



UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA

BookMe

Gestão de Reservas de Livros Grupo 5



(a) Ana Ribeiro (A82474)



(b) Jéssica Lemos (A82061)



(c) Miguel Pereira (A78912)

MIEI - 4º Ano - Engenharia de Aplicações

Braga, 20 de Julho de 2020

Conteúdo

1	Introdução	3
1.1	Contextualização	3
1.2	Definição do Sistema	3
2	Estabelecimento de Requisitos	3
2.1	Perfis de Utilizador	3
2.2	Requisitos Funcionais	4
2.2.1	Utilizador	4
2.2.2	Requisitante	4
2.2.3	Funcionário	5
2.2.4	Responsável	5
2.2.5	Administrador	5
3	Modelação	5
3.1	Modelo de Domínio	5
3.2	Diagrama de Use Cases	6
3.3	Modelo Independente da Plataforma (PIM)	7
3.4	Modelo Específico da Plataforma (PSM)	9
4	Análise de Tarefas	12
5	Prototipagem	13
5.1	Mockups da aplicação	13
5.2	Mapa de Navegação	15
6	Implementação	16
6.1	Arquitetura	16
6.2	Camada de dados	17
6.3	Camada de negócio	17
6.4	Camada de apresentação	18
7	Interface Web	19
8	Deployment	24
8.1	Docker	24

8.2 Docker-Compose	25
9 Testes de carga	27
10 Conclusão	30

1 Introdução

1.1 Contextualização

Uma prática bastante comum na população é a requisição de livros nas bibliotecas, daí o número de bibliotecas existentes em todo o país ser tão elevado. Contudo, deparamo-nos com um sistema um pouco arcaico na maioria destas em que a reserva de um livro é feita por telemóvel ou por email, quando o é possível fazer. Para além disso, a informação do catálogo de livros de cada biblioteca ou não se encontra à disposição dos utilizadores ou então é dispersa. Estes são dois dos problemas mais visíveis e que levam a menor satisfação dos utilizadores sendo que na maior parte dos casos se opta pela compra desse mesmo livro. Deste modo, o sistema *BookMe* pretende facilitar e satisfazer todo o processo de requisição de livros.

1.2 Definição do Sistema

Esta aplicação web permite a gestão tanto de bibliotecas como dos livros. Assim, os requisitantes criando uma conta pode aceder às bibliotecas que fazem parte do sistema bem como os respetivos livros. Torna-se possível, para além de uma procura rápida e eficaz, saber em tempo real a disponibilidade dos livros nas bibliotecas.

Enquanto que as bibliotecas conseguem ter o melhor gestão tanto dos seus catálogos como dos livros disponíveis, efetuando a requisição e devolução dos livros reservados no sistema.

2 Estabelecimento de Requisitos

2.1 Perfis de Utilizador

Analisando o sistema em causa conseguimos identificar os utilizadores que fazem parte do mesmo:

- O **requisitante** irá utilizar a aplicação para pesquisar e reservar livros de modo a efetuar a requisição aquando do levantamento do mesmo na biblioteca, bem como a respetiva devolução

- O **funcionário** da biblioteca está responsável por gerir o catálogo de livros disponibilizados pela biblioteca, bem como registar como reservados e devolvidos os livros
- O **responsável** da biblioteca, para além de poder efetuar as mesmas funções que o funcionário tem ainda a responsabilidade de gerir os funcionários
- O **administrador** apenas irá gerir as bibliotecas existentes no sistema

2.2 Requisitos Funcionais

Considerando os principais objetivos desta aplicação, nesta secção são apresentas os requisitos funcionais estabelecidos para cada tipo de utilizador.

2.2.1 Utilizador

1. Como utilizador deverá poder efetuar login, indicando o email e a password, e logout.

2.2.2 Requisitante

2. O sistema deverá permitir que um requisitante se registe.
3. O requisitante poderá editar o seu perfil.
4. O requisitante deverá poder efetuar a reserva de um livro, desde que não tenha devoluções em atraso.
5. O sistema deverá permitir que um requisitante cancele uma das suas reservas.
6. O requisitante deverá poder renovar uma requisição caso o livro esteja disponível.
7. O requisitante deverá poder consultar todos os seus processos, nomeadamente as requisições, as reservas e as devoluções.
8. O requisitante poderá consultar as suas notificações.
9. O requisitante deverá poder consultar os livros do sistema, bem como aplicar filtros.
10. O requisitante poderá consultar as bibliotecas existentes.

2.2.3 Funcionário

10. O sistema deverá permitir que o funcionário consulte o catálogo de livros da sua biblioteca, bem como aplicar filtros.
11. O funcionário deverá poder efetuar o registo de um livro.
12. O sistema deverá permitir o funcionário editar o seu perfil.
13. O sistema deverá fornecer uma listagem dos livros reservados e requisitados, bem como permitir aplicar filtros.
14. O funcionário deverá marcar quando um livro é requisitado.
15. O funcionário deverá assinalar quando um livro é devolvido.

2.2.4 Responsável

14. O sistema deverá permitir o responsável consultar os funcionários da biblioteca.
15. O responsável poderá editar o perfil da biblioteca.
16. O responsável deverá poder registar os funcionários da biblioteca.

2.2.5 Administrador

15. O administrador poderá efetuar o registo de uma biblioteca no sistema, bem como do seu responsável.
16. O sistema deverá permitir que o administrador elimine uma biblioteca do sistema.
17. O administrador poderá consultar as bibliotecas presente no sistema, podendo aplicar filtro para uma pesquisa mais rápida.

3 Modelação

3.1 Modelo de Domínio

Com base no levantamento de requisitos efetuado, foi possível desenvolver o modelo de domínio apresentado de seguida e onde é possível verificar as principais entidades

do sistema e o modo como se relacionam.

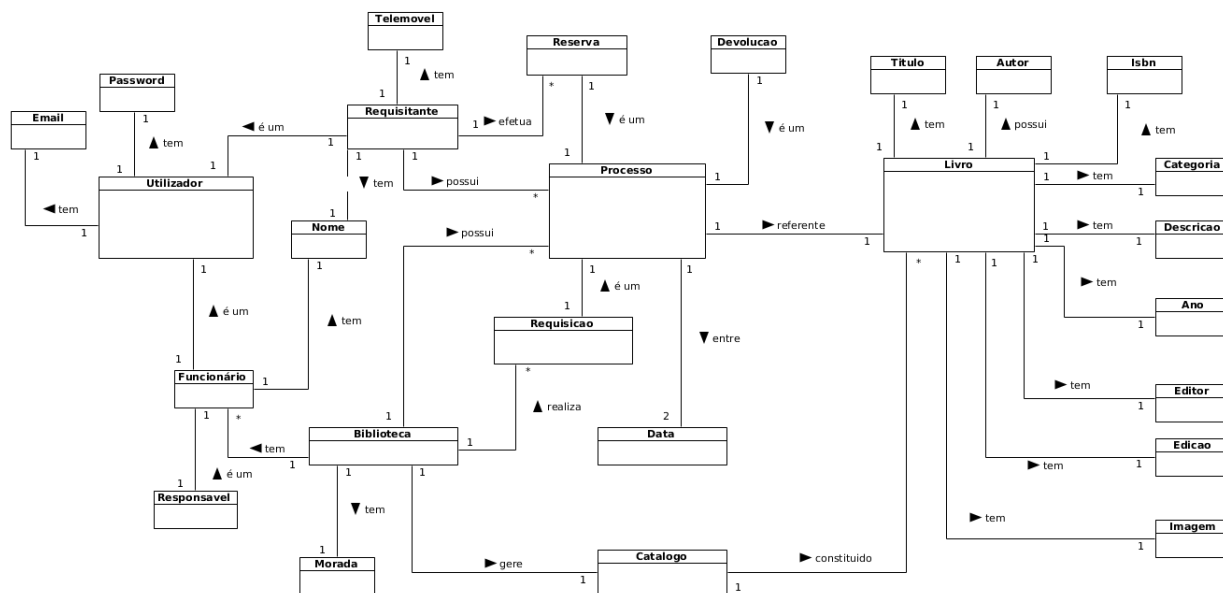


Figura 2. Modelo de domínio

3.2 Diagrama de Use Cases

Através dos requisitos funcionais desenvolvemos um diagrama dos casos de uso, que permite ter uma melhor perspetiva do que cada ator irá poder fazer na aplicação e das funcionalidades que a aplicação terá de disponibilizar.

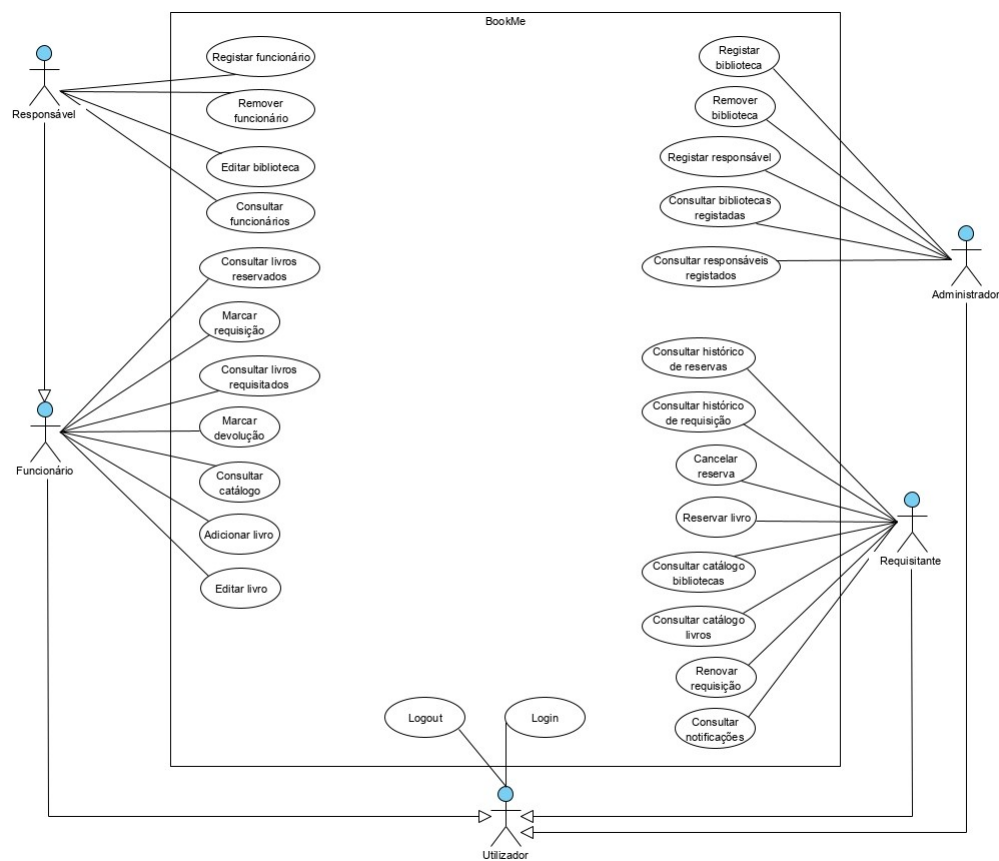


Figura 3. Diagrama de Use Cases

3.3 Modelo Independente da Plataforma (PIM)

Com base nas informações recolhidas até ao momento, desenvolvemos o PIM onde é possível observar a lógica de negócio do sistema. Este modelo é independente do tipo de tecnologia utilizado na fase de desenvolvimento.

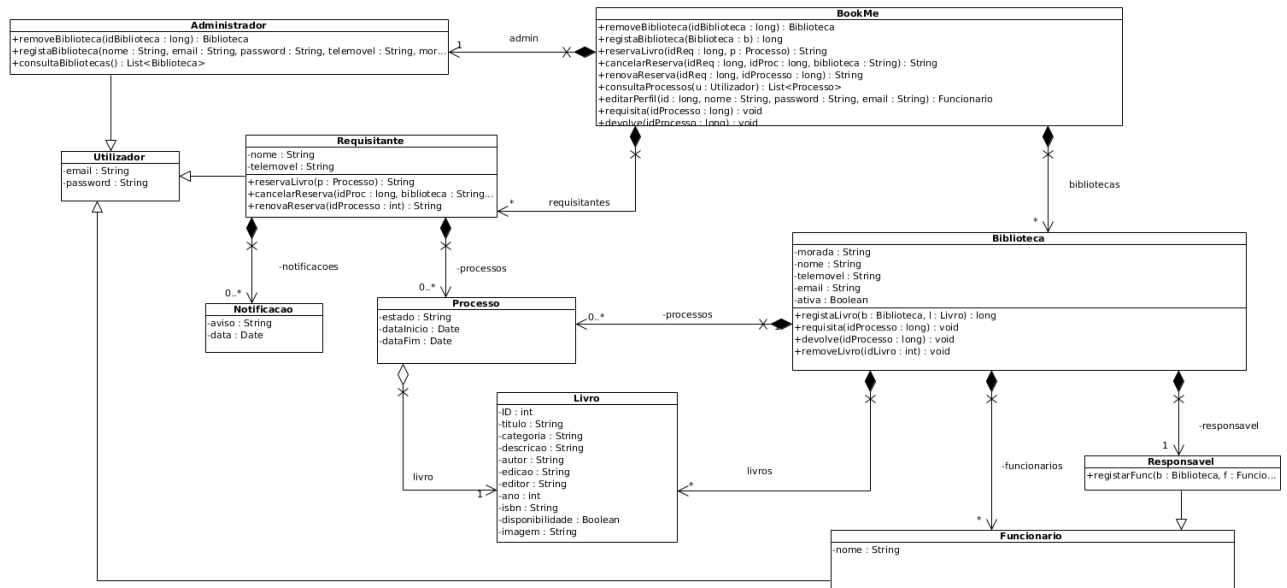


Figura 4. Modelo Independente da Plataforma

Tendo em conta o diagrama apresentado na figura 4, de seguida explicamos algumas das classes e das respetivas relações.

- **BookMe** - Esta classe é composta por todas as bibliotecas, os requisitantes e o administrador. A BookMe possui todos métodos especificados no diagrama de Use Cases apresentado anteriormente.

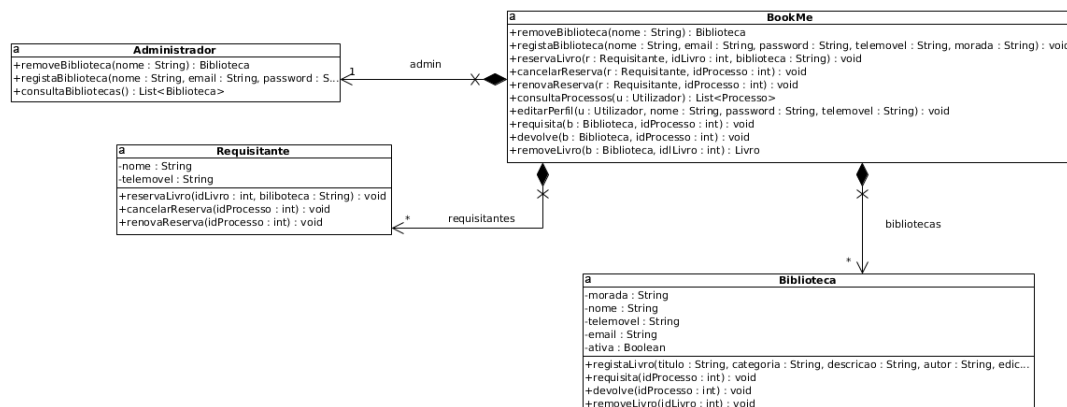


Figura 5. Diagrama de classes das relações com a classe BookMe

- **Utilizador** - esta classe retrata os diferentes utilizadores que interagem com o sistema. Como podemos observar na imagem apresentada de seguida, identificamos quatro tipos de utilizadores, nomeadamente:

- **Requisitante:** utilizador comum, que utiliza a aplicação com o intuito de reservar de livros.
- **Funcionário:** representa os trabalhadores das diferentes bibliotecas e que são responsáveis, essencialmente, pela gestão de catálogos, requisições e devoluções.
- **Responsável:** é um funcionário com responsabilidades acrescidas, nomeadamente o registo de funcionários no sistema.
- **Administrador:** responsável pela gestão das bibliotecas existentes no sistema, bem como dos respetivos responsáveis.

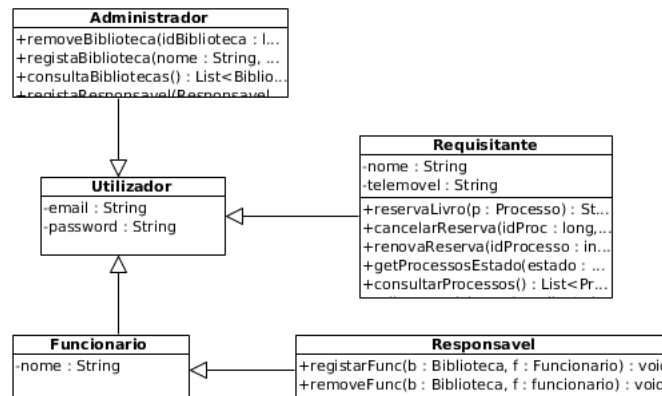


Figura 6. Diagrama de classes das relações com a classe Utilizador

- **Processo** - esta classe contém toda a informação necessária para efetuar o registo da reserva de um livro, incluindo um atributo estado, que indica se o livro está reservado, se já foi requisitado ou devolvido. Como podemos verificar de seguida, o processo para além do atributo estado, tem as datas de início e fim associadas, bem como o livro a que se refere. Por sua vez, tanto os requisitantes como as bibliotecas contêm os seus próprios processos.

3.4 Modelo Específico da Plataforma (PSM)

Uma vez definido o diagrama de classes, optamos por desenvolver mais dois diagramas que permitem o refinamento do PIM apresentado anteriormente. Como para a camada de dados vamos utilizar o *Hibernate*, começamos, por anotar as classes a persistir com *ORM Persistable*, como se pode observar na figura apresentada de seguida.

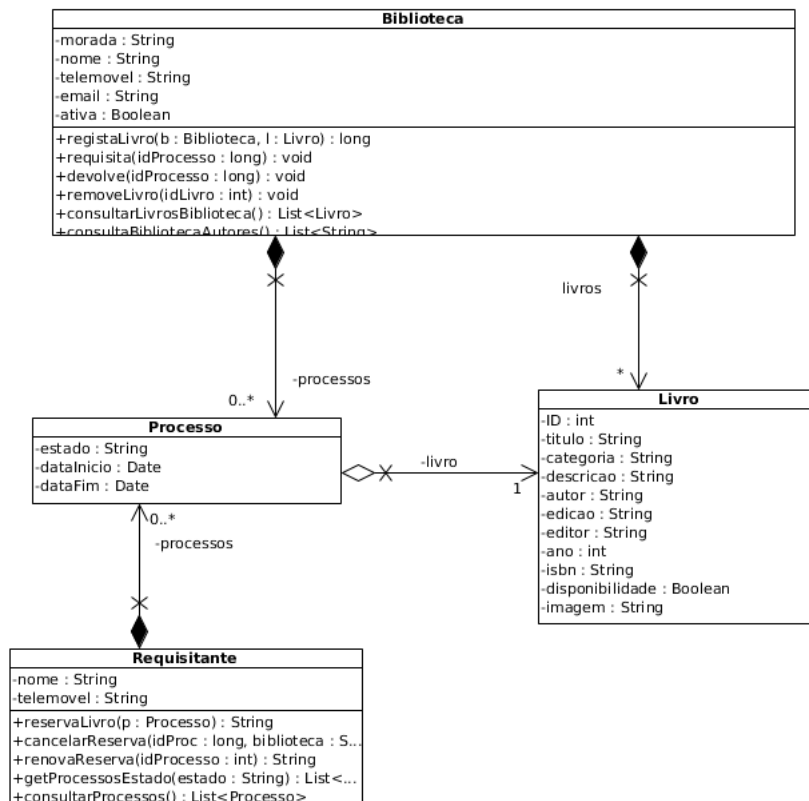


Figura 7. Diagrama de classes das relações com a classe Processo

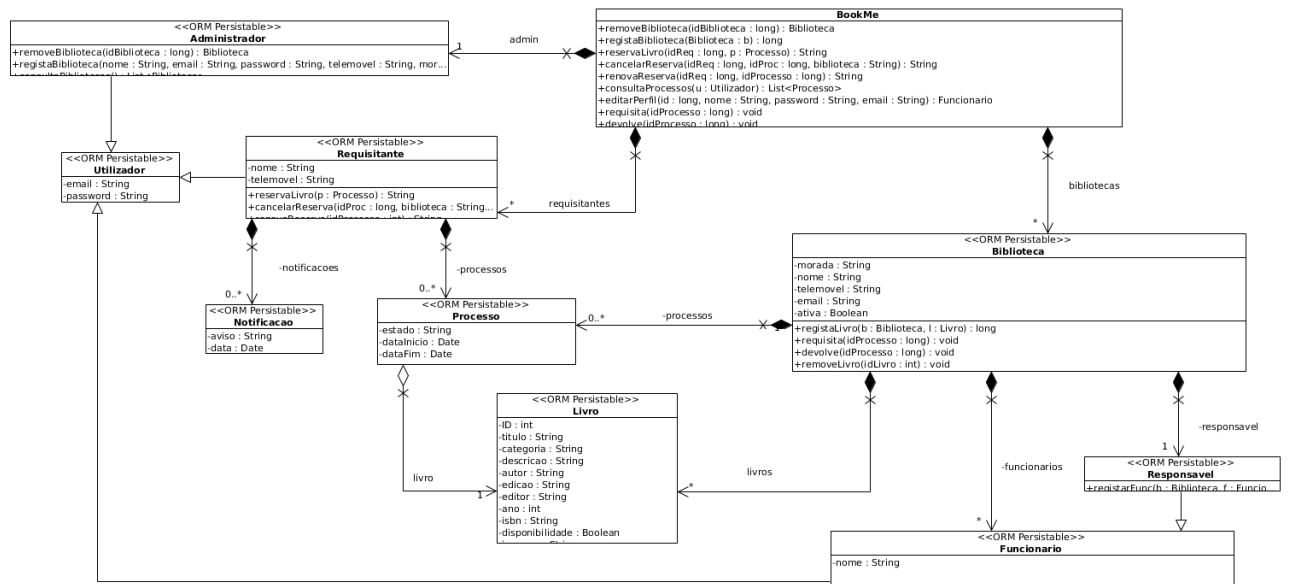


Figura 8. Modelo Específico da Plataforma

Já para a camada de negócio optamos pelo *Spring*, pelo que foi essencial definir

os nossos *Stateless Session Spring Beans* para aproveitar o *Spring Container*. Desta forma, definimos três tipos de *Spring Beans*, nomeadamente:

- **RequisitanteBean** - responsável pela consulta de livros e gestão de reservas.
- **AdministradorBean** - responsável pela gestão das bibliotecas existentes no sistemas e dos seus respetivos responsáveis.
- **BibliotecaBean** - responsável por tratar da gestão dos seus livros e funcionários, bem como das devoluções e requisições.
- **FuncionarioBean** - responsável pela edição do seu perfil.
- **ReponsavelBean** - responsável pela gestão dos funcionários da biblioteca.

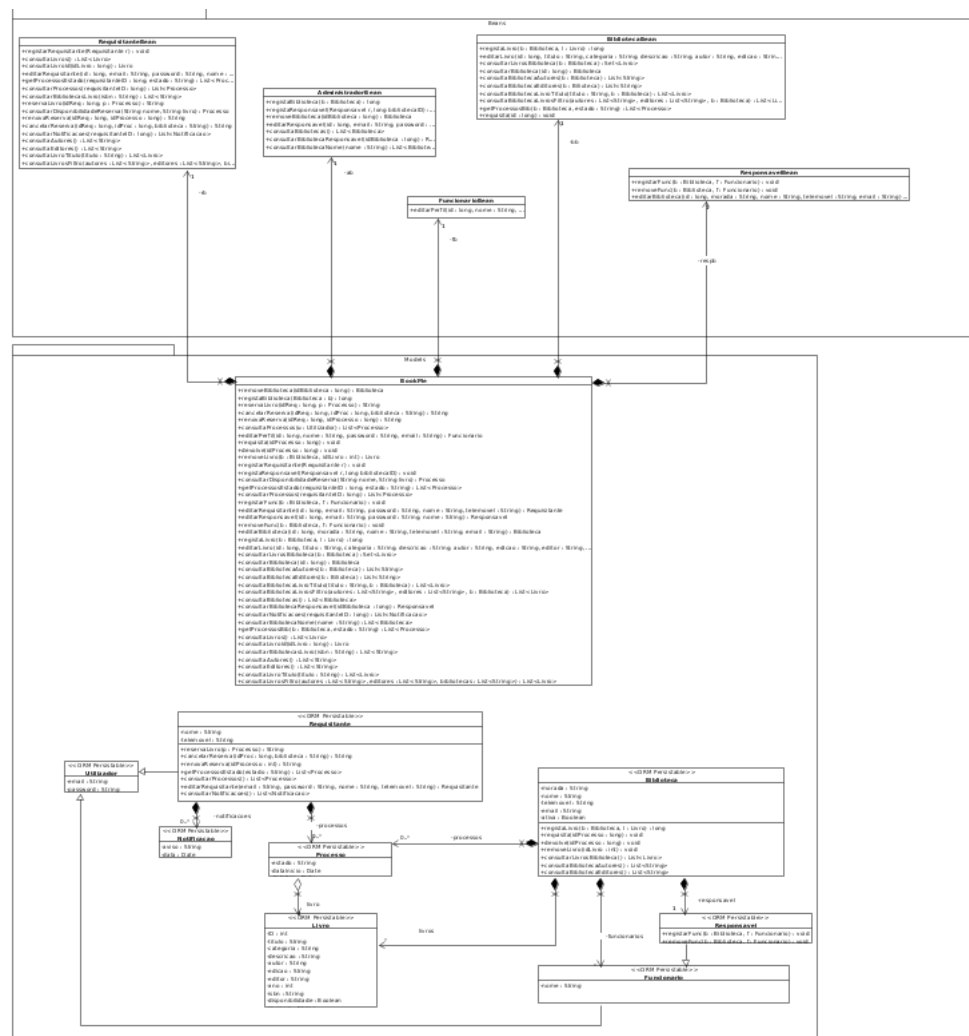


Figura 9. Modelo Específico da Plataforma com base nas frameworks utilizadas

4 Análise de Tarefas

De forma a analisar as principais tarefas suportadas pelo nosso sistema, optamos por elaborar diagramas HTA *Hierarchical Task Analysis*. Sendo o objetivo principal da aplicação a reserva de livros online optamos por desenvolver para a tarefa de efetuar uma reserva e para a marcação de um livro como requisitado que é efetuado pelo funcionário de uma biblioteca. Como a gestão dos livros do sistema também é essencial optamos por fazer também para o registar livro. Assim sendo, de seguido é possível observar cada um dos diagramas desenvolvidos.



Plano 0: Efetuar 1, 2, 3 e 4 por esta ordem

Plano 1: Efetuar 1.1, 1.2(opcional), 1.3 por esta ordem

Figura 10. Tarefa reservar livro realizada por um requisitante



Plano 0: Efetuar 1, 2, 3 e 4 por esta ordem

Plano 3: Efetuar 3.1 e 3.2 por esta ordem

Figura 11. Tarefa registar livro realizada por um funcionário da biblioteca



Plano 0: Efetuar 1, 2 e 3 por esta ordem

Plano 1: Efetuar 1.1, 1.2(opcional) e 1.3 por esta ordem

Figura 12. Tarefa requisitar livro realizada por um funcionário da biblioteca

5 Prototipagem

5.1 Mockups da aplicação

Com o intuito de tomar decisões sobre a forma como a interface iria ser desenvolvida, optamos pela elaboração de mockups numa fase inicial do problema.

A aplicação BookMe pode ser utilizada tanto pelos requisitantes como pelos funcionários das várias bibliotecas existentes no sistema pelo que de seguida, a título exemplificativo, apresentamos alguns dos mockups desenvolvidos para o perfil de requisitante.

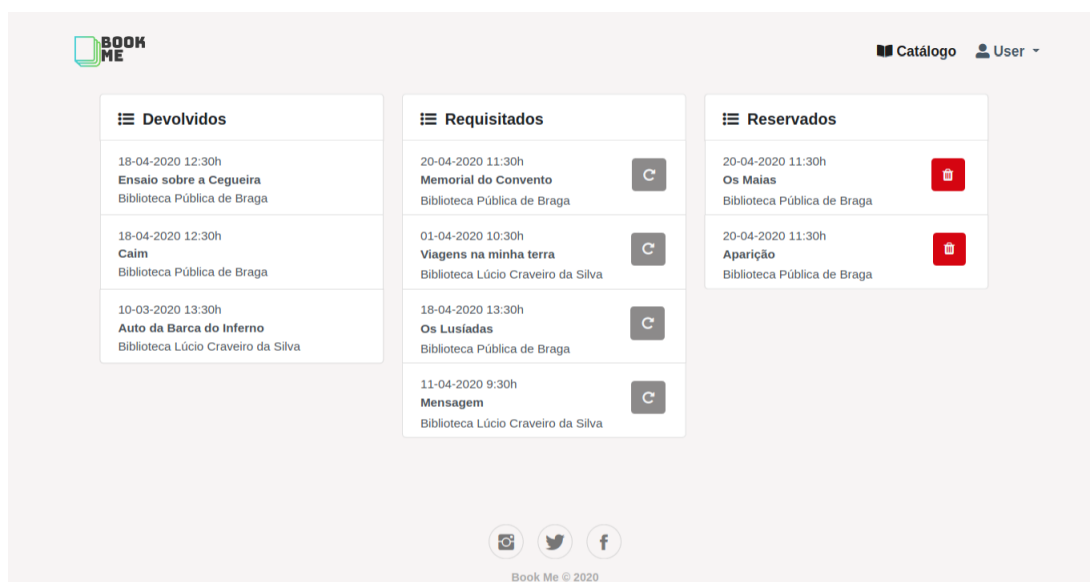


Figura 14. Consultar o histórico de devoluções, requisições e reservas

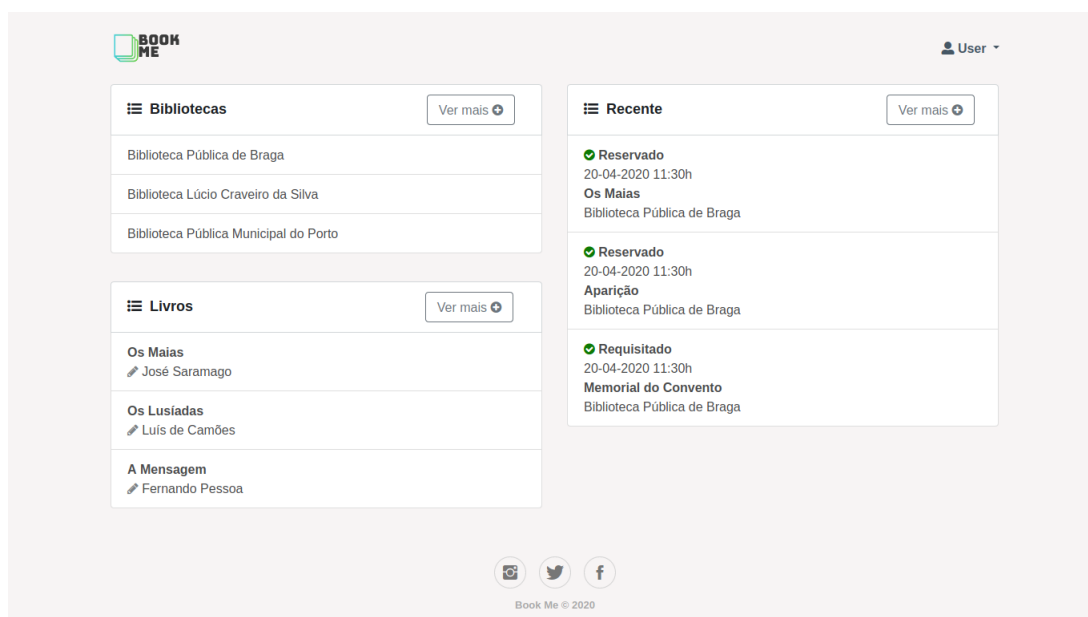


Figura 13. Página inicial do Requisiteante

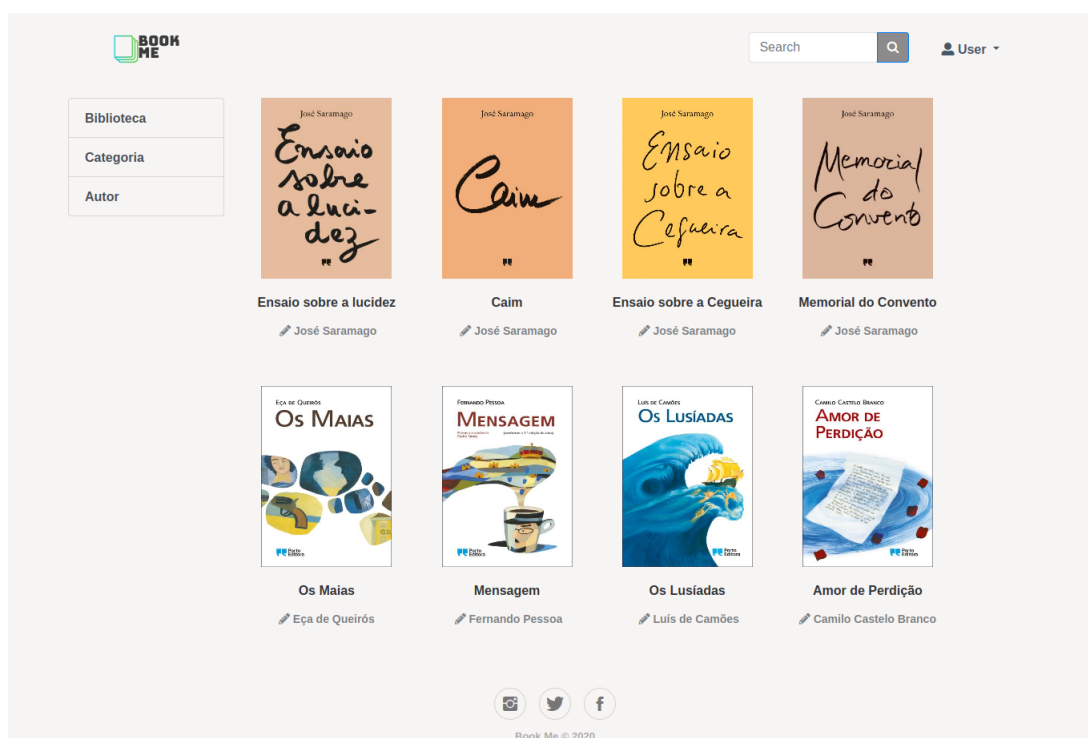


Figura 15. Consultar catálogo de livros



Figura 16. Consultar livro e/ou efetuar reserva

5.2 Mapa de Navegação

Assim sendo o mapa de navegação desenvolvido para a nossa plataforma é o apresentado de seguida, no qual podemos ter uma visão global do comportamento da interface para os diferentes atores da aplicação.

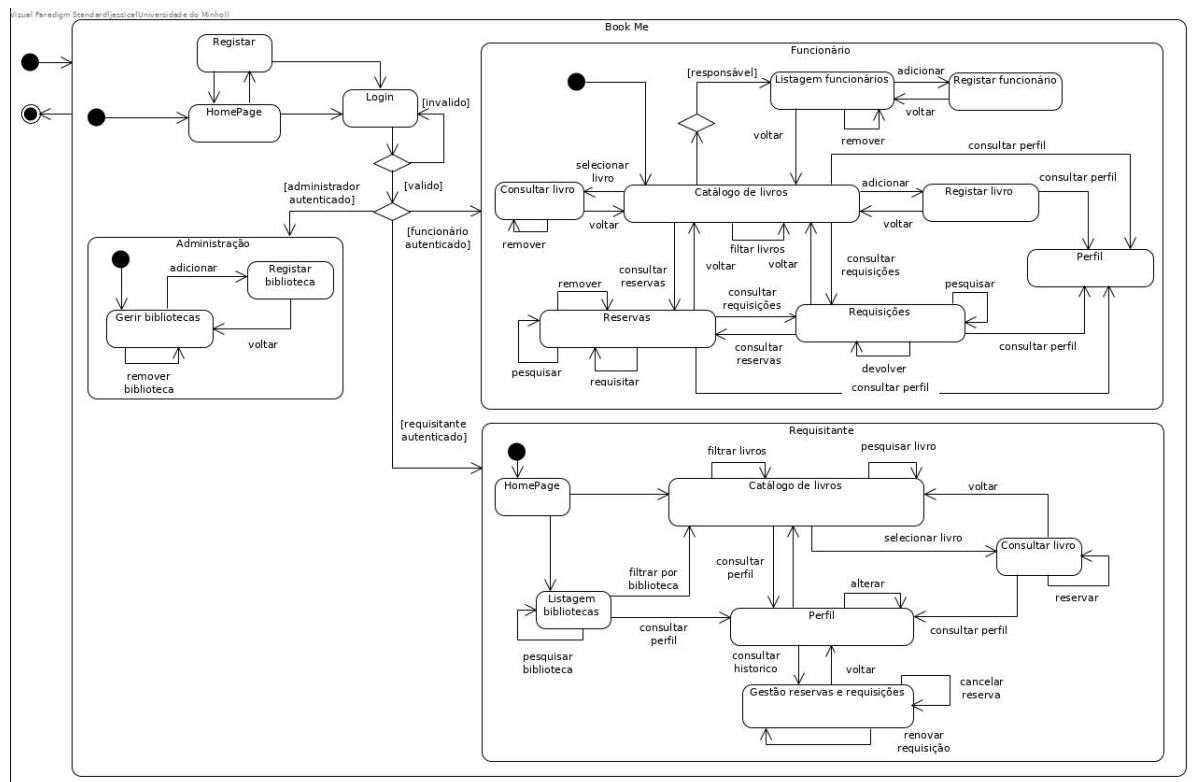


Figura 17. Mapa de navegação da aplicação

6 Implementação

6.1 Arquitetura

A arquitetura final não divergiu da arquitetura proposta inicialmente, sendo que as tecnologias e *frameworks* utilizadas no projeto foram as apresentadas de seguida.

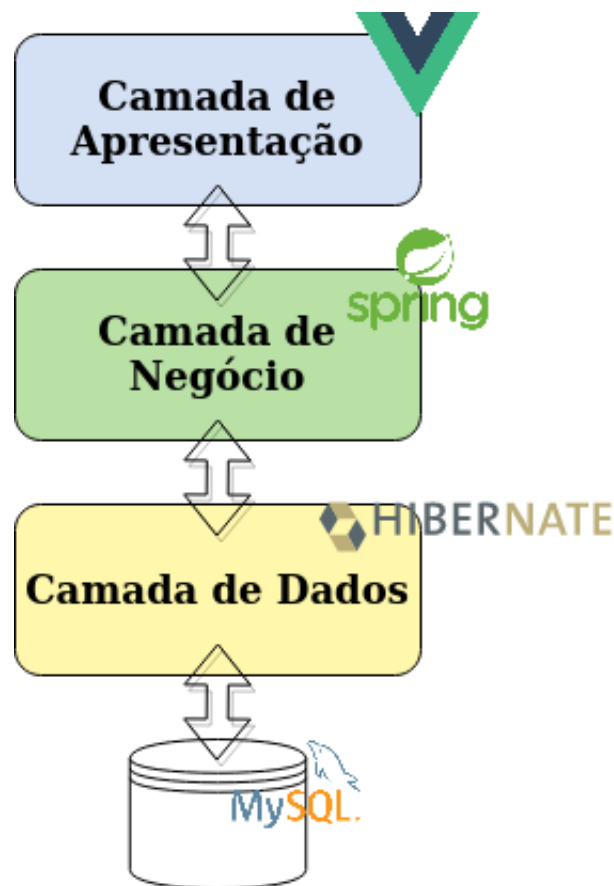


Figura 18. Arquitetura Geral

6.2 Camada de dados

Para implementar a camada de dados decidimos recorrer ao *Hibernate*. Assim sendo, a partir do PSM apresentado anteriormente foi possível gerar as classes e os respetivos métodos. De seguida, recorreremos a *Anotações Hibernate* para especificar a camada de dados. Desta forma, anotamos com *@Entity* as classes a persistir e especificamos o respetivo id com *@ID*. Para além disso, foi necessário especificar o mecanismo de herança a utilizar para o *Utilizador*. A estratégia adotada passou por uma relação por classe concreta, *InheritanceType.TABLE_PER_CLASS*. É importante salientar que o motor de gestão de base de dados utilizado foi o *MySQL*.

6.3 Camada de negócio

Relativamente à camada de negócio, optamos por utilizar o *Spring*. A integração do mesmo com o *Hibernate* foi feita a partir do módulo *Spring Data JPA*. Para a au-

tenticação dos utilizadores, recorreremos ao módulo *Spring Security* que nos permitiu tirar partido de mecanismos como *JWT Tokens*. Para a implementar a lógica da aplicação, especificamos as nossas classes de serviço como *Stateless Session Spring Beans*, a partir das quais foi possível disponibilizar a nossa API recorrendo à *BookMe*, a nossa facade. Para definir os *Controller RESTful* aos quais o *front-end* deve aceder, tiramos partido do *@RestController* do *Spring*. De referir que os pedidos GET e POST são especificados por *GetMapping* e *PostMapping* respetivamente. Nós optamos por definir cinco controllers, nomeadamente:

- **AdministradorController** - Disponibiliza as funcionalidades associadas ao administrador, como a gestão das bibliotecas do sistema e dos respetivos responsáveis.
- **RequisitanteController** - Fornece as funções associadas ao requisitante, nomeadamente a gestão das reservas (reserva, renovação, e cancelamento), bem como as consultas de histórico, livros e bibliotecas.
- **BibliotecaController** - Disponibiliza as funcionalidades associada à biblioteca, nomeadamente a gestão dos livros, das requisições e devoluções.
- **FuncionarioController** - Dispõe as funcionalidades relativas ao funcionário, nomeadamente a gestão dos seus dados.
- **ResponsavelController** - Disponibiliza as funcionalidades extra de um responsável, como a gestão dos funcionários e a edição do perfil da biblioteca.
- **LoginController** - Disponibiliza o login.

6.4 Camada de apresentação

Para esta camada decidimos utilizar *Vue.js*, tendo em conta que esta é uma *framework* muito utilizada em aplicações web e também simples de aprender e desenvolver. Para além disso, esta está munida de uma vasta documentação e várias *frameworks* e bibliotecas que fornecem componentes que podem ser reutilizados ao longo da aplicação, tal como o *Bootstrap Vue*.

A integração com o servidor de backend por *REST* foi feita recorrendo ao *axios*, uma dependência que permite a realização de pedidos HTTP.

Na implementação do *client-side* existiram basicamente 2 fases:

- Implementação da interface, ou seja, da estrutura da interface e do estilo de forma estática recorrendo a dados fictícios
- Integração dos dados, em que se implementou os métodos para a integração com o backend e se efetuou as alterações necessários com os dados recebidos

7 Interface Web

Um dos aspetos tido em conta no desenvolvimento da interface foi que esta deveria-se adaptar aos vários contextos e aos utilizadores, pelo que os ecrãs desenvolvidos são responsivos, adaptando-se a vários tamanhos de ecrãs.

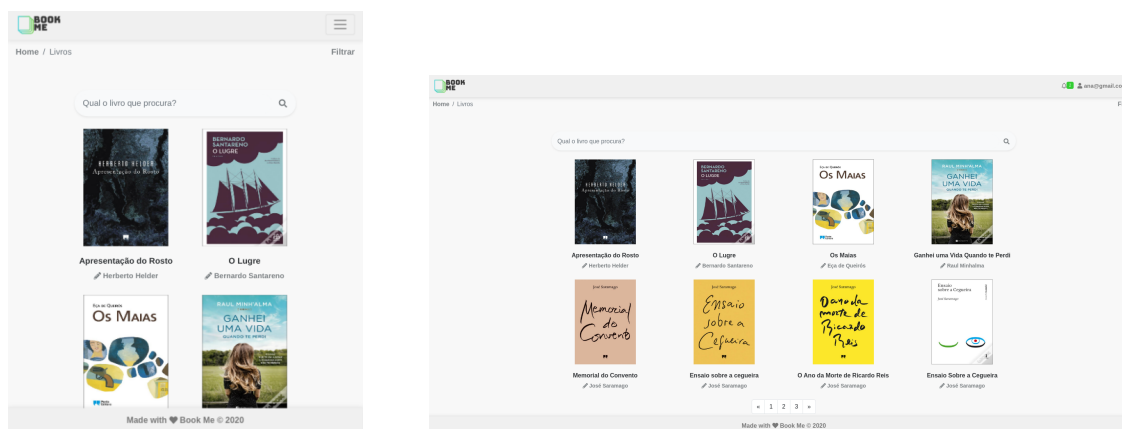


Figura 19. Exemplo de responsividade da aplicação

Para além disso, ao longo do desenvolvimento procuramos ter em conta as diversas heurísticas que passamos a demonstrar.

De modo a fornecer atalhos e saídas ao utilizador, recorreremos à navbar que assim permite a navegação pelas várias páginas bem como permitir o acesso em todas as páginas relacionadas com o utilizador, como por exemplo no caso do funcionário e requisitante o acesso ao consultar/editar perfil. Ainda utilizamos o padrão *breadcrumb* com o intuito de o utilizador perceber o percurso que realizou e desta forma voltar a pontos anteriores.

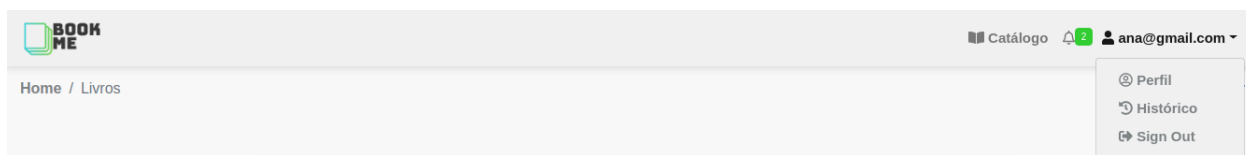


Figura 20. Exemplo de navegação e de fornecer informação da aplicação

Com o intuito de facilitar a procura do utilizador quer de livros ou bibliotecas, mas também dos processos de reservas e requisições decidimos colocar barras de *search* bem como filtros no caso da consulta dos catálogos.

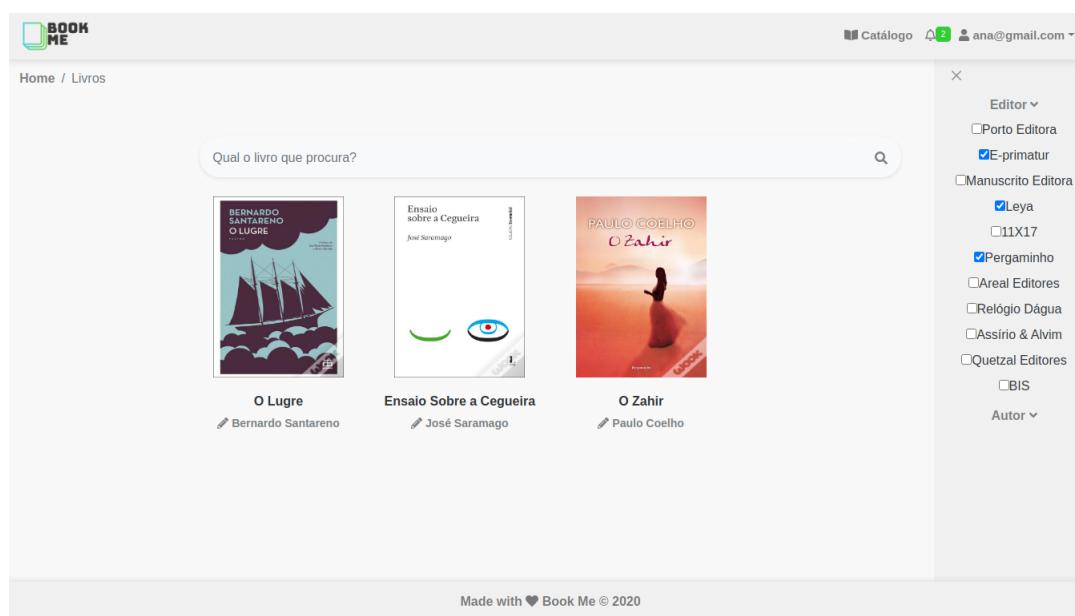
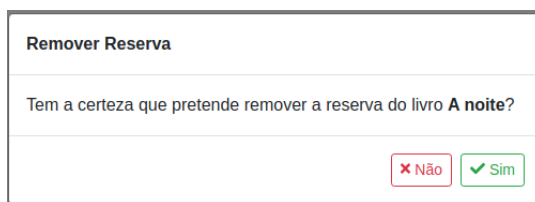


Figura 21. Exemplo de search e de filtros

Em casos, como eliminar funcionários ou bibliotecas nas interfaces do administrador e do funcionários responsável, e efetuar requisições ou devolução na interface do funcionário, bem como renovar e eliminar reservas pelos requisitantes decidimos pedir a confirmação da operação com o intuito de que não ocorram erros por parte dos utilizadores. Para tal, recorreremos aos *Modal* que confirma se o utilizador pretende avançar.



A dialog box titled "Remover Reserva". It contains the text "Tem a certeza que pretende remover a reserva do livro A noite?". At the bottom right, there are two buttons: "Não" (No) with a red 'X' icon and "Sim" (Yes) with a green checkmark icon.

Figura 22. Confirmação de remoção/renovações

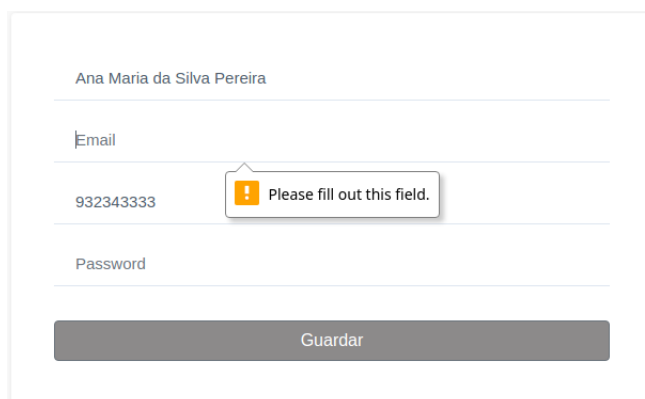
Para além disto, tivemos o cuidado de fornecer *feedback* ao utilizador sobre as suas ações. Assim, através de duas formas, com *Alerts* e *Toasts* mostramos o sucesso ou não de uma dada ação.



Two toast messages are shown side-by-side. The left one is green and titled "Sucesso" (Success), with the text "Reserva cancelada com sucesso" (Reservation canceled successfully). The right one is red and titled "Error", with the text "Não é possível renovar este livro" (It is not possible to renew this book). Both have a close button (X) in the top right corner.

Figura 23. Exemplo de responsividade da aplicação

No preenchimento ou edição de formulários, de modo a evitar erros não é permitido a submissão dos mesmos sem todas as informações preenchidas tal como podemos verificado de seguida, sendo pedido ao utilizador que preencha esses campos.



A form with the following fields: "Ana Maria da Silva Pereira" (text), "Email" (text), "932343333" (text), and "Password" (text). A validation error message "Please fill out this field." with an orange exclamation mark icon is shown above the "932343333" field. At the bottom, there is a "Guardar" (Save) button.

Figura 24. Impedir a submissão de formulários com campos vazios

Uma vez que é permitida a reserva imediata de um livro mas também quando já existe uma reserva do mesmo, tivemos o cuidado de informar o utilizador acerca da disponibilidade do mesmo, sendo indicada uma previsão do levantamento no último caso. Desta forma, inicialmente é indicado como é o funcionamento da aplicação.

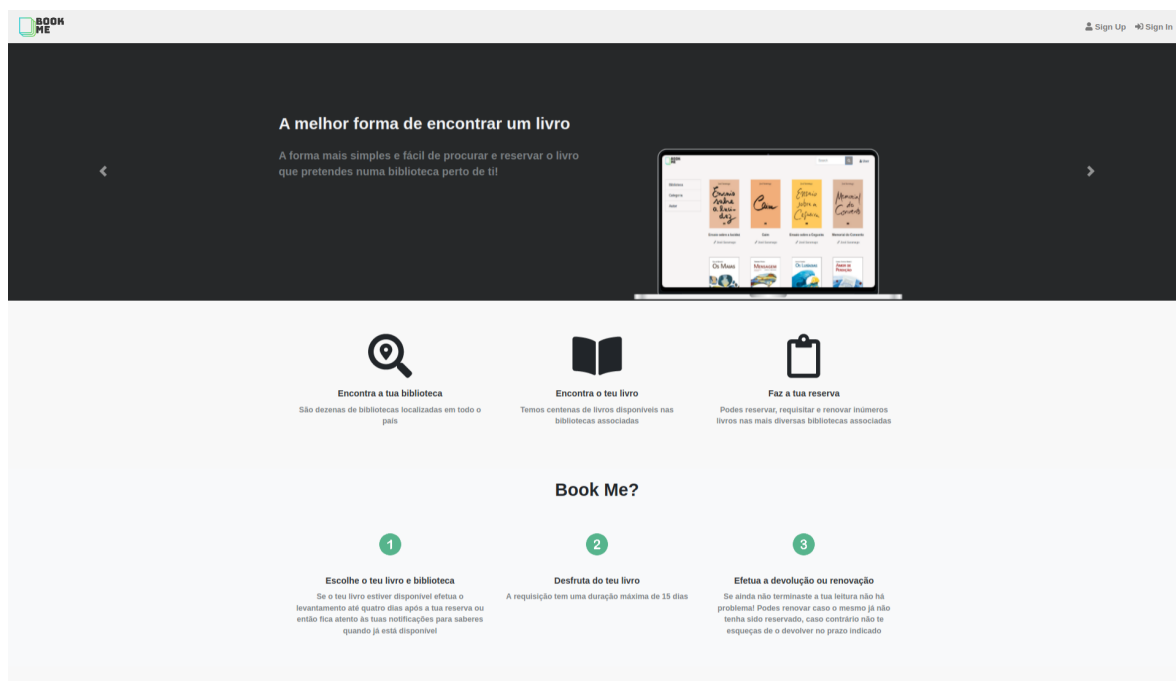


Figura 25. HomePage com a explicação do funcionamento

Contudo, de modo ao utilizador tomar uma decisão informada recorreremos ao *Tool-tips* para indicar processo que decorrerá após a reserva. Contudo, optamos também por dar a possibilidade de remover reservas efetuadas na página de histórico dos mesmos.



Figura 26. Exemplo de prevenir erros

Desta forma, procuramos em toda a aplicação manter uma consistência, com uma linguagem simples, adequada e natural. Fornecemos atalhos e saídas assinaladas de forma clara bem como mensagens de erro claras e principalmente prevenir que erros acontecessem. Com o intuito de minimizar a carga de memória e também fornecer uma forma cómoda de consulta utilizamos também paginação tanto nos catálogos, como nas listagens de bibliotecas, reservas e requisições.

Passamos agora a demonstrar uma tarefa delineada anteriormente, sendo apresentado de seguida o diagrama de tarefa, correspondente à reserva de um livro.

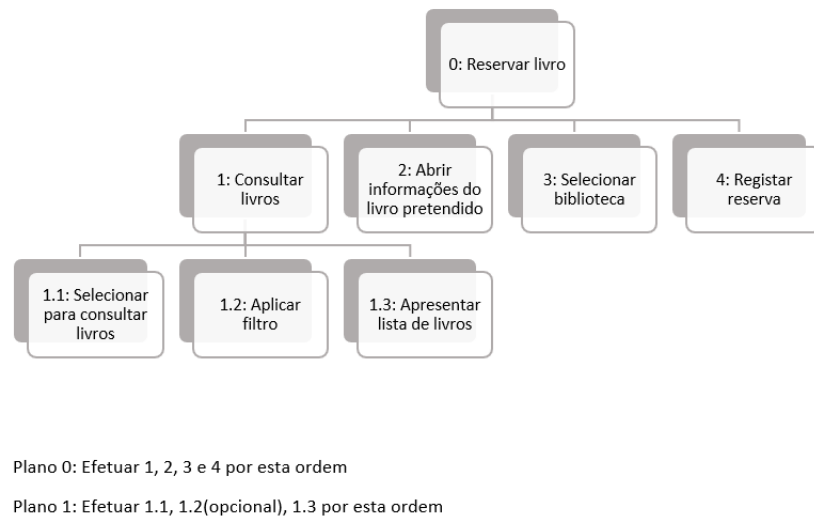


Figura 27. Tarefa reservar livro realizada por um requisitante

Após o requisitante efetuar o login é-lhe apresentada a sua *HomePage* na qual o mesmo pode escolher visualizar o catálogo de livros, clicando em ver mais no exemplo de livros existentes na aplicação, ou através da opção existente na navbar. Desta forma, é-lhe apresentada a listagem de livros existentes podendo procurar pelo livro que pretendo ou efetuar a filtragem na opção de filtrar existente podendo selecionar os filtros que pretender. Desta forma, observando os livros que estão de acordo com as preferências do utilizador este seleciona o que pretende. Selecionado o livro, é então apresentado toda a informação desse livro podendo selecionar a biblioteca da lista de bibliotecas apresentada sendo-lhe indicado a disponibilidade desse livro. Caso o utilizador pretenda o reservar então pressiona o botão reservar aparecendo informação referente ao sucesso da operação.

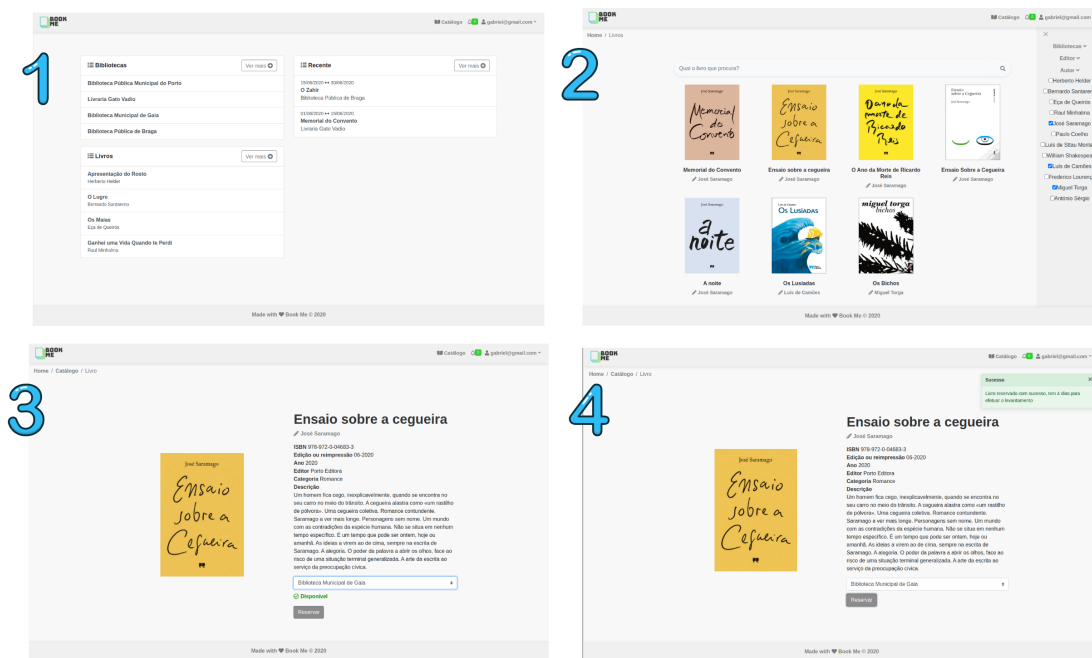


Figura 28. Reservar um livro na aplicação

8 Deployment

8.1 Docker

Para fazer o *deployment* da aplicação fizemos uso do *Docker*, que permite criar consistência entre os ambientes de desenvolvimento e produção, através da criação de *containers* que são específicos às necessidades da aplicação.

Para cada *container* podem ser definidos o software a instalar e configurações do mesmo, através de um *Dockerfile*. Assim sendo, para o projeto são necessários 3 *containers*: *frontend*, *backend* e base de dados.

Para o *frontend* fazemos uso de uma imagem base que contém *node* e *npm*, e indicamos quais as instalações necessárias e iniciamos o servidor:

```
FROM node:12
```

```
WORKDIR /app
```

```
ENV PATH /app/node_modules/.bin:$PATH
```

```
COPY package.json /app/package.json
```

```
RUN npm install
RUN npm install @vue/cli@4.4.6 -g
```

```
CMD ["npm", "run", "serve"]
```

Para o *backend* tivemos por base uma imagem *ubuntu*, e fazemos a instalação do software necessário, *Java* e *Maven*, para o *deploy* do *backend*:

```
FROM ubuntu:18.04

WORKDIR /server

COPY . .
COPY ./wait-for-it.sh .
RUN chmod +x wait-for-it.sh

RUN apt-get update
RUN apt-get install openjdk-8-jdk-headless -y
RUN apt-get install openjdk-8-jre-headless -y
RUN apt-get install maven -y
RUN apt-get install mysql-client -y
```

Em relação à base de dados não foi necessário criar um *Dockerfile*, pois criámos o mesmo no *docker-compose*, devido à simplicidade do mesmo, fazendo uso de uma imagem já existente.

8.2 Docker-Compose

Com os *containers* definidos, falta agora definir como estes interagem entre si. Para tal fazemos uso do *docker-compose*, que permite definir os vários serviços que serão inicializados, a partir de que imagens e configurar as interações entre os múltiplos *containers*.

Sendo assim no ficheiro *docker-compose.yaml*, definimos os serviços que serão criados: *frontend*, *backend* e base de dados. Para cada serviço associámos o respetivo *Dockerfile*, definido na secção anterior, que serve de imagem base para o serviço, e indicámos as variáveis de ambiente de cada imagem, como as configurações da base de

dados e as configurações do *backend* para aceder à base de dados. Neste ficheiro também definimos as diferentes portas para cada serviço para comunicarem com o *host* e entre si. Temos assim o *docker-compose* definido:

```
version: '3.3'
```

```
services:
```

```
  bookme_db:
```

```
    image: mysql:5.7.26
```

```
    environment:
```

```
      MYSQL_DATABASE: bookme
```

```
      MYSQL_ROOT_PASSWORD: password
```

```
    volumes:
```

```
      - './populacao.sql:/scripts/populacao.sql'
```

```
    ports:
```

```
      - 3300:3306
```

```
    networks:
```

```
      - network_backend
```

```
  frontend:
```

```
    build: ./client
```

```
    volumes:
```

```
      - './client:/app'
```

```
    ports:
```

```
      - 8081:8080
```

```
  backend:
```

```
    build:
```

```
      context: ./server
```

```
    command: ["/wait-for-it.sh", "bookme_db:3306", "--",  
             "mvn", "install"]
```

```
    command: ["mvn", "spring-boot:run"]
```

```
environment:
  WAIT_HOSTS: mysql:3306
  SPRING_DATASOURCE_URL: jdbc:mysql://bookme_db:3306/
  bookme?useUnicode=true&useJDBCCompliantTimezoneShift=true
  &useLegacyDatetimeCode=false&serverTimezone=Europe/Lisbon
  SPRING_DATASOURCE_USERNAME: root
  SPRING_DATASOURCE_PASSWORD: password
ports:
  - 9000:8080
networks:
  - network_backend
restart: always

networks:
  network_backend:
    driver: "bridge"
```

9 Testes de carga

Com o intuito de analisar a performance da nossa aplicação em relação à carga que consegue suportar realizamos testes recorrendo ao *JMeter*. Nesta fase optamos por definir dois testes, sendo estes a consulta do catálogo por parte dos requisitantes e a consulta de um livro do catálogo pelo mesmo utilizador. Assim sendo, para o testes efetuado os utilizadores efetuam o login, consultam o catálogo de livros e selecionam um para consultar.

Começamos por efetuar os testes para 50, 100, 150 e 170 requisitantes simultâneos sendo os resultados apresentados de seguida.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received ...	Sent KB/s...
http://localhost:9000/login	50	1616	1607	2025	2040	2115	1119	2115	0.00%	23.5/sec	22.65	8.44
http://localhost:9000/livros/	50	226	219	267	269	286	168	286	0.00%	172.4/sec	3689.72	85.49
http://localhost:9000/consulta/bibliotecas	50	491	473	718	773	915	196	915	0.00%	52.5/sec	112.51	26.68
http://localhost:9000/Autores	50	227	213	416	421	480	8	480	0.00%	70.8/sec	47.86	35.12
http://localhost:9000/Editores	50	70	62	111	131	399	6	399	0.00%	75.5/sec	44.85	37.53
http://localhost:9000/livro/l	50	59	50	107	145	193	6	193	0.00%	183.8/sec	315.05	91.15
http://localhost:9000/bibliotecas/livro/97...	50	33	32	63	72	95	5	95	0.00%	359.7/sec	258.54	188.20
Test	50	2725	2735	3115	3146	3196	2196	3196	0.00%	15.3/sec	431.09	51.58
TOTAL	400	681	219	2427	2786	3124	5	3196	0.00%	122.2/sec	862.18	103.15

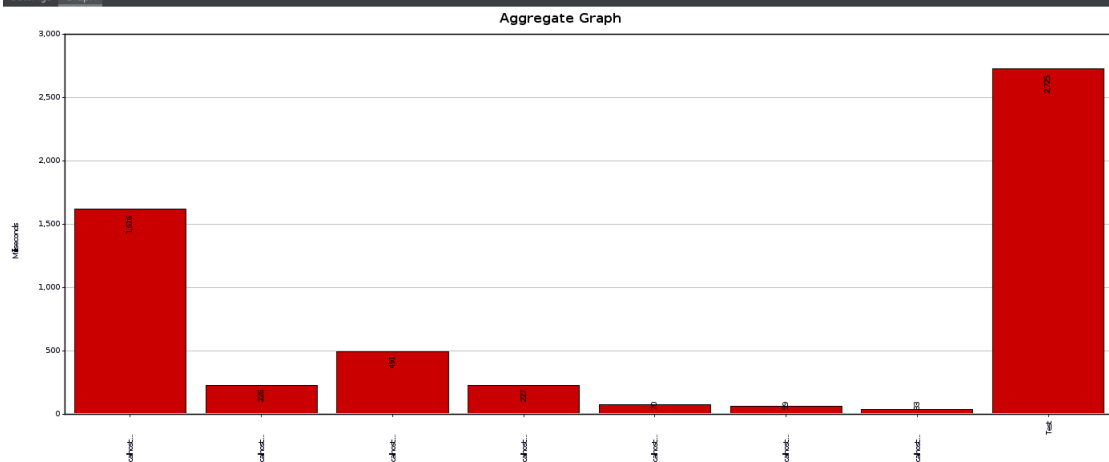


Figura 29. Resultados para 50 utilizadores simultâneos

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Receive...	Sent KB...
http://localhost:9000/login	100	1469	1488	2234	2375	2475	202	2743	0.00%	29.5/sec	28.45	10.60
http://localhost:9000/livros/	100	335	236	779	1027	1380	9	1533	0.00%	28.9/sec	618.87	14.33
http://localhost:9000/consulta/bibliotecas	100	519	487	893	1093	1254	77	1459	0.00%	27.9/sec	59.94	14.21
http://localhost:9000/Autores	100	217	154	472	613	770	3	1090	0.00%	31.4/sec	21.25	15.59
http://localhost:9000/Editores	100	181	117	415	570	889	3	1062	0.00%	33.6/sec	19.98	16.71
http://localhost:9000/livro/l	100	160	116	364	495	1149	6	1169	0.00%	36.7/sec	62.96	18.21
http://localhost:9000/bibliotecas/livro/978...	100	114	106	213	364	479	2	784	0.00%	37.9/sec	27.22	19.80
Test	100	2999	3052	3348	3379	3537	1622	3655	0.00%	25.0/sec	705.78	84.40
TOTAL	800	749	266	2743	3150	3355	2	3655	0.00%	200.2/sec	1411.56	168.80

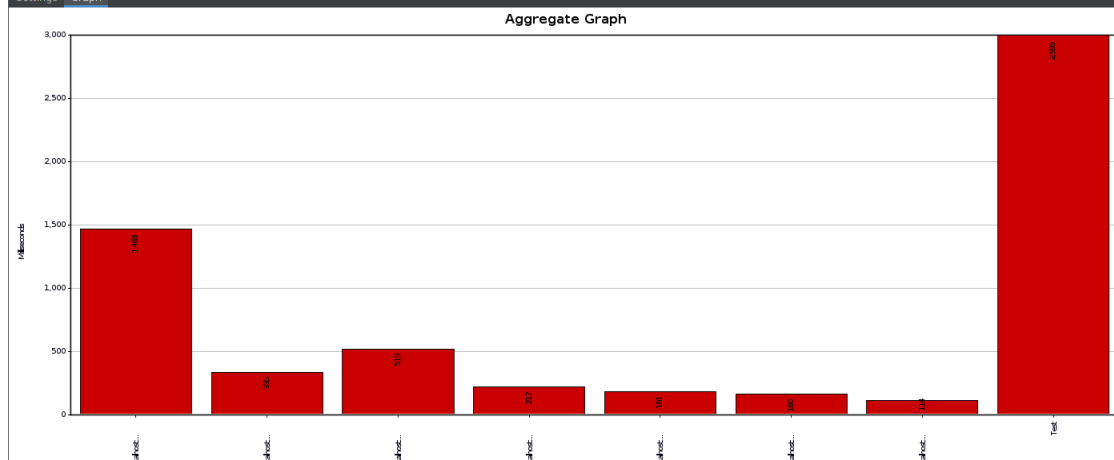


Figura 30. Resultados para 100 utilizadores simultâneos

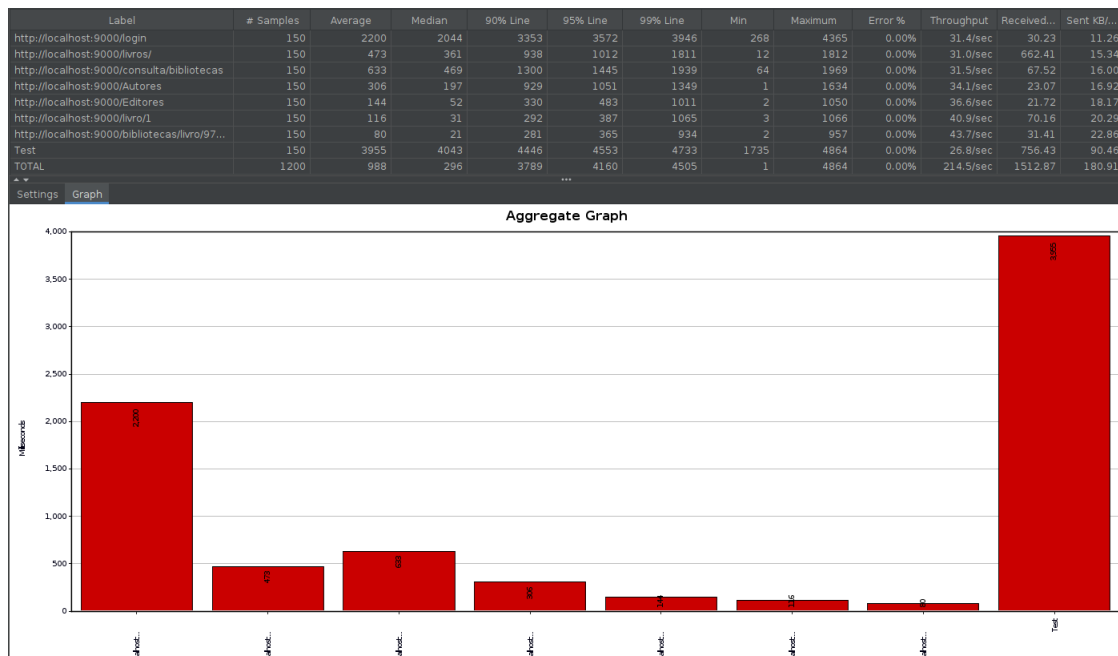


Figura 31. Resultados para 150 utilizadores simultâneos

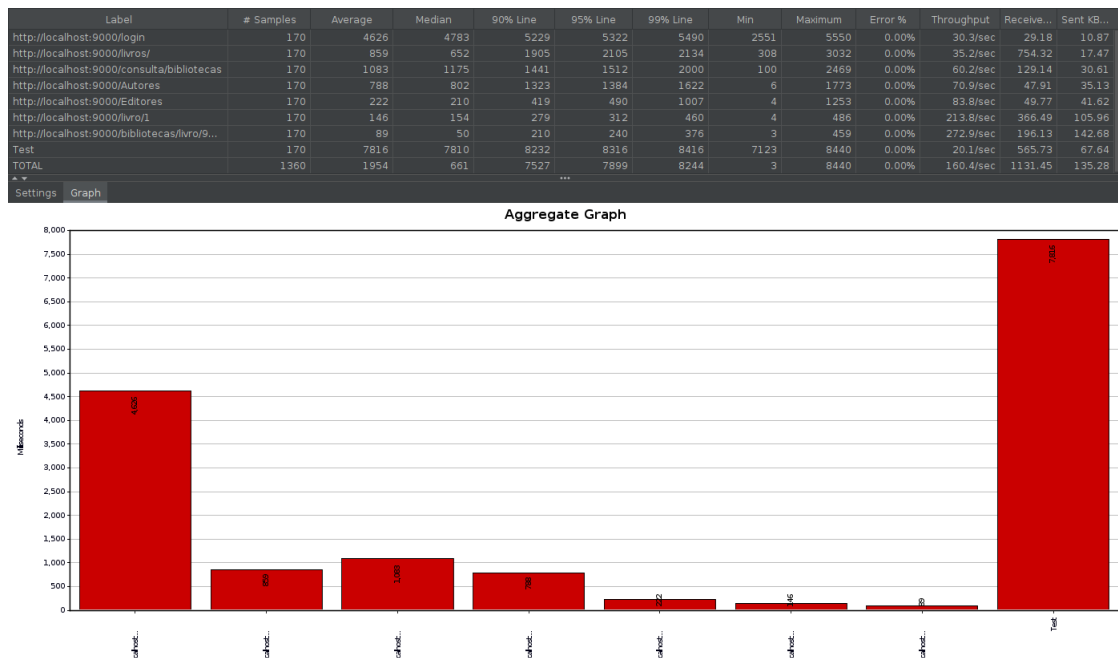


Figura 32. Resultados para 170 utilizadores simultâneos

Analisando os gráficos obtidos observamos que os tempos médios de resposta aumentam à medida que o número de utilizadores também aumentam, sendo que se destacou que o aumento do tempo médio dos 150 utilizadores para os 170 que mais do que

duplicou. Uma vez que não realizamos o balanceamento da carga por não termos conseguido efetuar o *deploy* em várias máquinas seria expectável que a aplicação tivesse maior tempo médio de resposta com um menor número de utilizadores.

10 Conclusão

Numa primeira fase do projeto começamos por realizar o levantamento de requisitos bem como a modelação da aplicação, sendo notório que esta aplicação teria uma dimensão considerável. Nesta etapa deparamo-nos com o primeiro problema que foi a criação de um PIM detalhado de modo a gerar o código através do *Visual Paradigm*. Uma grande vantagem desta fase para além da modelação de todo o projeto foi a prototipagem uma vez que conseguimos desenvolver mockups bastante semelhantes ao que pretendíamos para a aplicação sendo possível usar algum do código gerada nas fases posteriores.

Na segunda etapa começamos a implementação, em que o grupo não enfrentou grandes problemas sendo apenas necessário o tempo de habituação à framework Spring com a qual o grupo ainda não tinha tido contacto. Quanto à interface esta assemelhou-se bastante aos mockups sendo dispendido mais tempo em cuidados referentes à usabilidade, como o feedback aos utilizadores.

A terceira fase do projeto foi a que se tornou mais problemática tendo em conta que pretendíamos fazer o *deployment* em máquinas na *Google Cloud* contudo, problemas na implementação da comunicação entre máquinas/*containers* que impossibilitavam a ligação entre os vários serviços, fez com que apenas fosse possível a criação de um *docker-compose* da nossa estrutura numa só máquina, pelo que no futuro com o intuito de suportar mais carga poderíamos por exemplo criar um *container* intermédio com o *nginx* para realizar a distribuição de carga pelos vários servidores aplicacionais.

Na última etapa foram realizados os testes de carga em que consideramos que como trabalho futuro poderíamos realizar testes mais completos.

Em última instância, consideramos que os objetivos propostos foram cumpridos, sendo que existem pontos, tais como os referidos anteriormente, que poderiam ser realizados como trabalho futuro.