar empréstimos

5. Storyboard Caso Lista Pedidos e multar

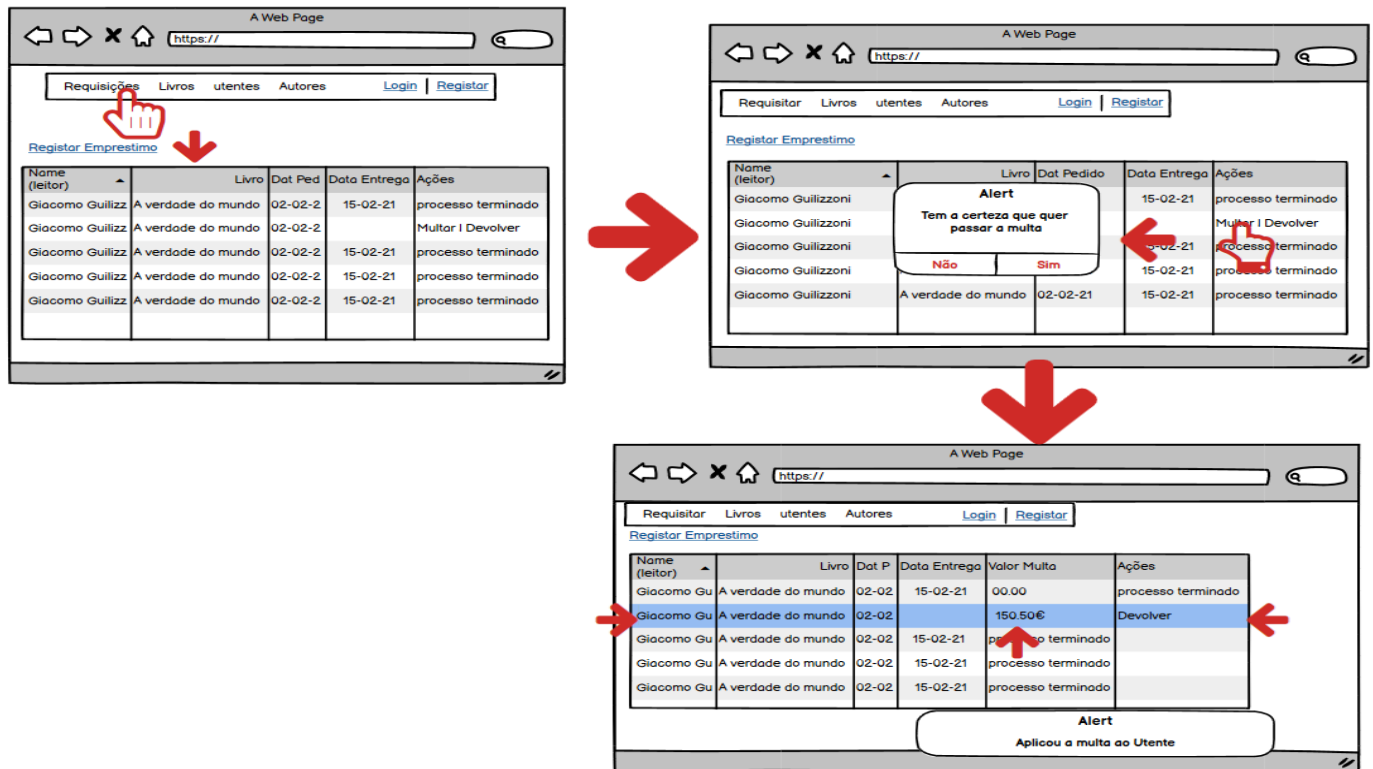


Figura 13 Storyboard Multar

6. Storyboard estado encomenda

A Figura 14 representa o processo para verificar o estado da encomenda feita pelo utente.

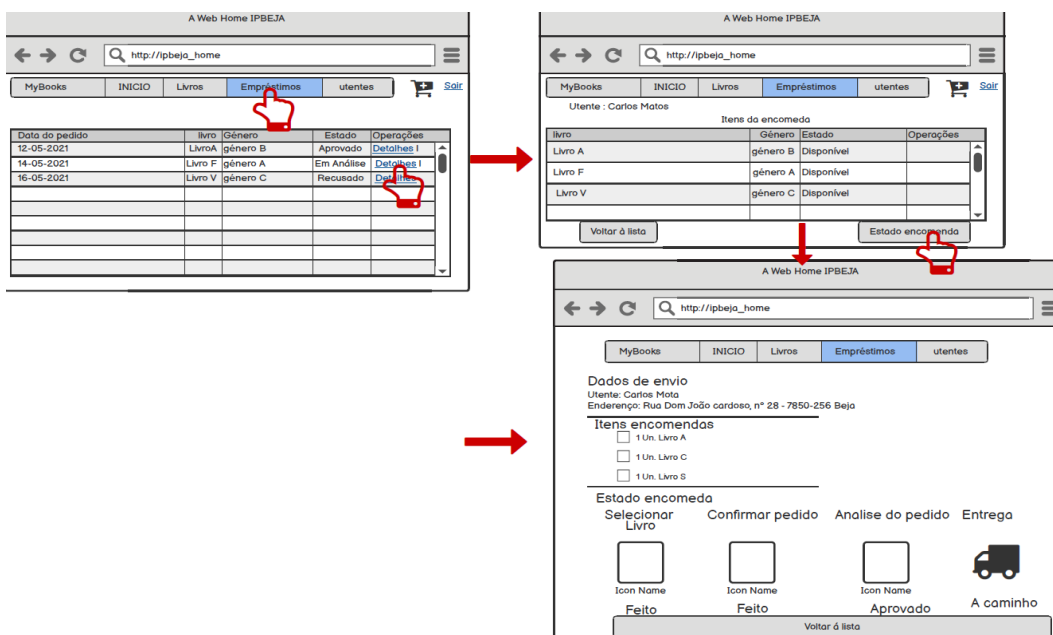


Figura 14 storyboard estado encomenda utente

7. Storyboard adicionar ao cesto e confirmar encomenda

Na Figura 15 demonstra-se o processo de adicionar um livro ao cesto e posteriormente ver os itens que estão no cesto da encomenda. Também se mostra a confirmação da encomenda.

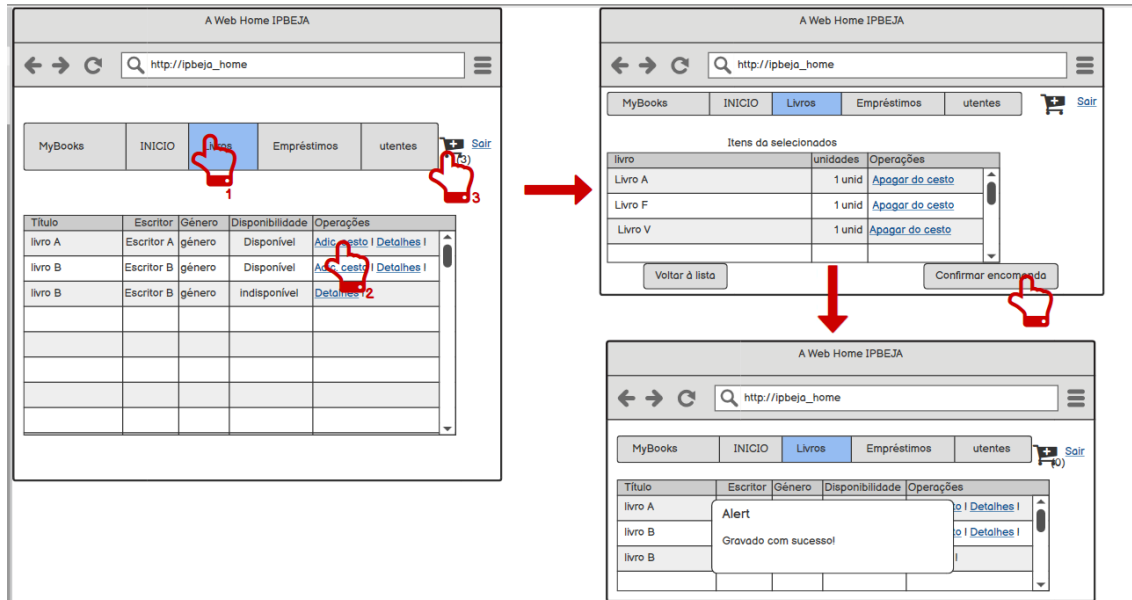


Figura 15 storyboard adicionar livro ao cesto e confirmar encomenda

8. Storyboard gestão Autores

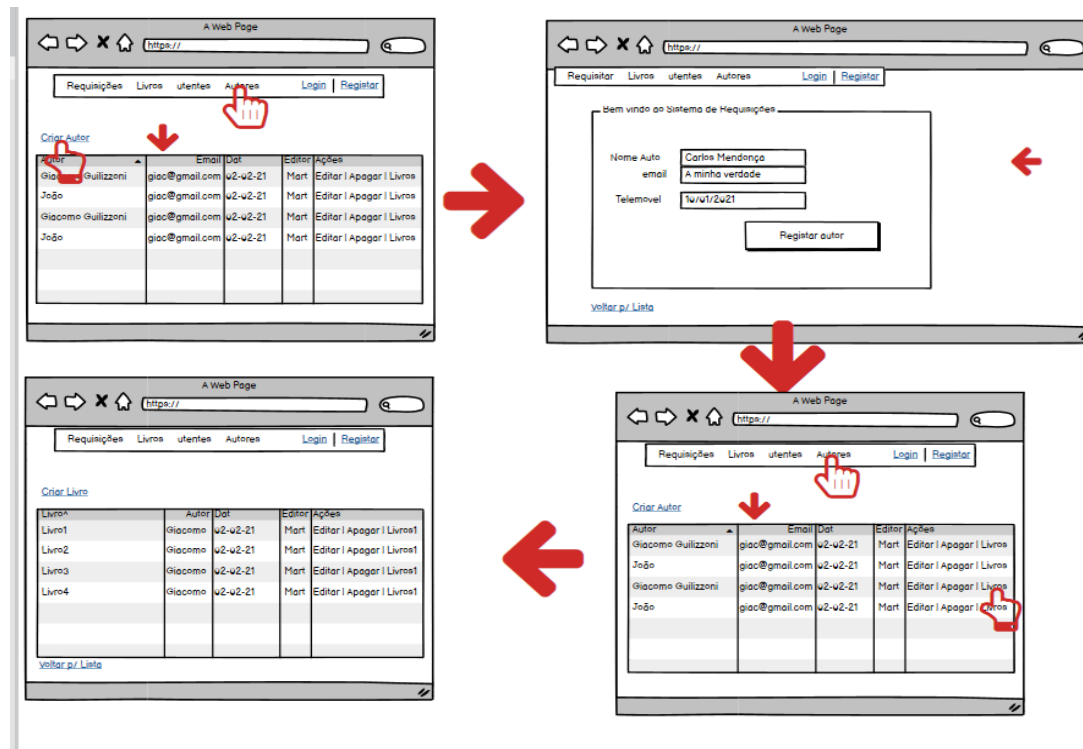


Figura 16 storyboard gestão de Autores

3.2 Modelação das duas bases de dados

3.2.1 Diagrama ER

O diagrama ER desenvolvido para este sistema encontra-se na Figura 17 representa o diagrama ER da base de dados.

Este diagrama oferece um elevado nível de abstração e permite representar as relações entre as várias entidades que compõem a estrutura da base de dados. Assim sendo, podemos a entidade Funcionário pode aprovar ou recusar vários Empréstimos feito pela entidade Utente, também se compreende que um utente tem uma conta de login e uma conta de login pertence á um utente.

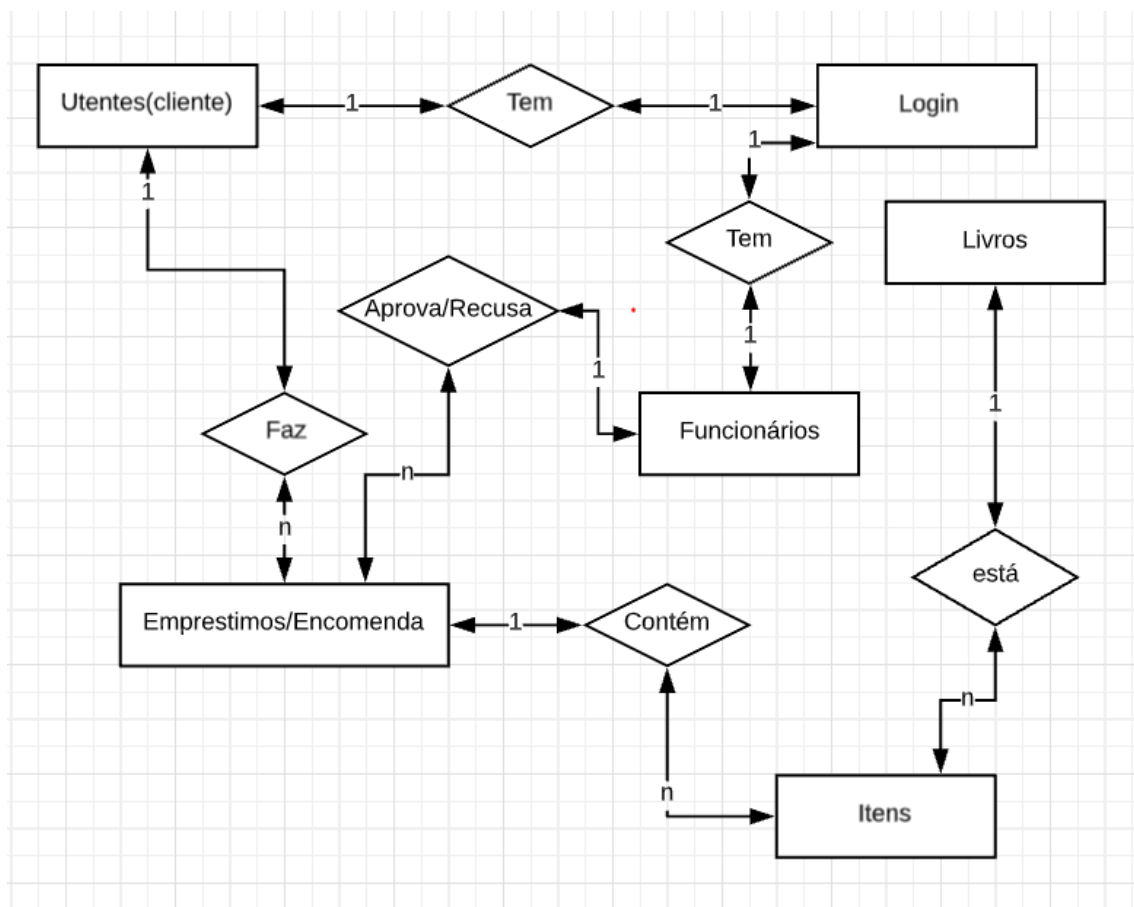


Figura 17 Diagrama E/R

3.2.2 Modelo Físico

A Figura 18 Representa o modelo físico da base de dados do projeto e as relações entre as entidades

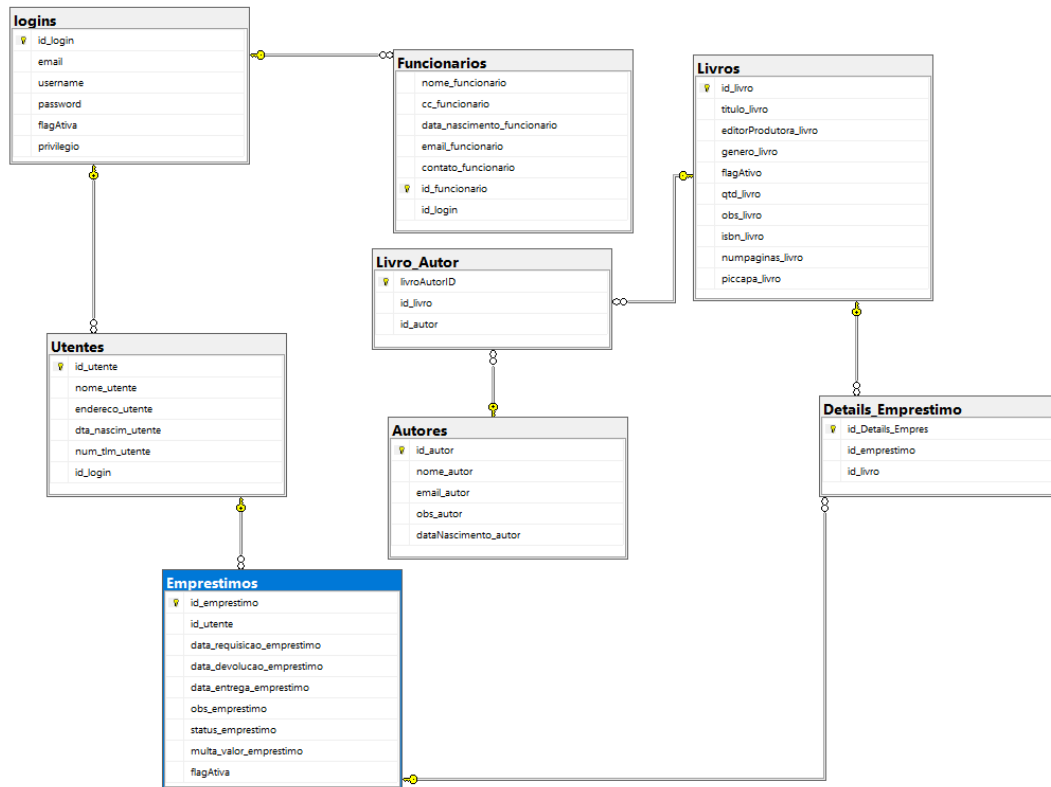


Figura 18 Modelo Físico Base de dados - SQL server

4. Definição da arquitetura do sistema na integração entre a aplicação e a API

Para apresentar a definição da arquitetura do sistema na integração entre aplicação e a API foi realizado um diagrama de blocos que está representado pela, onde existem 3 blocos distintos, os utilizadores, a aplicação e a API, existe também outra parte importante que liga aos blocos da aplicação e da API que são as respetivas bases de dados.

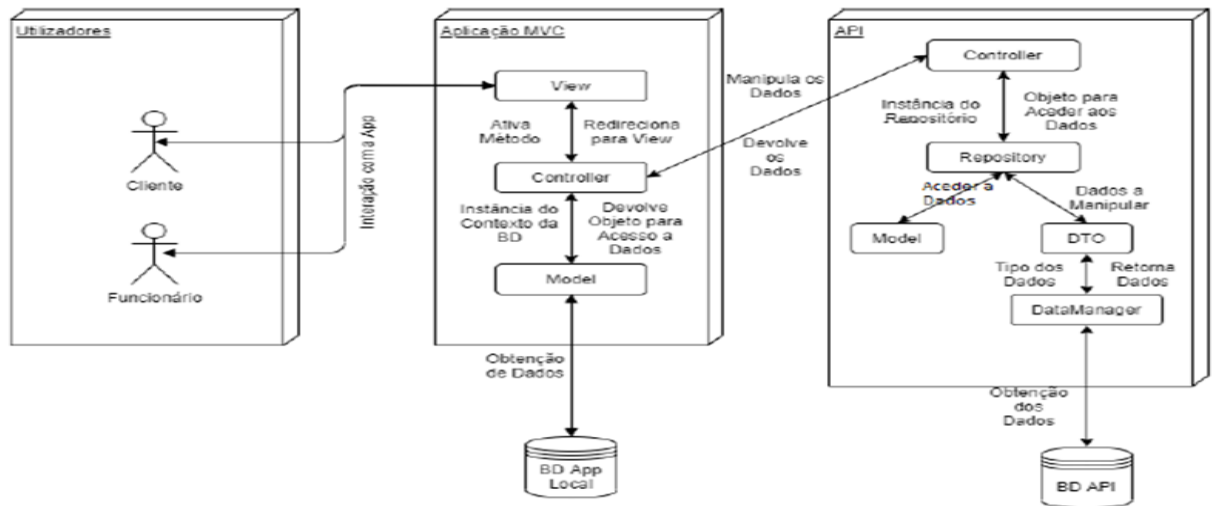


Figura 19 Diagrama de bloco API

4.1 API

Nesta secção serão demonstradas as decisões sobre a API, tal como, (1) a definição da arquitetura do sistema na integração entre a aplicação e a API e (2) a especificação da interface da API do que diz respeito aos seus endpoints.

4.1.1 endpoints da API

A **Erro! A origem da referência não foi encontrada.** representa todos os endpoints da API para realizar operações sobre a entidade “Livros”.

Verbo HTTP	Caminho	Descrição	Corpo de pedido	Corpo de Resposta
GET	localhost:44373/api/livros	Obtém-se todos os livros	Nada	Array 2d com a info dos livros todos
GET	localhost:44373/api/livros/id_livro	Obtém-se um livro por id	Nada	Objeto com a info. Do livro
POST	localhost:44373/api/livros	Adiciona um novo livro	O livro a inserir	Nada
PUT	localhost:44373/api/livros/id_livro	Atualiza um livro, pelo id	O livro a atualizar	Nada
DELETE	localhost:44373/api/livros/id_livro	Apaga um o livro	Nada	Nada

Figura 20 endpoint API

A Figura 21 representa os endpoints para conectar a entidade presente na base de dados da API.

Verbo HTTP	Caminho	Descrição	Corpo de pedido	Corpo de Resposta
GET	localhost:44373/API/empréstimo	Obtém-se todos os empréstimo	Nada	Array 2d com a infor. dos empréstimos todos
GET	localhost:44373/api/emprestimo/id	Obtém-se um livro por id	Nada	Objeto com a infor. Do empréstimo
POST	localhost:44373/api/emprestimos	Adiciona um novo empréstimo	O empréstimo a inserir	Nada
PUT	localhost:44373/api/emprestimos/id	Atualiza um empréstimo, pelo id	O empréstimo a atualizar	Nada
DELETE	localhost:44373/api/emprestimos/id	Apaga um o empréstimo	Nada	Nada

Figura 21 endpoint empréstimos

A Figura 22 representa os endpoint para conectar a entidade presente na base de dados da API.

Verbo HTTP	Caminho	Descrição	Corpo de pedido	Corpo de Resposta
GET	localhost:44373/api/utentes	Obtém-se todos os utente	Nada	Array 2d com a info dos
GET	localhost:44373/api/utentes/id	Obtém-se um livro por id	Nada	Objeto com a info. Do utente
POST	localhost:44373/api/utentes	Adiciona um novo utente	O utente a inserir	Nada
PUT	localhost:44373/api/utentes/id	Atualiza um utente, pelo id	O utente a atualizar	Nada
DELETE	localhost:44373/api/utentes/id	Apaga um o utente	Nada	Nada

Figura 22 endpoint utentes

A Figura 23 representa *endpoint* para aceder as informações do funcionário através da API do sistema.

Verbo HTTP	Caminho	Descrição	Corpo de pedido	Corpo de Resposta
GET	localhost:44373/api/funcionario	Obtém-se todos os funcionários	Nada	Array 2d com a infor. de todos os funcionários
GET	localhost:44373/api/funcionario/id	Obtém-se um livro por id	Nada	Objeto com a infor. do funcionário
POST	localhost:44373/api/funcionario	Adiciona um novo funcionário	O funcionário a inserir	Nada
PUT	localhost:44373/api/funcionario/id	Atualiza um funcionário, pelo id	O funcionário a atualizar	Nada
DELETE	localhost:44373/api/funcionario/id	Apaga um o funcionário	Nada	Nada

Figura 23 endpoint funcionários

5. Implementação do sistema

Neste capítulo será apresentado pequenos excertos de código quer da API, quer da aplicação e será apresentado também alguns exemplos de resultados da aplicação obtidos pelo *software Postman* que é uma ferramenta que permite testar e simplificar o processo da criação de uma API tal como está presente na descrição do sítio web oficial do *Postman* [1].

Os dois temas abordados neste capítulo estão divididos em duas secções, a secção da parte da API e outra da aplicação. Ambos os projetos foram desenvolvidos segundo o modelo de desenvolvimento *database-first*, onde as bases de dados foram criadas e em seguida foram implementadas nos projetos através do comando:

```
Scaffold-DbContext "Server=x;Database=y;Trusted_Connection=True;"  
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

Este comando insere os modelos respetivos na pasta Models do projeto.

5.1 Implementação API

Para a realização da API como foi dito acima, foi utilizado um tutorial disponibilizado pelo professor, criando então a API com uma estrutura que contem *Controllers*, *Models*, *Repository*, *DataManager* e DTO, garantido a segurança de acesso aos modelos da base de dados da API. Foi também seguido outro tutorial para obter mais um nível de segurança na API, este consistiu na utilização de uma chave para aceder à API, ou seja, uma API Key, para a verificação desta chave foi necessário criar um *Middleware* que trata da verificação automática todas as vezes que existem acessos a API, assim, caso não chegue uma chave ou a chave que seja, incorreta a API não devolve resultados.

A estrutura final da API pode-se ver Figura 24 Solution Explorer API que é a estrutura presente no *Solution Explorer*.

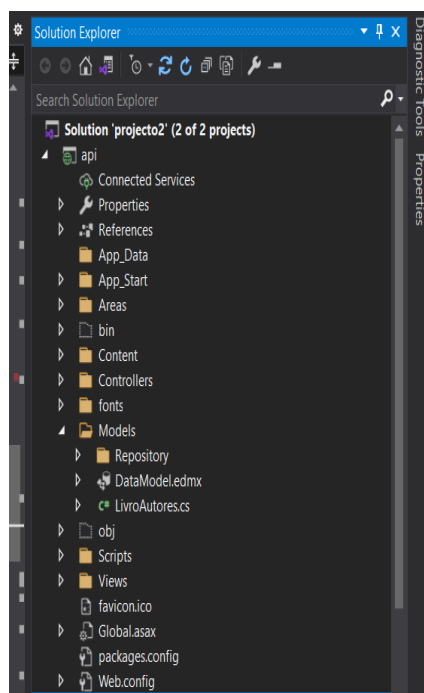


Figura 24 Solution Explorer API

O seguinte excerto de código representa o *Middleware* que trata da chave de acesso, em que é verificado se chega algo nos *headers* do endereço, caso não chegue é dado um erro, caso chegue é então verificado se a chave que chegou corresponde a chave correta, se sim então o processo continua sem erros, caso contrário, é retornado outro erro a dizer que a chave não é autorizada.

```
public async Task InvokeAsync(HttpContext context)
{
    // Chave não fornecida
    if (!context.Request.Headers.TryGetValue(APIKEYNAME, out var
    extractedApiKey))
    {
        context.Response.StatusCode = 401;
        await context.Response.WriteAsync("Api Key was not provided. (
    Using ApiKeyMiddleware) ");
        return;
    }

    var appSettings = context.RequestServices.GetRequiredService<
    IConfiguration>();

    var apiKey = appSettings.GetValue<string>(APIKEYNAME);
```

Figura 25 código autorização API

A Figura 26 excerto de código da parte da API representa os métodos presentes no controlador que tratam dos processos descritos acima, ou seja, tratam da obtenção da lista de empréstimos ou de um único empréstimo.

```
0 references
public class EmpréstimosController : ApiController
{
    private dbEntities db = new dbEntities();

    // GET: api/Empréstimos
    0 references
    public IQueryable<Empréstimo> GetEmpréstimos()
    {
        return db.Empréstimos;
    }

    // GET: api/Empréstimos/5
    [ResponseType(typeof(Empréstimo))]
    0 references
    public IHttpActionResult GetEmpréstimo(int id)
    {
        var empréstimo = db.sp_GetEmpréstimo(id).ToList();

        // Empréstimo empréstimo = db.Empréstimos.Find(id);
        if (empréstimo == null)
        {
            return NotFound();
        }

        return Ok(empréstimo);
    }
}
```

Figura 26 controller Empréstimos

5.2 Teste da API

Por fim os resultados da *API* foram os seguintes como mostra o exemplo na Figura 27, que mostra o resultado obtido pelo *Postman* na obtenção de todos os "livros" com a utilização da chave necessária.

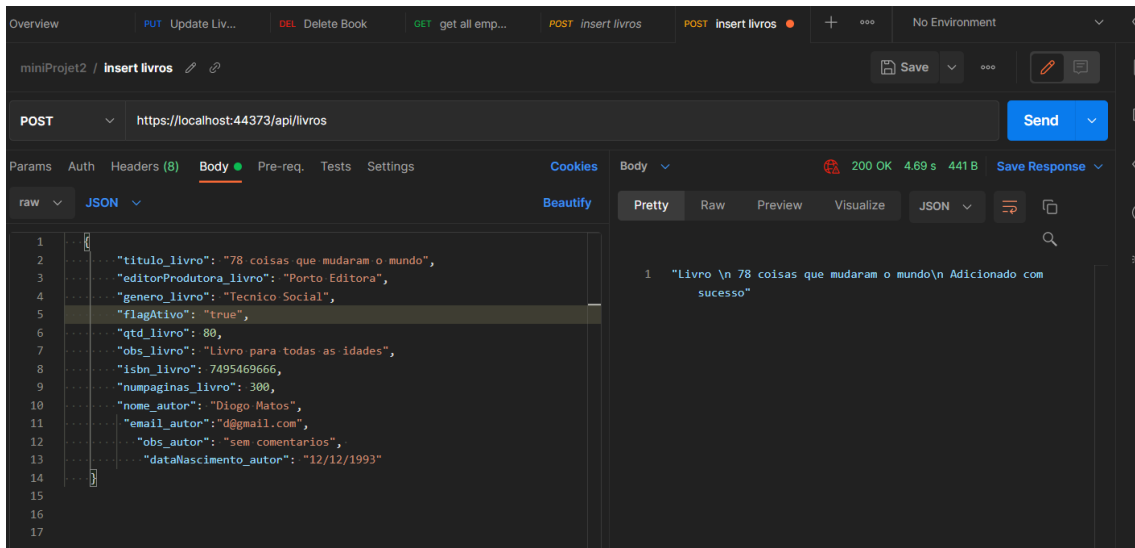


Figura 27 Resultado Livros – Postman

5.3 Aplicação

Em baixo seguem-se alguns excertos de código da aplicação e a sua estrutura. A estrutura da aplicação seguiu o que era requisitado que era o modelo MVC (Model-View-Controller) sendo que nos modelos foi necessário criar os modelos iguais aos da API para se conseguir ter compatibilidade entre os dados da API e os utilizados na aplicação, ficando os modelos divididos em duas diretorias uma da aplicação e outra da API, a estrutura está representada na Figura 28 que é o conteúdo do *Solution Explorer* da aplicação.

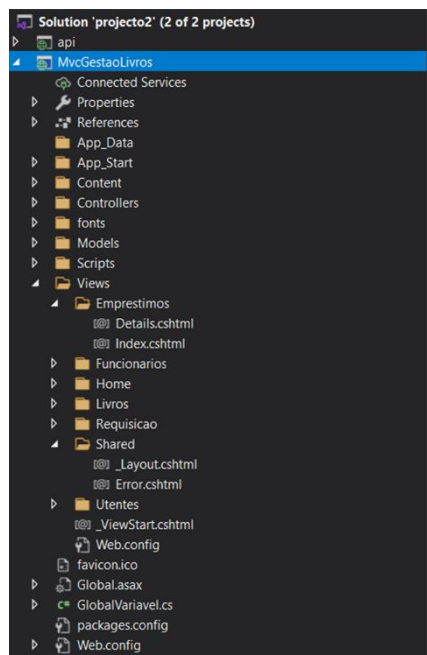


Figura 28 Solution explorer aplicação

5.4 Implementação dos processos críticos

Os processos mais críticos da implementação dos casos de uso, encontram-se representados com excertos de código em baixo.

Código classe cart

Na Figura 29 classe Cart represente uma viewModel para mostra no view checkout, a classe é constituída por quatro atributos que são:

- Id_livro – representa identificador do livro que está na lista.
- Titulo livro – representa o título do livro que está no cesto.
- Qtd – idêntica a quantidade deste livro adicionado á lista.
- Obs_livro – representa uma eventual restrição ou informação adicional para o utente.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.ComponentModel.DataAnnotations;
5  using System.Linq;
6  using System.Web;
7
8  namespace MvcGestaoLivros.Models
9  {
10     0 references
11     public class Cart
12     {
13         [DisplayName("ID")]
14         0 references
15         public int id_livro { get; set; }
16
17         [DisplayName("Título")]
18         //[Required(ErrorMessage = "titulo livro - é obrigatório")]
19         0 references
20         public string titulo_livro { get; set; }
21         [DisplayName("Qtd")]
22         0 references
23         public string qtd { get; set; }
24
25         [DisplayName("Obs")]
26         0 references
27         public string obs_livro { get; set; }
28
29         // [DisplayName("Editor/Produtor")]
30     }
31 }
```

Figura 29 classe Cart

Código para adicionar Livro no carrinho

Na Figura 30 está representado o trecho de código que agiliza o processo de adicionar livro ao cesto. O método recebe como parâmetro id do livro e redireciona para área do livro.

O método AddCart tem como função adicionar um novo livro a lista de livros. Primeiro vai á API procurar a informação do livro, posteriormente passa as informações para o *model* do Livro, verifica se o cesto está vazio, se sim então cria uma *Session* com novo livro e adiciona a variável global *BooktoCart* "Lista de Livros". Caso no cesto encontra-se algum livro então adiciona o novo livro ao BooktoCart e guarda na *session*. No final atualiza a *session* quantidade itens - para apresentar ao utilizador o número de item que tem no cesto.

```
159  /*  
160  metodo para adicionar Livro no carrinho e guardar numa lista de Livros  
161  */  
162  0 references  
163  public ActionResult AddCart(int id)  
164  {  
165      try  
166      {  
167          HttpResponseMessage response = GlobalVariavel.web.GetAsync("Livros/" + id.ToString()).Result;  
168          LivrosModel livro = response.Content.ReadAsAsync<LivrosModel>().Result;  
169          LivrosModel newItem = new LivrosModel();  
170          newItem.id_livro = livro.id_livro;  
171          newItem.titulo_livro = livro.titulo_livro;  
172          newItem.editorProdutora_livro = livro.editorProdutora_livro;  
173          newItem.genero_livro = livro.genero_livro;  
174          newItem.flagAtivo = livro.flagAtivo;  
175          newItem.obs_livro = livro.obs_livro;  
176          newItem.isbn_livro = livro.isbn_livro;  
177          newItem.numpaginas_livro = livro.numpaginas_livro;  
178          newItem.piccapa_livro = livro.piccapa_livro;  
179          newItem.qtd_livro = 1;  
180          if(Session["cart"] == null)  
181          {  
182              BookToCart.Add(newItem);  
183              Session["cart"] = BookToCart;  
184          }  
185          else  
186          {  
187              //update in list cart  
188              foreach (var item in Session["cart"] as List<MvcGestaoLivros.Models.LivrosModel>)  
189              {  
190                  BookToCart.Add(item);  
191              }  
192              BookToCart.Add(newItem);  
193              Session["cart"] = BookToCart;  
194          }  
195          Session["qtdItems"] = (int)BookToCarts.Count;  
196      }catch (Exception ex)  
197      {  
198          Console.WriteLine(ex.ToString());  
199      }  
200  }
```

Figura 30 implementação caso critico - Adicionar livro ao cesto

Processo critico remoção do item do cesto

Na Figura 31 representa-se a implementação de uns dos casos críticos do sistema. Esse método é chamado na página web do cliente quando pressiona o botão de remover item do cesto.

Quando é chamado esse método traz id do item que deseja ser removido. Verifica-se se o carrinho tem algum livro, se sim então adiciona os itens guardados na variável do sistema Session ["cart"] faz o cast e adiciona á lista de livros. De seguida remove o item com igual ao id do parâmetro de entrada, atualiza as variáveis Session["cart"] e Session["qtditens"], para finalizar redireciona para página que o chamou.

```

322 public ActionResult RemoveItem(int id)
323 {
324     try
325     {
326         if (Session["cart"] != null)
327         {
328             //remove item to list cart
329             foreach (var item in Session["cart"] as List<MvcGestaoLivros.Models.LivroModel>)
330             {
331                 BookToCart.Add(item);
332             }
333
334             BookToCart.RemoveAll((x) => x.id_livro.Equals(id));
335
336             // BookToCart.Add(newItem);
337             Session["cart"] = BookToCart;
338         }
339
340         Session["qtditens"] = (int)BookToCarts.Count;
341         UpdateListBook();
342     }
343     catch (Exception ex)
344     {
345         Console.WriteLine(ex.ToString());
346         Console.WriteLine(ex.Message);
347     }
348
349     return RedirectToAction("Checkout");
350 }
351
352
353
354

```

Figura 31 implementação do processo critico - remover item do cesto

Classe Funcionário

A classe funcionário representa a entidade da base de dados Funcionário, mas com papéis diferentes. Essa classe tem como função representar os funcionários na aplicação, é constituído por catorze atributos e também métodos de validação e requisição dos dados de forma coerente aos utilizadores do sistema. A Figura 32 representa a classe funcionário um elemento importante na implementação do sistema.

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Web;
6
7 namespace MvcGestaoLivros.Models
8 {
9     3 references
10     public class Funcionarios
11     {
12         [Display(Name = "Nome")]
13         [Required(ErrorMessage = "Nome funcionario - é obrigatório")]
14         public string nome_funcionario { get; set; }
15         [Display(Name = "Nº Documento")]
16         public string cc_funcionario { get; set; }
17
18         [Display(Name = "Data Nascimento")]
19         [Required(ErrorMessage = "data nascimento funcionario - é obrigatório")]
20         [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
21         public System.DateTime data_nascimento_funcionario { get; set; }
22         [Display(Name = "Email")]
23         public string email_funcionario { get; set; }
24         [Display(Name = "Telemovel")]
25         public string contato_funcionario { get; set; }
26         [Display(Name = "ID")]
27         public int id_funcionario { get; set; }
28         [Display(Name = "ID login")]
29         [Required(ErrorMessage = "id login - é obrigatório")]
30         public int id_login { get; set; }
31     }
32 }

```

Figura 32 implementação da classe funcionário

6. Testes de usabilidade

Para ser possível testar a eficiência e clareza dos processos desenvolvidos foi necessário recolher informações juntos de possíveis utilizadores do sistema, com vista na avaliação da usabilidade do protótipo.

Na realização dos testes foram utilizados protótipos final de modo a detetar possíveis falhas e aspetos a melhorar.

Os testes foram realizados através do zoom online, num ambiente calmo e silencioso, com recurso a apresentação dos PBMF dos ecrãs referentes ao protótipo do sítio web. Foram utilizadas câmaras de smartphones para gravação dos vídeos dos testes.

Antes da realização dos testes, foi pedido aos utilizadores que assinassem um consentimento que autoriza que as suas ações sejam gravadas em vídeo e, posteriormente, utilizadas para fins de avaliação. No final do teste, os utilizadores responderam a um breve questionário que avalia a usabilidade do sistema.

Cada utilizador foi recebido individualmente e foi-lhe exposto que tarefa iria realizar através de um enunciado. As instruções foram declaradas de forma objetiva deixando espaço os utilizadores realizarem as mesmas de forma de forma autónoma e sem ajuda. No final de cada tarefa o utilizador responderá ao questionário.

6.1 Questionário e o Resultado do teste

O questionário é composto pelas seguintes perguntas e devem ser respondidas com valores entre 1 e 5, com exceção da última pergunta, em que 1 é o valor menos positivo e 5 é o valor mais positivo:

1. Simplicidade das tarefas?
2. Objetividade da interface?
3. Facilidade na navegação?
4. Satisfação ao usar o sistema?
5. Sugestões?

6.2 Resultado da avaliação

Nesta avaliação foram feitas a um grupo curto de pessoas devido a pandemia, as idades compreendias entre 24 aos 65 anos, o utente mais velho tem 65 anos, com poucas experiências em informática, mas apresentou facilidade na utilização das tarefas. Como se pode observar na Figura 33

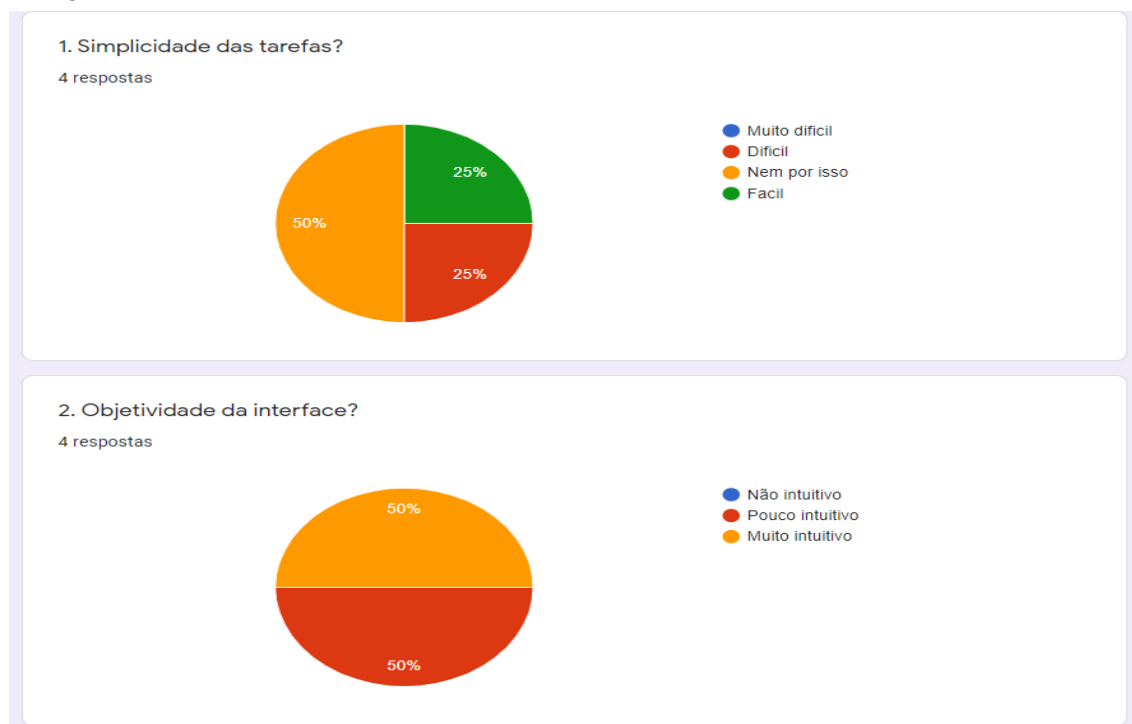


Figura 33 Resultado 1

Nesta avaliação tentou-se obter uma ideia de como está a usabilidade das tarefas e interação dos utilizadores. Como se pode observar no gráfico abaixo (Figura 34) existe um elevado grau de satisfação quanto a utilização do sistema, mas os mais adultos (representam os 16.7% - na Facilidade de navegação) com pouca experiência em informática acharam que o sistema é difícil de navegar entre as tarefas.

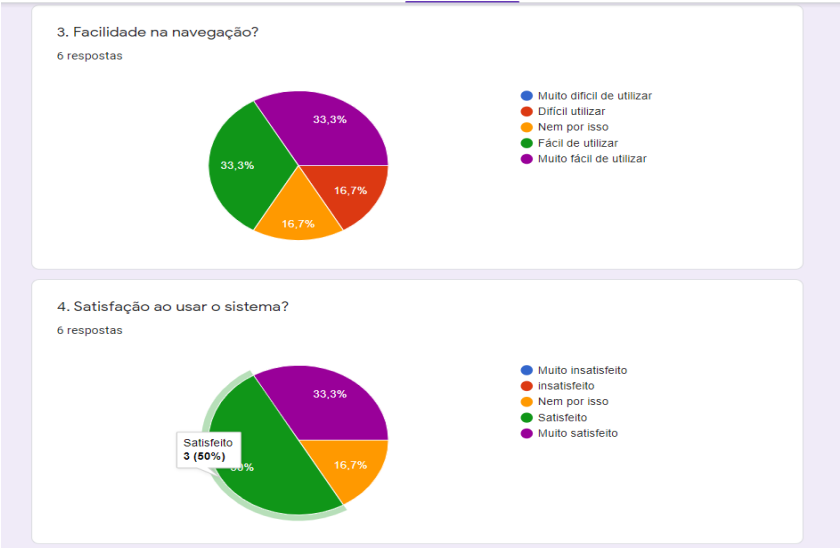


Figura 34 Resultado 2

6.3 Métrica do teste de usabilidade

Na tabela abaixo - Figura 35, apresenta-se valores do teste retirado do teste de usabilidade. Métricas obtidas através dos testes realizados:

Tarefas	Tempo	Número de Páginas abertas	Número de cliques	Número de erros cometidos
1	16 segundos	2	2	0
2	32 segundos	3	5	0
3	13 segundos	2	2	0
4	-----	3	8	0
5	19 segundos	2	3	0
6	17 segundos	2	8	0

Figura 35 Métricas obtidas através dos testes realizados

7. Conclusão

Com o desenvolvimento deste trabalho foi possível perceber a importância das fases de Análise e Desenho antes da fase de Implementação. Estas permitiram ter uma melhor compreensão do sistema que se pretendia desenvolver, assim como, dos seus utilizadores e as ações que realizariam no sistema. Deste modo foi possível avançar para a fase de Implementação com ideias bem definidas, reduzindo assim o risco de serem cometidos erros.

A escolha da tecnologia ASP.NET MVC com API revelou ser a acertada, pois permitiu aplicar o padrão MVC de uma forma intuitiva e pouco demorada.

Surgiram algumas dificuldades na fase de implementação especialmente na parte da implementação do repositório e dos DTO pois o código do tutorial seguido não estava completo e nem explicava como fazer certas partes importantes e foi necessário fazer essa parte consoante as necessidades, houve vezes que não funcionava da forma mais correta e foi necessário adaptar novamente, ficando no final a funcionar como necessário.