



ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO  
LICENCIATURA EM ENGENHARIA INFORMÁTICA  
DESENVOLVIMENTO DE APLICAÇÕES WEB

PROJETO - FASE DE IMPLEMENTAÇÃO  
BIBLIOTECA ESCOLAR DIGITAL - BOOKTALK

Inês Maria Pereira do Nascimento  
Rita Alexandra Lampreia Dias



Beja, janeiro de 2026

INSTITUTO POLITÉCNICO DE BEJA  
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO  
LICENCIATURA EM ENGENHARIA INFORMÁTICA  
DESENVOLVIMENTO DE APLICAÇÕES WEB

**PROJETO - FASE DE IMPLEMENTAÇÃO  
BIBLIOTECA ESCOLAR DIGITAL - BOOKTALK**

**Inês Maria Pereira do Nascimento - 24481  
Rita Alexandra Lampreia Dias - 23240**

**DOCENTE**

Professor Luís Carlos Bruno

Trabalho realizado no âmbito da unidade curricular Desenvolvimento de Aplicações Web

Beja, janeiro de 2026

# Índice

1	Introdução.....	7
2	Análise do Sistema.....	8
2.1	Caracterização dos atores .....	8
2.2	Diagrama de Casos de Uso .....	11
3	Desenho do Sistema.....	15
3.1	Modelação da Base Dados .....	15
3.1.1	Diagrama E/R .....	16
3.1.2	Modelo Lógico Relacional.....	17
3.2	Modelação de Interfaces Gráficas com o Utilizador.....	18
3.2.1	Storyboards .....	20
3.2.2	Wireframes .....	32
3.2.3	Wireframes .....	32
4	Melhorias efetuadas na análise e desenho do sistema.....	47
5	Implementação.....	48
5.1	Arquitetura do sistema.....	48
5.2	Tecnologias usadas .....	49
5.3	Desenvolvimento da API.....	51
5.3.1	Especificação da interface .....	51
5.3.2	Decisões de implementação .....	52
5.3.3	Principais casos relevantes de programação .....	53
5.4	Desenvolvimento da App frontend/MVC .....	56
5.4.1	Decisões de implementação .....	57
5.4.2	Principais casos relevantes de programação .....	59
6	Conclusão e Trabalho Futuro.....	65
	Referências Bibliográficas (normas APA).....	66

# Índice de Figuras

Figura 1 - Diagrama de Casos de Uso do Sistema BookTalk.....	11
Figura 2 - Modelo Entidade Relação do Sistema BookTalk.....	16
Figura 3 - Modelo Lógico Relacional do Sistema BookTalk .....	17
Figura 4 - Login com conta de bibliotecário .....	20
Figura 5 - Página inicial do bibliotecário.....	20
Figura 6 - Adicionar livros manualmente .....	21
Figura 7 - Página login.....	21
Figura 8 - Página inicial do leitor.....	22
Figura 9 - Página de detalhes de livro .....	22
Figura 10 - Página para agendar reserva.....	23
Figura 11 - Página para confirmar reserva.....	23
Figura 12 - Página de resumo de reserva .....	24
Figura 13 - Login como leitor .....	24
Figura 14 - Página inicial do leitor.....	25
Figura 15 - Pesquisar um livro .....	25
Figura 16 - Detalhes do livro .....	26
Figura 17 - Escrever review e avaliar o livro .....	26
Figura 18 - Review submetida.....	27
Figura 19 - Página login .....	28
Figura 20 - Página inicial bibliotecário.....	28
Figura 21 - Página reviews por aprovar.....	29
Figura 22 - Página review.....	29
Figura 23 - Página Estatísticas - Gráficos .....	30
Figura 24 - Página Estatísticas - Reservas.....	30
Figura 25 - Página Estatísticas - Reviews.....	31
Figura 26 - Login como bibliotecário .....	32
Figura 27 - Página inicial do bibliotecário.....	33
Figura 28 - Adicionar título .....	33
Figura 29 - Página de Login .....	34
Figura 30 - Página Inicial (Leitor).....	34
Figura 31 - Resultados da Pesquisa .....	35

Figura 32 - Página de detalhes do livro .....	35
Figura 33 - Página de Reserva com Calendário.....	36
Figura 34 - Confirmação da Reserva.....	36
Figura 35 - Resumo da Reserva .....	37
Figura 36 - Detalhes do Livro após Reserva .....	37
Figura 37 - Login como leitor .....	38
Figura 38 - Página inicial do leitor.....	38
Figura 39 - Pesquisa de um livro .....	39
Figura 40 - Detalhes do livro .....	39
Figura 41 - Escrever review e avaliar um livro.....	40
Figura 42 - Após submeter a review e a avaliação .....	40
Figura 43 - Página de Login .....	41
Figura 44 - Página Inicial da Conta Bibliotecário .....	41
Figura 45 - Acesso à secção "Reviews por aprovar".....	42
Figura 46 - Lista de "Reviews por aprovar" .....	42
Figura 47 - Moderar e Validar Review (popup da review selecionada) .....	43
Figura 48 - Confirmação da Aprovação de Review .....	43
Figura 49 - Resumo da ação .....	44
Figura 50 - Reviews por aprovar (após ter aprovado uma da lista) .....	44
Figura 51 - Página Estatísticas - Gráficos .....	45
Figura 52 - Página Estatísticas - Reservas.....	45
Figura 53 - Página Estatísticas - Reviews.....	46
Figura 54 - Diagrama de implantação.....	49
Figura 55 - Estrutura da API.....	52
Figura 56 - Gestão de many-to-many .....	54
Figura 57 - Incremento de visualizações do livro .....	54
Figura 58 - Endpoints para aprovar e rejeitar reviews.....	55
Figura 59 - Estrutura da app MVC .....	57
Figura 60 - Método Checkout do carrinho de compras (1).....	60
Figura 61 - Método Checkout do carrinho de compras (2).....	61
Figura 62 - Método GetOrCreateAutor.....	62
Figura 63 - Verificar disponibilidade da reserva .....	63
Figura 64 - Controller da página Estatísticas.....	64



# 1 Introdução

O presente projeto, intitulado “Biblioteca Escolar Digital (BookTalk)”, insere-se no âmbito do desenvolvimento de aplicações web para o setor educativo, mais concretamente para bibliotecas escolares dos 2.º e 3.º ciclos do ensino básico. O objetivo do sistema é disponibilizar uma plataforma digital que permita gerir de forma eficiente o catálogo de uma biblioteca escolar, facilitando o processo de pesquisa, reserva e interação dos alunos com os livros disponíveis, bem como apoiar o bibliotecário na gestão do arquivo e na análise da utilização do sistema.

O problema que se pretende resolver está relacionado com a dificuldade que muitas bibliotecas escolares ainda apresentam na organização e disponibilização digital dos seus catálogos, o que limita a consulta remota, a gestão de reservas e a recolha de informação sobre os hábitos de utilização dos alunos. O sistema BookTalk surge como uma solução integrada, permitindo aos leitores pesquisar livros, efetuar reservas e submeter reviews, enquanto o bibliotecário dispõe de funcionalidades para gestão do catálogo, moderação de conteúdo e consulta de estatísticas.

A motivação para a realização deste projeto resulta da necessidade de modernização das ferramentas de apoio ao ensino e da aplicação prática dos conhecimentos adquiridos ao longo da unidade curricular de Desenvolvimento de Aplicações Web. O desenvolvimento do sistema foi realizado de forma incremental, abrangendo as fases de Análise, Desenho e Implementação, refletidas na estrutura do presente relatório. Na fase de implementação, o sistema evoluiu para uma arquitetura cliente-servidor, composta por uma aplicação frontend desenvolvida em ASP.NET Core MVC e uma API REST responsável pela lógica de negócios e pelo acesso à base de dados relacional. Foram ainda implementadas funcionalidades como o carrinho de reservas, a moderação de reviews e um painel de estatísticas com indicadores de utilização do sistema.

Este relatório encontra-se organizado da seguinte forma:

No **Capítulo 2 - Análise do Sistema**, é apresentada a caracterização dos atores e o diagrama de casos de uso que descreve as principais interações com o sistema.

O **Capítulo 3 - Desenho do Sistema** aborda a modelação da base de dados, incluindo os diagramas entidade-relação e o modelo lógico relacional, bem como a modelação das interfaces gráficas através de storyboards e wireframes.

No **Capítulo 4** são descritas as melhorias efetuadas na análise e no desenho do sistema, resultantes do feedback obtido durante a apresentação dessas fases ao docente.

O **Capítulo 5** - Implementação detalha a arquitetura do sistema, as tecnologias utilizadas, o desenvolvimento da API REST e da aplicação frontend MVC, assim como os principais casos relevantes de programação.

Por fim, o **Capítulo 6** - Conclusão e Trabalho Futuro apresenta um balanço do trabalho realizado e identifica possíveis evoluções e melhorias a implementar no sistema.

## 2 Análise do Sistema

A fase de análise tem como objetivo identificar, descrever e compreender o funcionamento do sistema BookTalk antes do início da sua implementação. Nesta etapa são definidos os atores envolvidos, os casos de uso que representam as principais interações com o sistema, bem como os requisitos funcionais e não funcionais que orientam o desenvolvimento.

Esta análise permite estabelecer uma visão clara sobre o que o sistema deve fazer, quem o utiliza e como essas interações se processam, servindo de base para o desenho posterior das interfaces e da estrutura de dados.

O projeto parte da evolução do trabalho desenvolvido na unidade curricular anterior (Tecnologias para a Web e Ambientes Móveis), mantendo os quatro casos de uso originais e acrescentando um quinto caso de uso dedicado ao módulo de Data Analytics, em conformidade com os requisitos da unidade curricular de Desenvolvimento de Aplicações Web.

O resultado desta fase é um conjunto estruturado de modelos e especificações que descrevem de forma detalhada o comportamento esperado do sistema, garantindo coerência e fundamentação técnica para a fase seguinte - o Desenho.

### 2.1 Caracterização dos atores

O sistema BookTalk envolve dois tipos principais de utilizadores, cada um com responsabilidades e permissões distintas. Durante a análise foram definidos os seguintes atores:

**U1 – Bibliotecário:** Responsável pela gestão do catálogo de livros, moderação de conteúdo e análise de dados estatísticos. É o utilizador com permissões de administração.

Permissões: Inserir, editar e remover livros; moderar reviews; consultar estatísticas (módulo Analytics).

**U2 – Leitor (aluno):** Utilizador final da aplicação, com acesso ao catálogo digital. Pode pesquisar livros, efetuar reservas, classificar e escrever reviews.

Permissões: Pesquisar livros, efetuar reservas, escrever e editar as suas reviews, consultar reviews aprovadas.

O sistema foi concebido para que ambos os perfis interajam num ambiente seguro, com interfaces distintas mas complementares: o bibliotecário atua no back-office administrativo e o leitor no front-office público. Para compreender melhor os perfis e necessidades dos utilizadores, foram criadas duas *personas* representativas — uma para cada ator identificado. Estas personas refletem motivações, objetivos e desafios típicos no contexto de utilização da aplicação.

## **Persona 1 (Bibliotecária)**

**Nome:** Joana Mendes

**Idade:** 43 anos

**Função:** Bibliotecária na Escola Básica 2.º e 3.º Ciclo Santa Clara

**Contexto:** Gere manualmente a coleção de livros da escola há vários anos. Os registos são mantidos em folhas de cálculo e cadernos físicos.

### **Objetivos:**

- Digitalizar o catálogo da biblioteca;
- Facilitar a reserva de livros;
- Modificar e validar as reviews escritas pelos alunos;
- Obter estatísticas sobre utilização da biblioteca (livros mais populares, épocas de maior procura).

### **Desafios:**

- Perda de tempo com processos manuais;
- Dificuldade em acompanhar a procura dos livros;
- Falta de ferramentas de controlo e de métricas de utilização.

**Competências tecnológicas:** Utilizadora intermédia; confortável com sistemas web simples e interfaces gráficas intuitivas.

## **Persona 2 (Aluna/Leitora)**

**Nome:** Maria Monteiro

**Idade:** 11 anos

**Ano de Escolaridade:** 5º ano (2º Ciclo)

**Contexto:** Utilizadora frequente da biblioteca escolar; gosta de ler e partilhar opiniões sobre livros com colegas.

### **Objetivos:**

- Procurar livros de forma rápida e intuitiva;
- Reservar um livro;
- Escrever e publicar reviews sobre as leituras;
- Ver opiniões de outros alunos.

### **Desafios:**

- Dificuldade em saber se um livro está disponível;
- Ter de ir fisicamente à biblioteca para reservar;
- Não conseguir partilhar a sua opinião facilmente.

**Competências tecnológicas:** Utilizadora jovem e habituada a aplicações simples.

## 2.2 Diagrama de Casos de Uso

O diagrama de casos de uso da aplicação BookTalk apresenta os principais atores e as funcionalidades que cada um pode executar.

O sistema conta com dois atores:

- Bibliotecário, responsável pela gestão do catálogo, moderação de reviews e consulta do painel de estatísticas.
- Leitor (Aluno), que pesquisa livros, efetua reservas e submete reviews.

O diagrama inclui cinco casos de uso principais:

1. Adicionar livro ao catálogo
2. Reservar um livro
3. Escrever e submeter review
4. Moderar e validar review
5. Consultar painel de estatísticas (Data Analytics)

A Figura 1 ilustra as interações entre estes atores e o sistema.

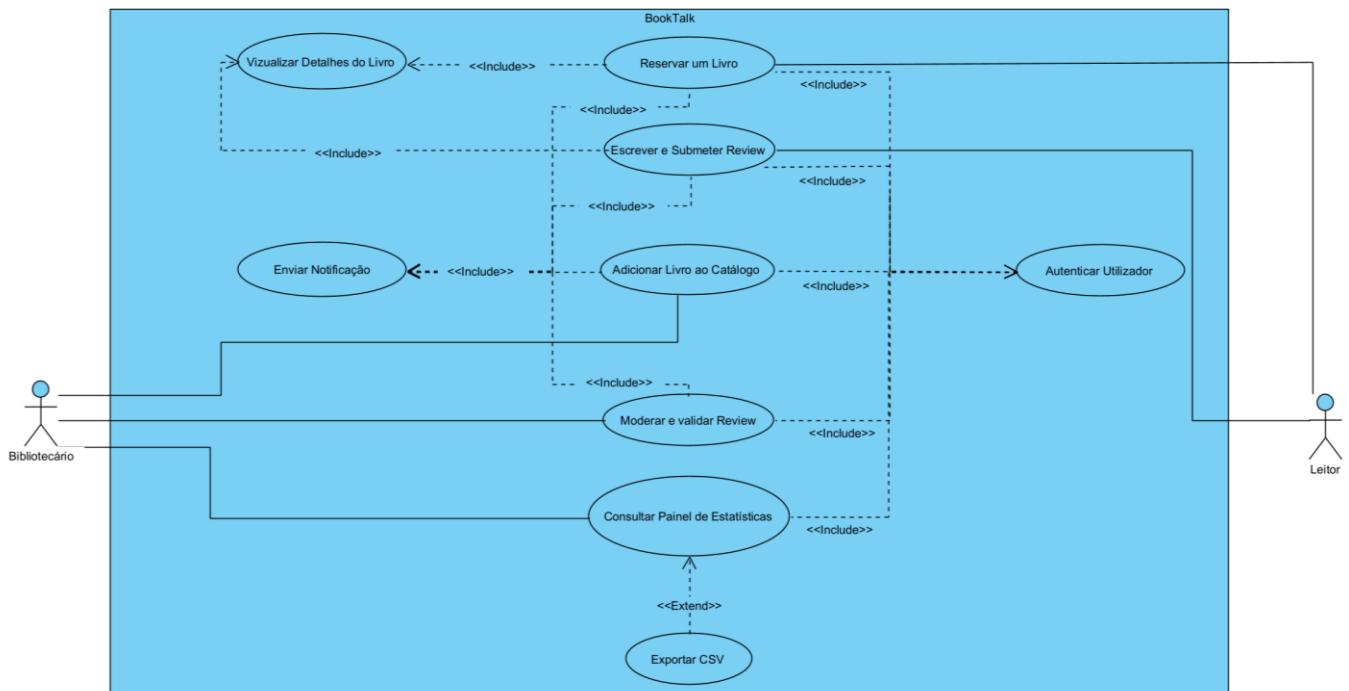


Figura 1 - Diagrama de Casos de Uso do Sistema BookTalk

Após a identificação dos casos de uso no diagrama anterior, apresentam-se de seguida as respetivas tabelas de especificação de requisitos, elaboradas segundo a notação UML.

Estas tabelas descrevem o comportamento esperado de cada caso de uso, as interações com os atores e as condições associadas à execução de cada funcionalidade do sistema.

<b>UC1 – Adicionar livros ao catálogo</b>	
<b>Atores</b>	Bibliotecário
<b>Descrição</b>	O bibliotecário introduz manualmente os dados de um novo livro (título, autores, ISBN, editora, categoria, data de publicação, língua, sinopse, capa). O sistema valida e grava o registo.
<b>Pré-condições</b>	Bibliotecário autenticado.
<b>Pós-condições</b>	Livro criado com estado <b>ativo</b> e disponível para pesquisa e reserva.
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1) Aceder “Gestão de Catálogo” → “Adicionar livro”.</li> <li>2) Preencher formulário.</li> <li>3) Submeter.</li> <li>4) Sistema valida (campos obrigatórios, ISBN único).</li> <li>5) Sistema grava e confirma.</li> </ol>
<b>Cenários Alternativos</b>	<ul style="list-style-type: none"> <li>A1) Cancelar antes de gravar.</li> <li>A2) Adiar upload de capa (pode ser adicionado depois).</li> <li>A3) Deteção de duplicado por título + autor → aviso e opção “Criar mesmo assim”.</li> </ul>
<b>Erros/Exceções</b>	<ul style="list-style-type: none"> <li>E1) ISBN duplicado.</li> <li>E2) Campos obrigatórios em falta.</li> <li>E3) Erro de persistência.</li> </ul>

## UC2 – Reservar um livro

<b>Atores</b>	Leitor
<b>Descrição</b>	O leitor consulta a ficha do livro, verifica disponibilidade e efetua uma reserva semanal.
<b>Pré-condições</b>	Leitor autenticado; livro disponível no intervalo selecionado.
<b>Pós-condições</b>	Reserva criada (estado ativa), associada ao leitor e ao livro; registo com semana_início/semana_fim.
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1) Pesquisar e abrir o livro.</li> <li>2) “Agendar reserva”.</li> <li>3) Selecionar semana válida.</li> <li>4) Confirmar.</li> <li>5) Sistema cria reserva e apresenta resumo.</li> </ol>
<b>Cenários Alternativos</b>	<p>A1) Alterar ou cancelar antes de confirmar.</p> <p>A2) Se o leitor tentar reservar mais de um livro ao mesmo tempo, o sistema mostra um aviso de limite máximo e permite priorizar qual reserva manter.</p>
<b>Erros/Exceções</b>	<p>E1) Conflito de disponibilidade.</p> <p>E2) Regra: Máxima 1 reserva ativa por leitor/livro/semana.</p>

## UC3 – Escrever e submeter uma review

<b>Atores</b>	Leitor
<b>Descrição</b>	O leitor escreve uma review e atribui rating 1–10 a um livro. A review entra em pendente até moderação.
<b>Pré-condições</b>	Leitor autenticado.
<b>Pós-condições</b>	Review criada com estado pendente (não visível publicamente).
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1) Abrir detalhes do livro.</li> <li>2) “Escrever review”.</li> <li>3) Introduzir texto e rating.</li> <li>4) Submeter.</li> <li>5) Sistema regista e mostra aviso “Enviado para moderação”.</li> </ol>
<b>Cenários Alternativos</b>	A1) Editar ou remover a própria review enquanto pendente.
<b>Erros/Exceções</b>	E1) Texto vazio ou demasiado curto.

## UC4 – Validar ou rejeitar reviews

<b>Atores</b>	Bibliotecário
<b>Descrição</b>	O bibliotecário consulta a lista de reviews pendentes, analisa conteúdo e decide aprovar (torna visível) ou rejeitar.
<b>Pré-condições</b>	Bibliotecário autenticado; existência de reviews pendentes.
<b>Pós-condições</b>	Review passa a aprovada (visível) ou rejeitada (oculta). Guarda-se data de validação e moderador.
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1) Abrir “Reviews por aprovar”.</li> <li>2) Selecionar review.</li> <li>3) Rever conteúdo.</li> <li>4) Aprovar ou Rejeitar.</li> <li>5) Sistema atualiza estado e confirma.</li> </ol>
<b>Cenários Alternativos</b>	A1) Filtrar por livro/utilizador/periódico.
<b>Erros/Exceções</b>	E1) Review inexistente ou já removida pelo autor antes da moderação.

UC5 – Consultar painel de estatísticas (Data Analytics)	
<b>Atores</b>	Bibliotecário
<b>Descrição</b>	O bibliotecário acede a um dashboard com indicadores e gráficos sobre utilização do sistema (reservas, reviews, tendências).
<b>Pré-condições</b>	Bibliotecário autenticado; existência de dados/ eventos registados.
<b>Pós-condições</b>	Métricas e gráficos apresentados segundo filtros aplicados (não altera estado do sistema).
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1) Aceder a “Analytics”.</li> <li>2) Definir período/filtros (mês/trimestre/ano, categoria).</li> <li>3) Visualizar: Top 5 livros reservados, Reservas por mês, Tempo médio de moderação, Total de reviews aprovadas/rejeitadas.</li> <li>4) (Opcional) Exportar CSV/PDF.</li> </ol>
<b>Cenários Alternativos</b>	A1) Falta de dados no período → mensagem “Sem dados para o filtro selecionado”.
<b>Erros/Exceções</b>	E1) Consultas demoradas (aplicar paginação – exemplo: mostrar 10 registos de cada vez/filtros obrigatórios).

## 3 Desenho do Sistema

Esta secção descreve a estrutura conceptual e visual do sistema *BookTalk*, apresentando os modelos que suportam o seu funcionamento. O objetivo é definir a organização dos dados e as interfaces de interação entre o utilizador e a aplicação, garantindo coerência entre a componente lógica e a experiência de utilização.

### 3.1 Modelação da Base Dados

A modelação da base de dados define a estrutura de armazenamento da informação no sistema *BookTalk*. Nesta fase são apresentadas as entidades principais, as relações entre elas e o modelo relacional correspondente, assegurando a integridade e consistência dos dados ao longo de todas as operações do sistema.

### 3.1.1 Diagrama E/R

O diagrama Entidade/Relacionamento representa a estrutura conceptual da base de dados do sistema *BookTalk*. Nele são identificadas as principais entidades (*Livro*, *Review*, *Reserva* e *Utilizador*) e as relações entre elas. Este modelo permite compreender como os diferentes elementos do sistema se interligam, garantindo coerência na organização dos dados.

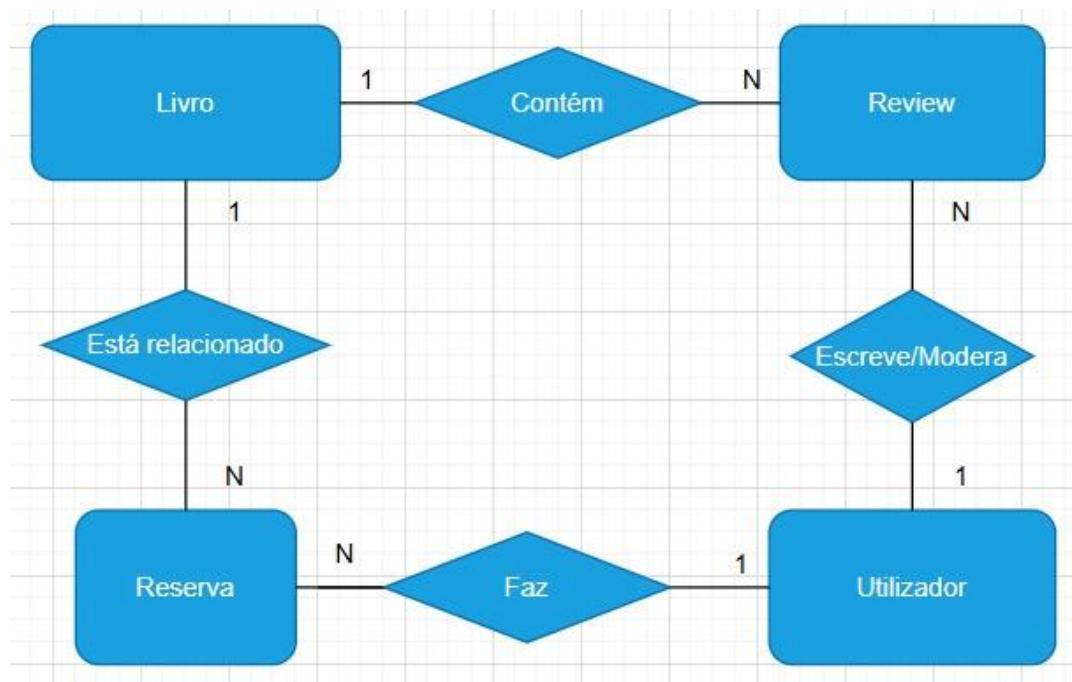


Figura 2 - Modelo Entidade Relação do Sistema BookTalk

### 3.1.2 Modelo Lógico Relacional

O modelo lógico relacional traduz o diagrama E/R para um conjunto de tabelas interligadas, definindo chaves primárias e estrangeiras que asseguram a integridade referencial da base de dados. (Conceptual, Logical and Physical Data Model, s.d.) O modelo inclui as tabelas *Livro*, *Review*, *Reserva* e *Utilizador*, permitindo o armazenamento estruturado das informações associadas à gestão da biblioteca digital *BookTalk*.

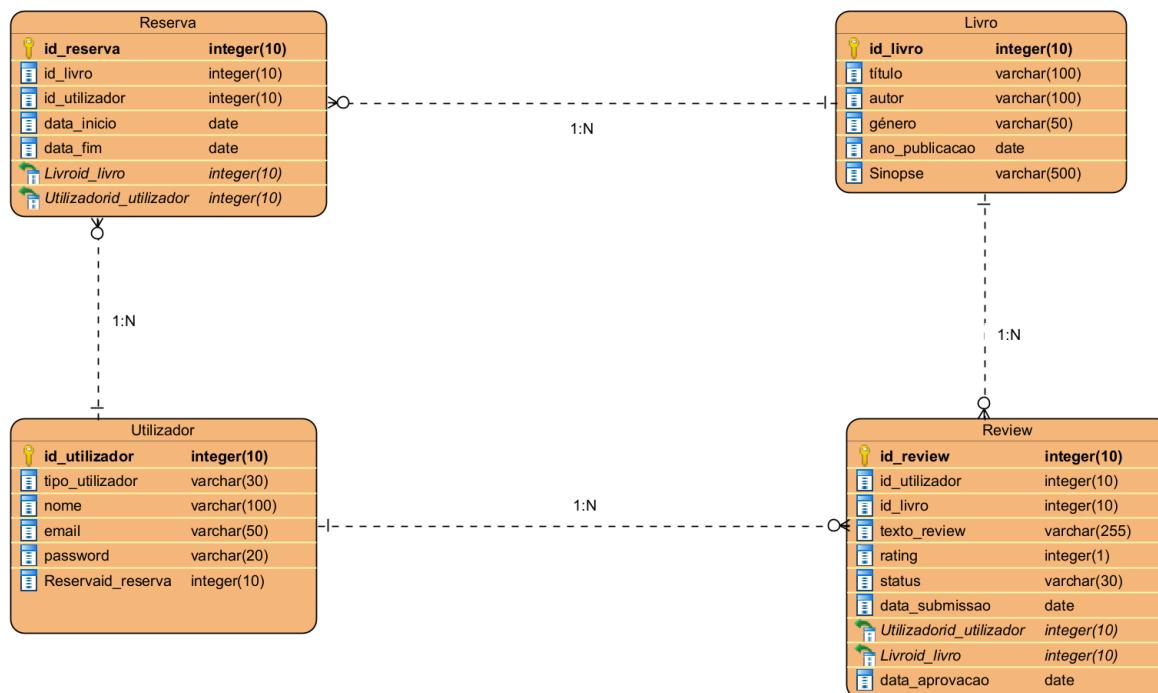


Figura 3 - Modelo Lógico Relacional do Sistema BookTalk

## 3.2 Modelação de Interfaces Gráficas com o Utilizador

Nesta secção apresenta-se uma breve descrição de cada caso de uso, seguido dos respetivos storyboards de baixa fidelidade (desenhados em papel) e dos wireframes de média fidelidade (desenvolvidos na ferramenta Balsamiq).

### 1º Caso de Uso: Adicionar livros ao catálogo

**Objetivo:** permitir ao bibliotecário criar novos registo no catálogo.

**Pré-condições:** bibliotecário autenticado.

**Pós-condições:** livro criado (estado *ativo*).

**Cenário:**

1. A Joana (bibliotecária) autentica-se e acede a Gestão do Catálogo → Adicionar livro.
2. Preenche os campos: Título, Autores, ISBN, Editora, Categoria, Data de publicação, Língua, Sinopse e upload de capa.
3. Carrega em Guardar.
4. O sistema valida campos obrigatórios e unicidade do ISBN.
5. Em caso de sucesso, grava o registo e mostra confirmação; o livro fica disponível para ser pesquisado e reservado.

### 2º Caso de Uso: Reservar um livro

**Objetivo:** permitir ao leitor reservar um livro.

**Pré-condições:** leitor autenticado; livro disponível no período.

**Pós-condições:** reserva criada (estado *ativa*).

**Cenário:**

1. A Maria (aluna) pesquisa um título e abre a página de detalhes do livro.
2. Selecciona Agendar reserva e surge um calendário semanal.
3. Escolhe uma semana válida e confirma.
4. O sistema verifica conflitos e cria a reserva; apresenta resumo (datas, título) e opção para ver reservas.

### **3º Caso de Uso: Classificar e escrever uma review sobre um livro**

**Objetivo:** permitir ao leitor avaliar e comentar livros.

**Pré-condições:** leitor autenticado.

**Pós-condições:** review criada (estado *pendente*).

**Cenário:**

1. A Maria (aluna) abre um livro e clica em Escrever Review.
2. Introduz texto e rating (1-10); submete.
3. O sistema grava a review em pendente e mostra aviso “Aguarda moderação”.

### **4º Caso de Uso: Moderar e validar reviews**

**Objetivo:** aprovar/rejeitar reviews pendentes.

**Pré-condições:** bibliotecário autenticado; existem reviews pendentes.

**Pós-condições:** review muda para aprovada (visível) ou rejeitada.

**Cenário:**

1. A Joana autentica-se e acede a Reviews por aprovar.
2. Escolhe uma review e abre-a.
3. Lê o conteúdo e decide Aprovar ou Rejeitar.
4. O sistema valida campos obrigatórios e unicidade do ISBN.
5. O sistema atualiza o estado e regista data e moderador.

### **5ª Caso de Uso: Consultar painel de estatísticas (Data Analytics)**

**Objetivo:** fornecer indicadores para decisão.

**Pré-condições:** bibliotecário autenticado; existência de dados.

**Pós-condições:** apresentação de dados conforme filtros.

**Cenário:**

1. A Joana autentica-se e abre o painel analytics.
2. Escolhe o separador que tem as tabelas que quer analisar
3. Analisa a tabela diretamente ou escolhe exportar em csv.

### 3.2.1 Storyboards

#### 3.2.1.1 Tarefa 1 – Adicionar livro ao catálogo

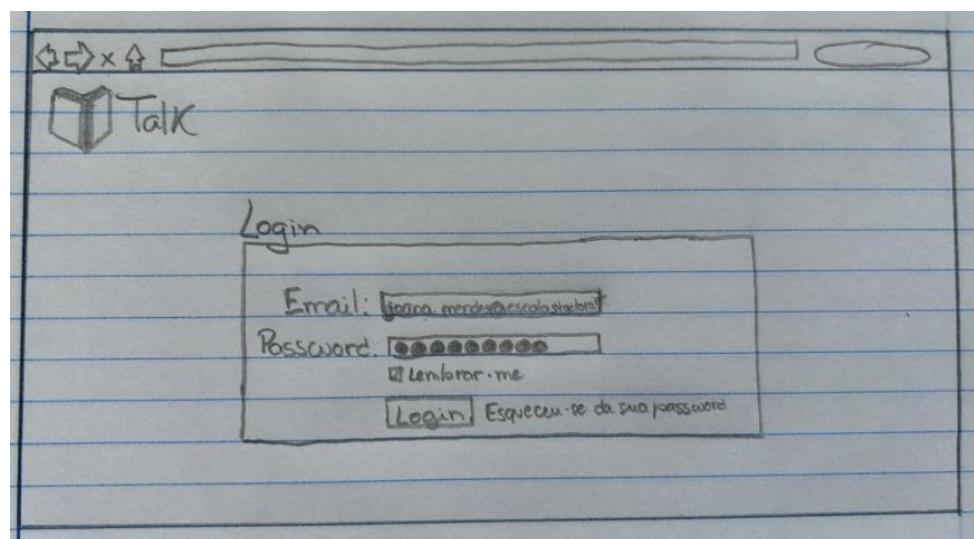


Figura 4 - Login com conta de bibliotecário

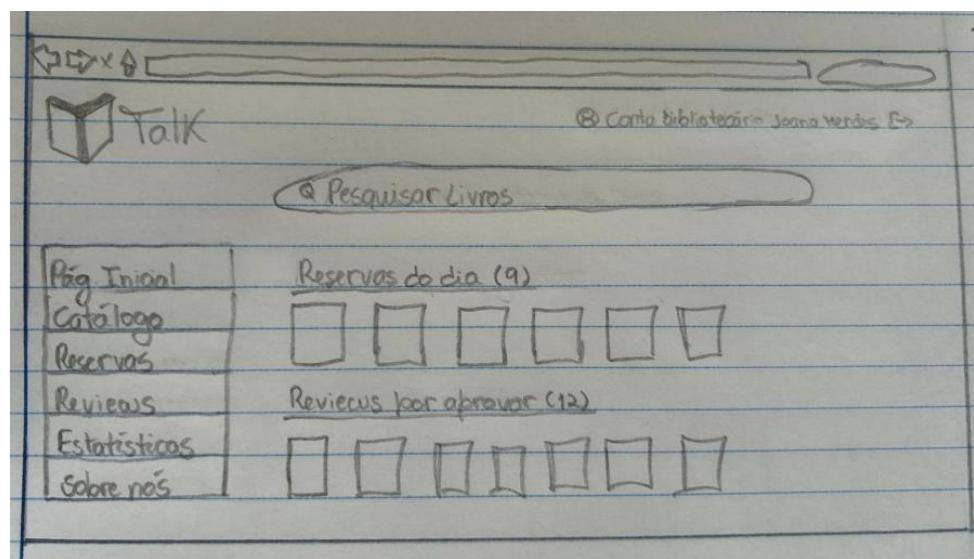


Figura 5 - Página inicial do bibliotecário

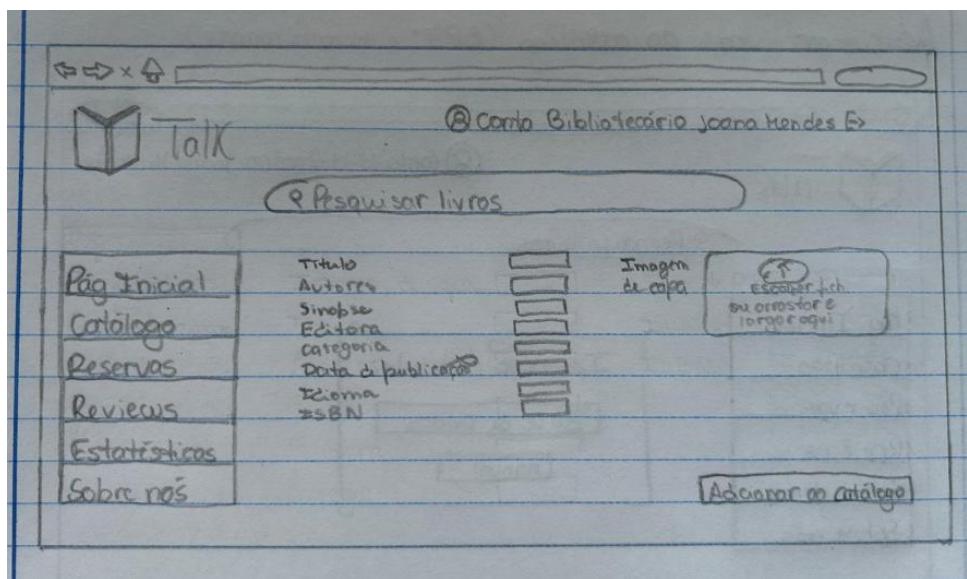


Figura 6 - Adicionar livros manualmente

### 3.2.1.2 Tarefa 2 – Reservar um livro

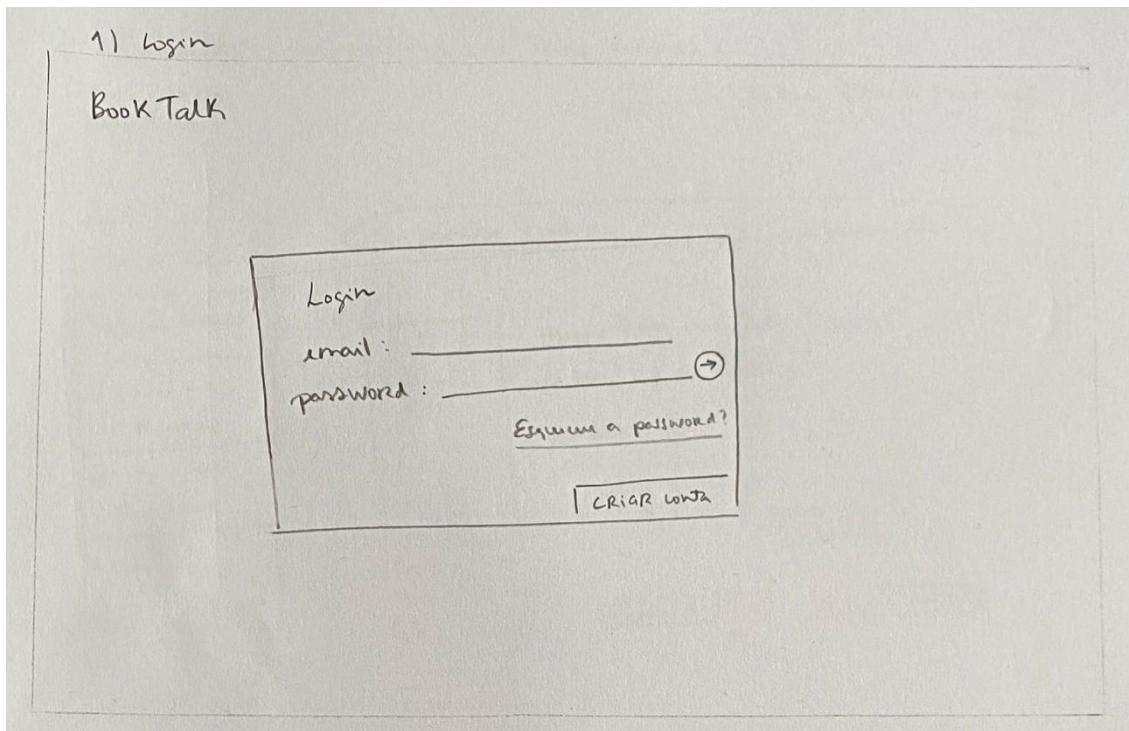


Figura 7 - Página login

2) Página inicial : conta leitor

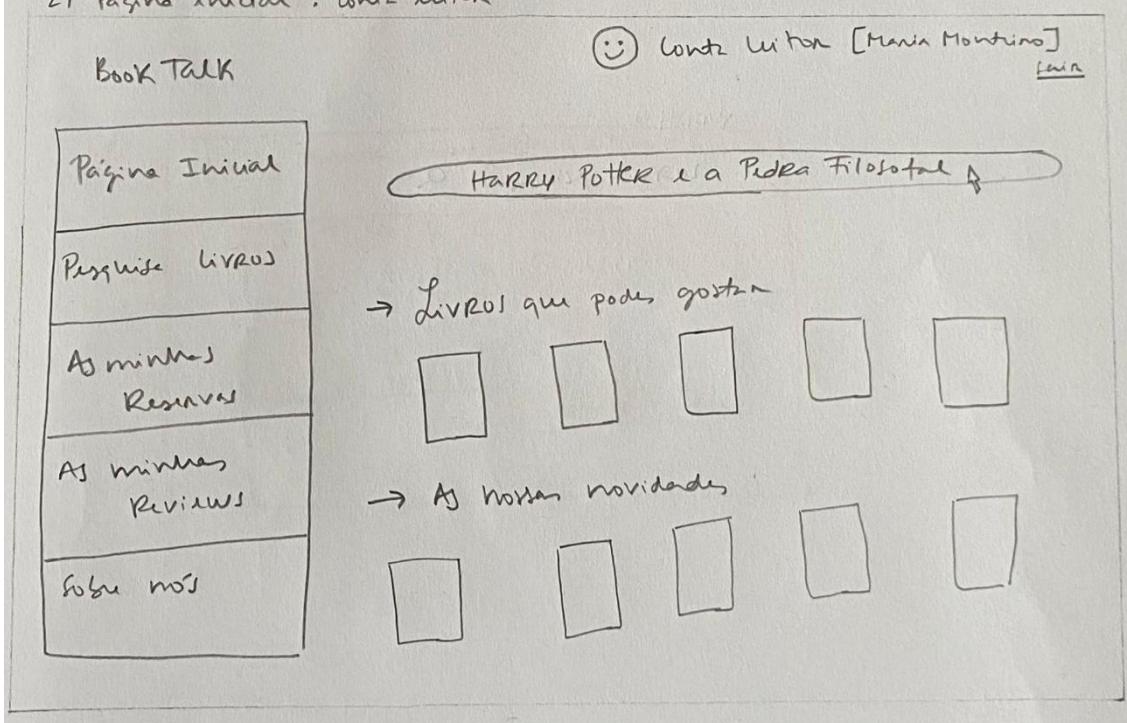


Figura 8 - Página inicial do leitor

3) Página livro : Harry Potter e a Pedra Filosofal

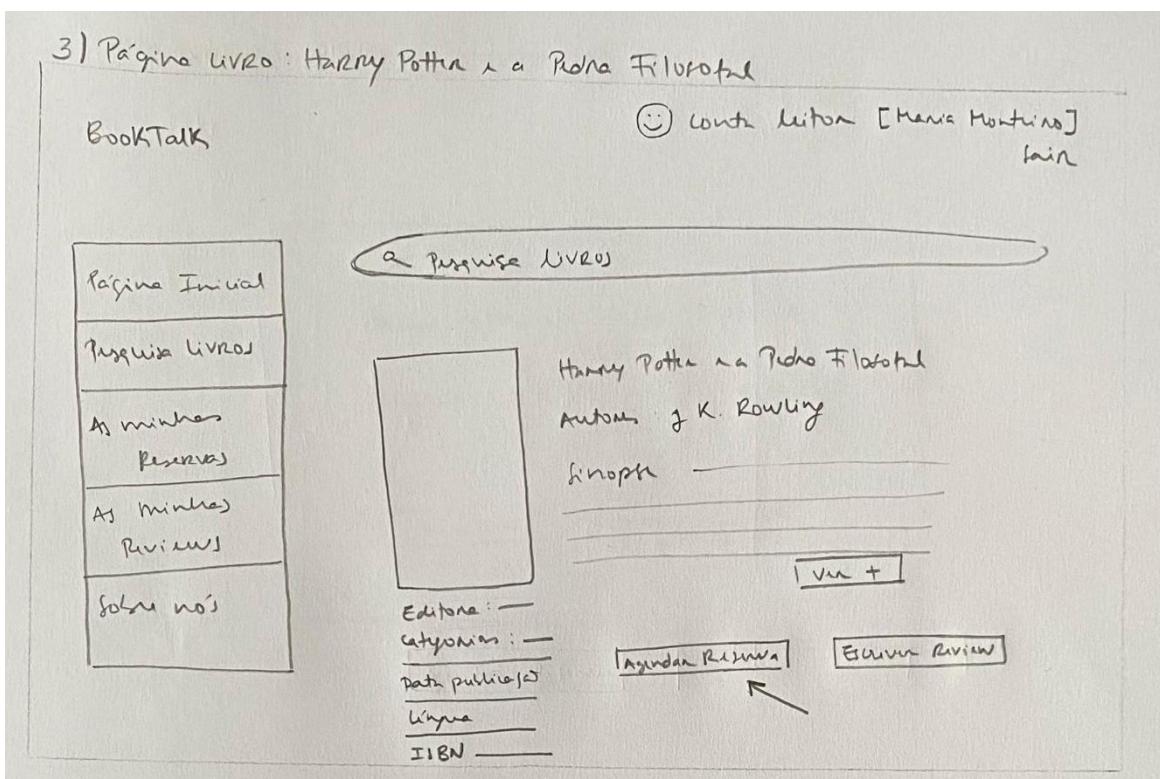


Figura 9 - Página de detalhes de livro

4) Agendar Reserva - pop up

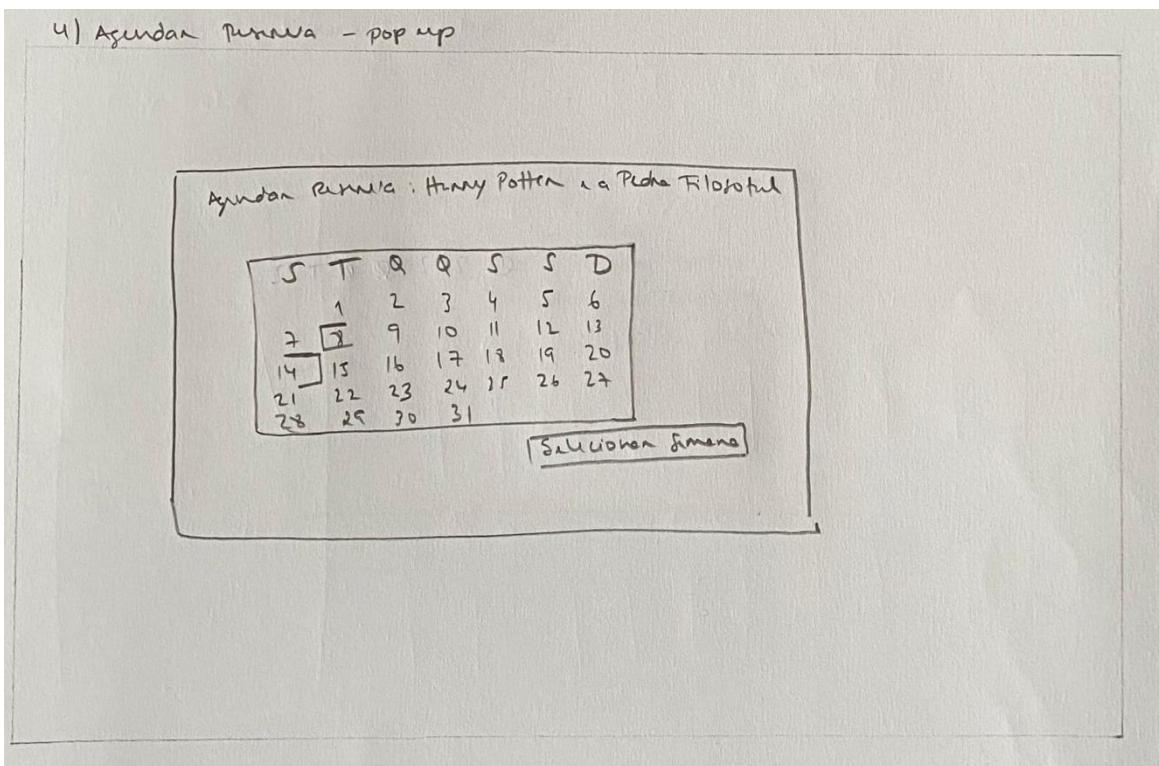


Figura 10 - Página para agendar reserva

5) Confirmar Reserva - pop up

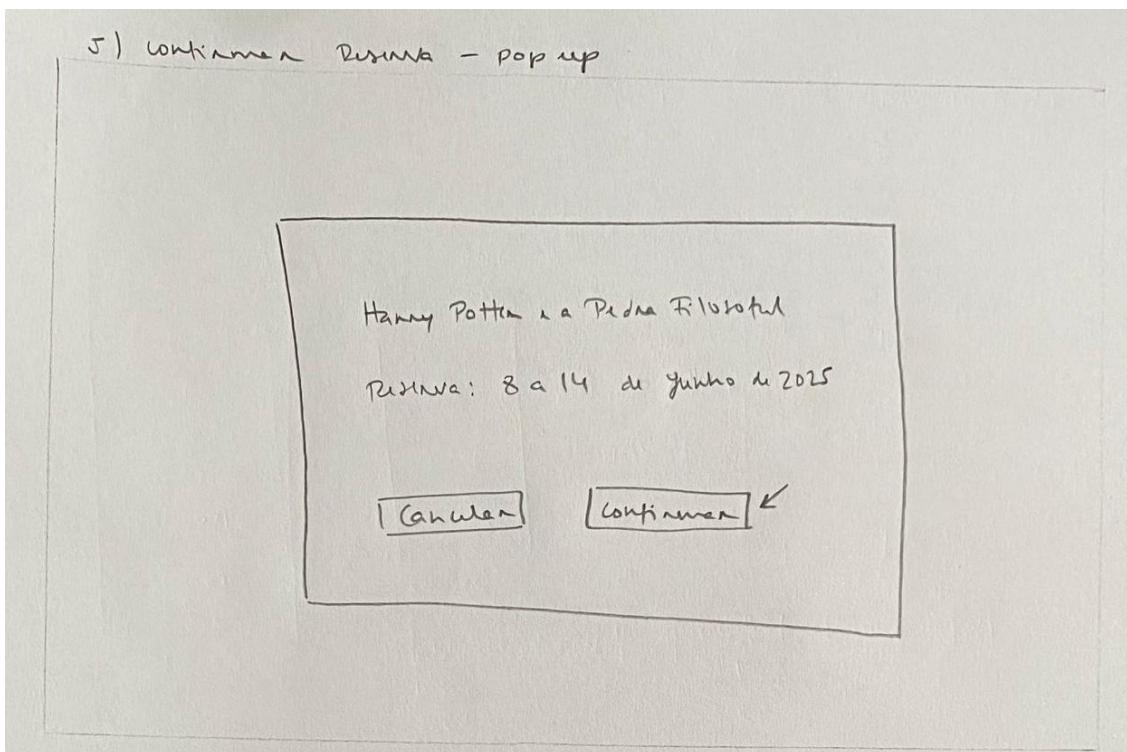


Figura 11 - Página para confirmar reserva

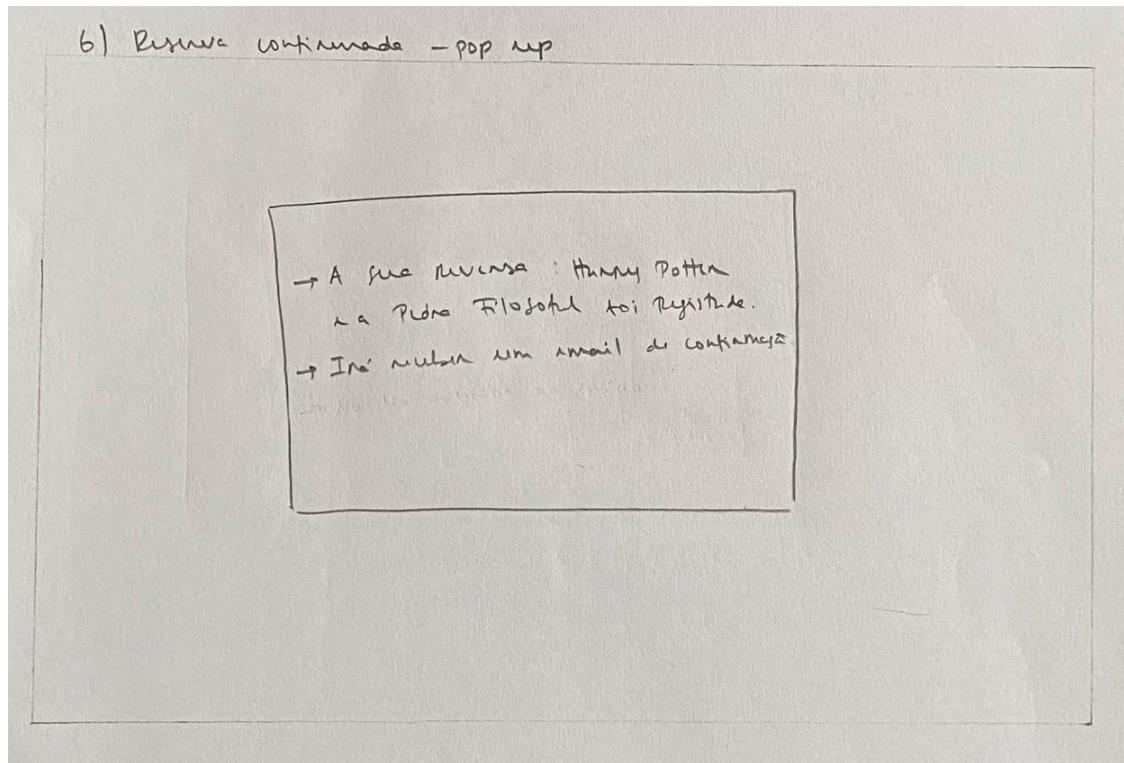


Figura 12 - Página de resumo de reserva

### 3.2.1.3 Tarefa 3 - Avaliar e escrever uma review sobre um livro

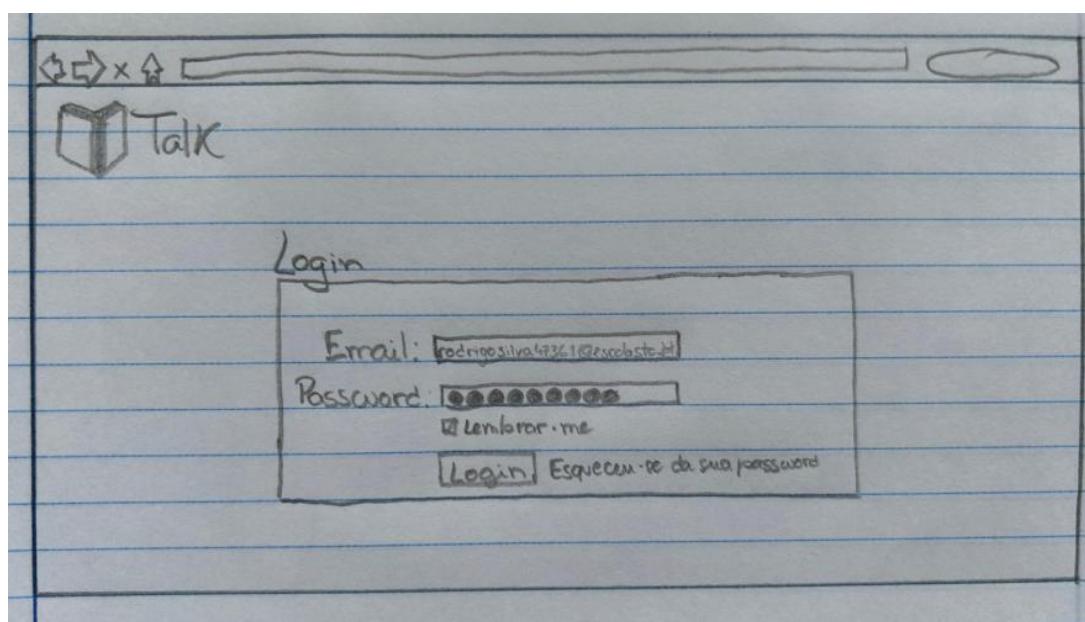


Figura 13 - Login como leitor

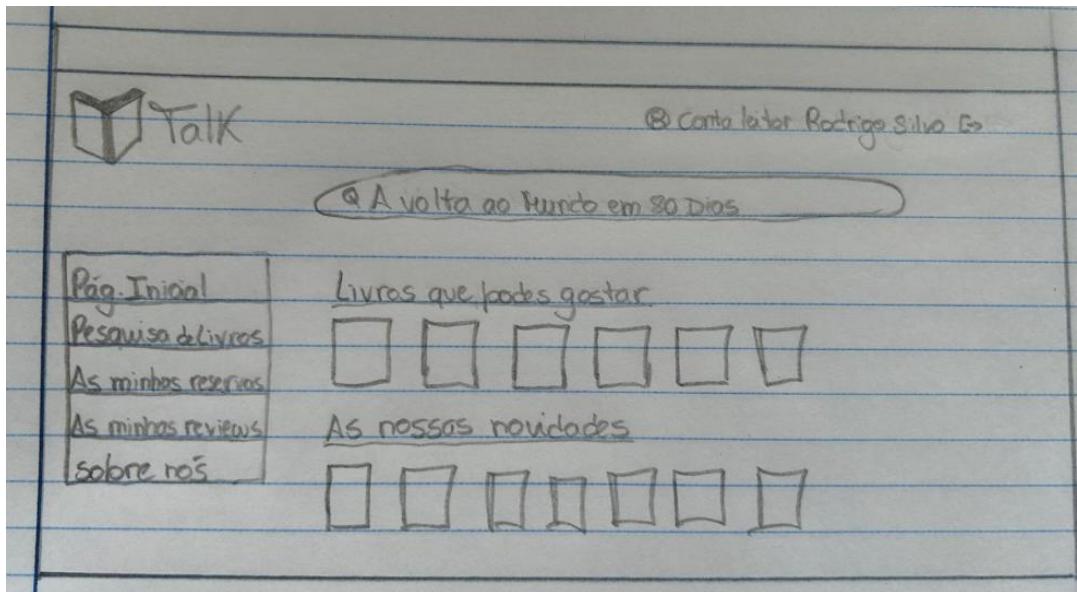


Figura 14 - Página inicial do leitor

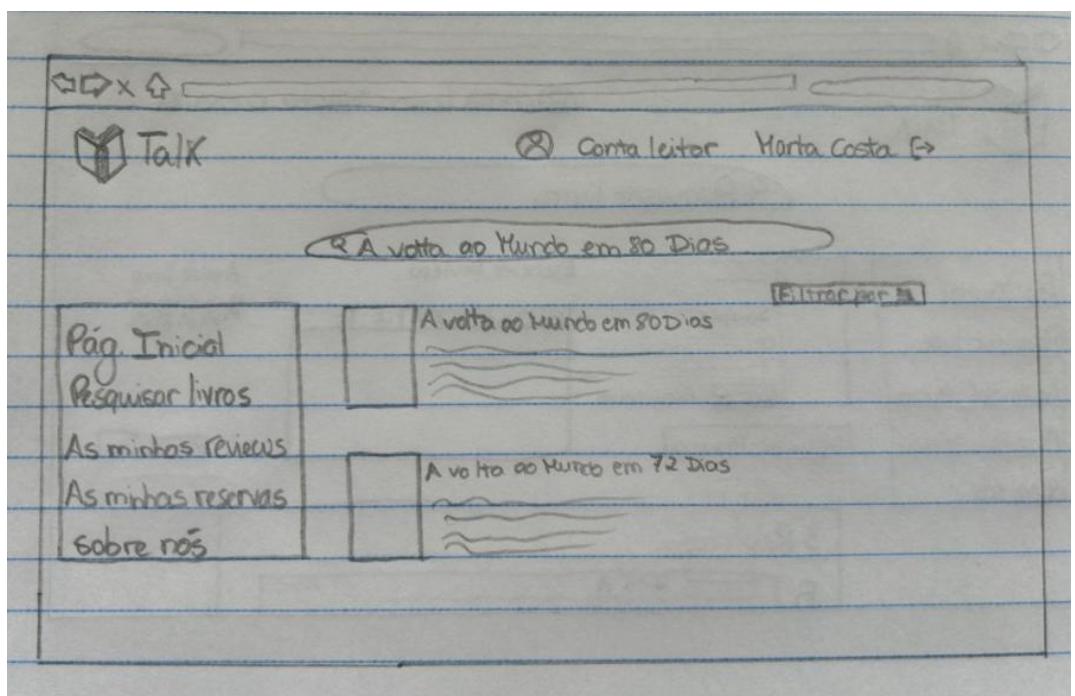


Figura 15 - Pesquisar um livro

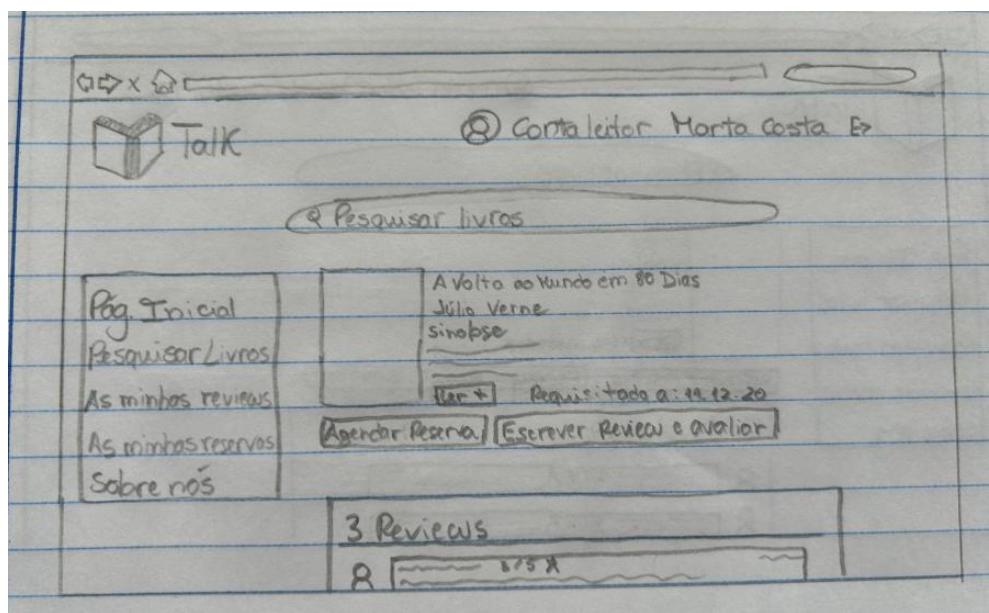


Figura 16 - Detalhes do livro

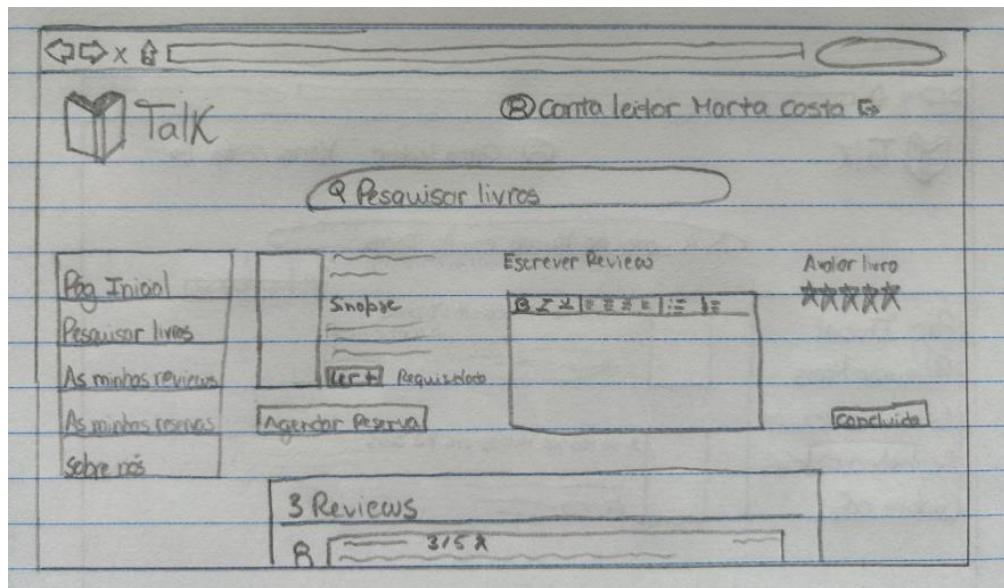


Figura 17 - Escrever review e avaliar o livro

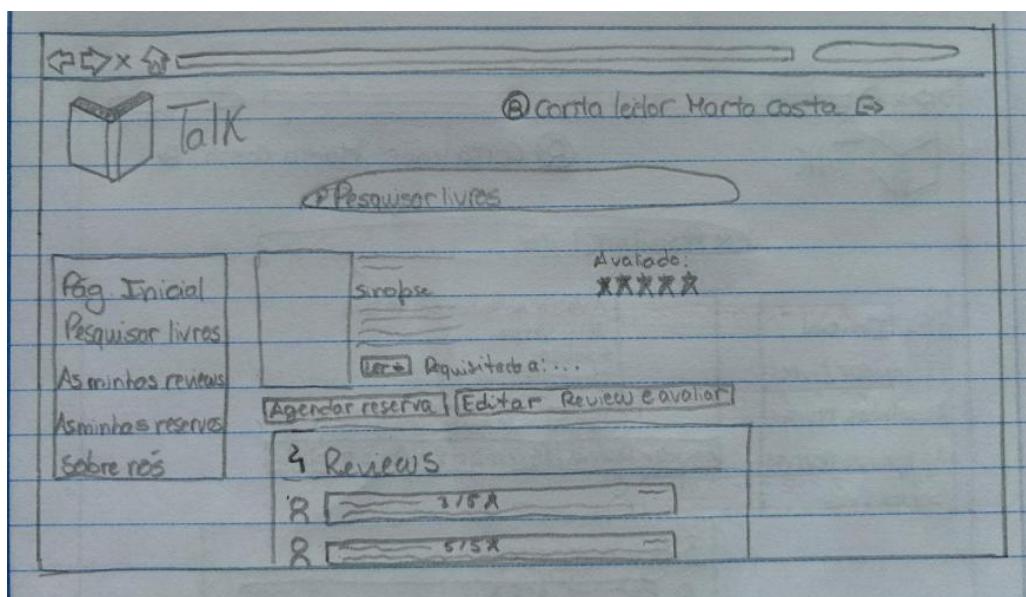


Figura 18 - Review submetida

### 3.2.1.4 Tarefa 4 – Moderar e validar review

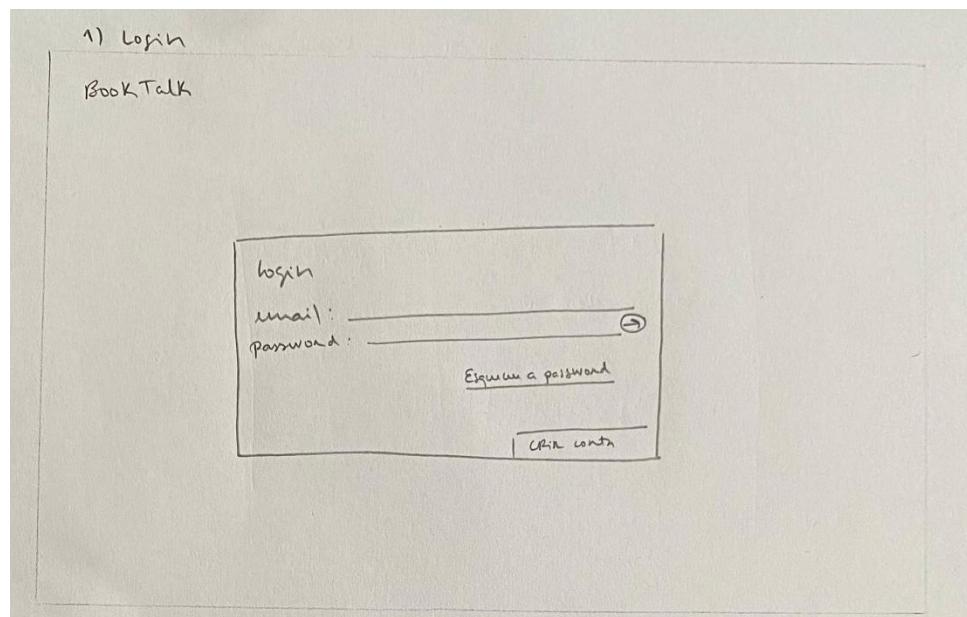


Figura 19 - Página login

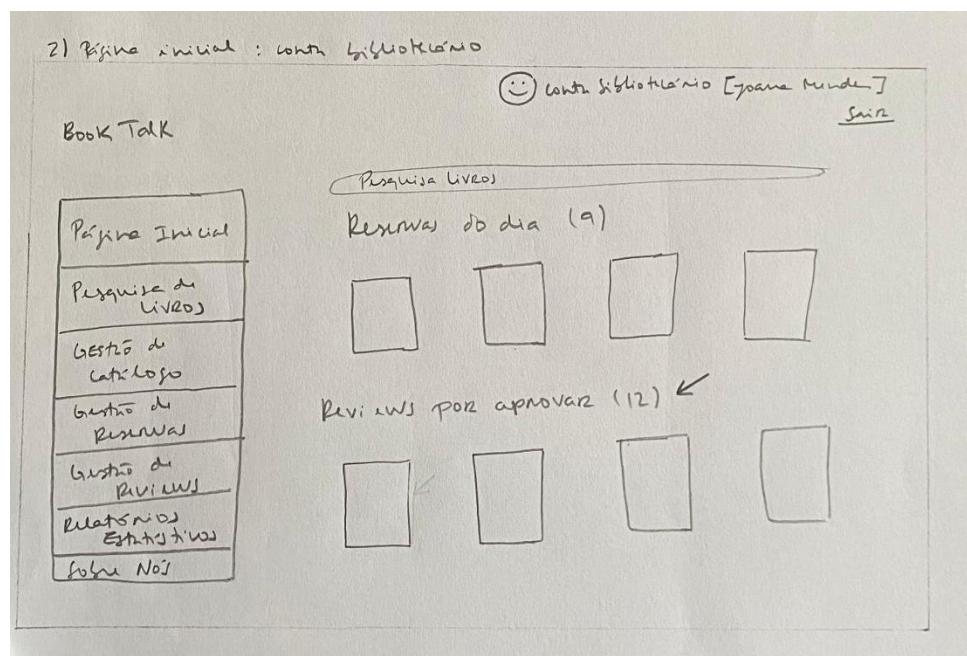


Figura 20 - Página inicial bibliotecário

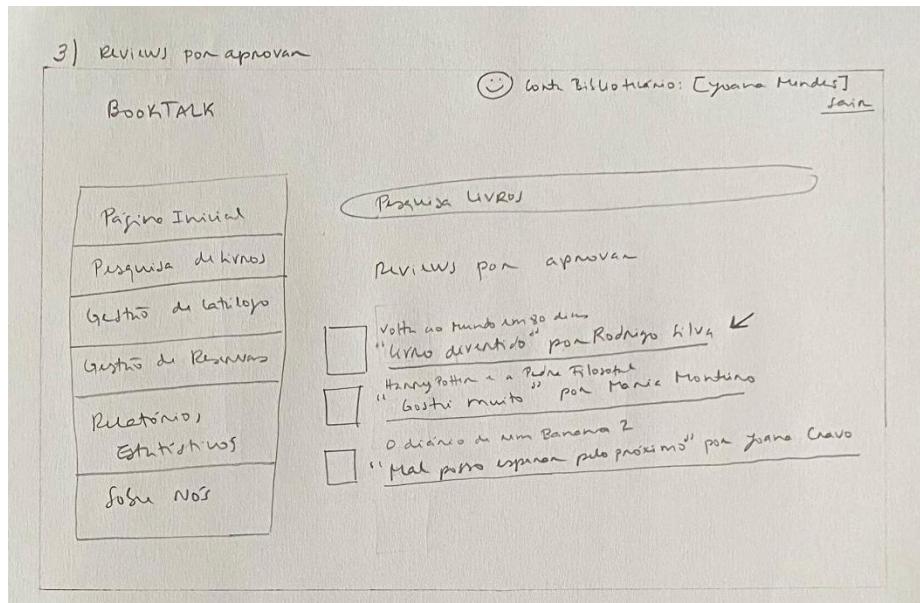


Figura 21 - Página reviews por aprovar

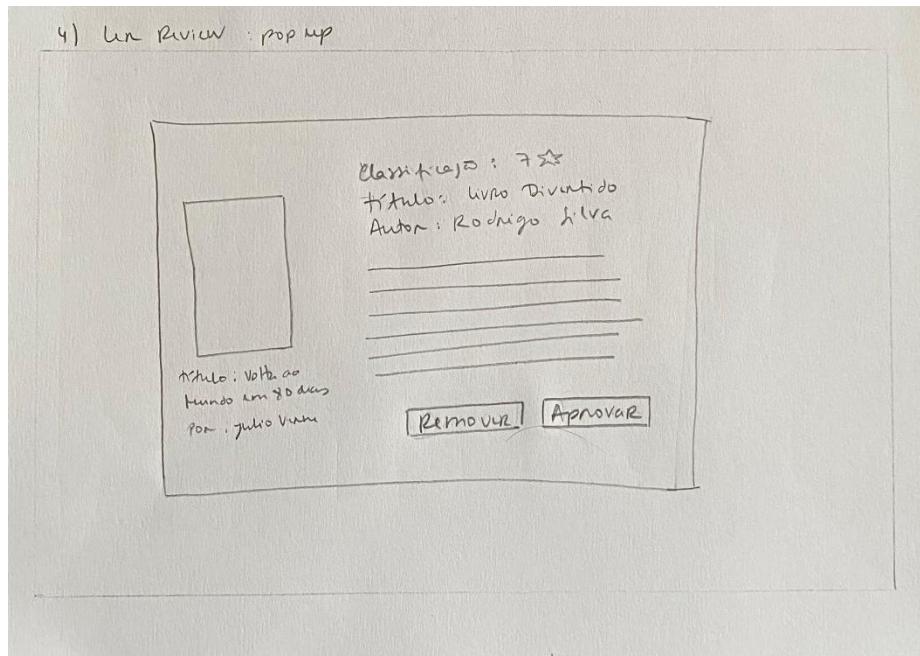


Figura 22 - Página review

### 3.2.1.5 Tarefa 5 – Consultar painel de estatísticas (Data Analytics)

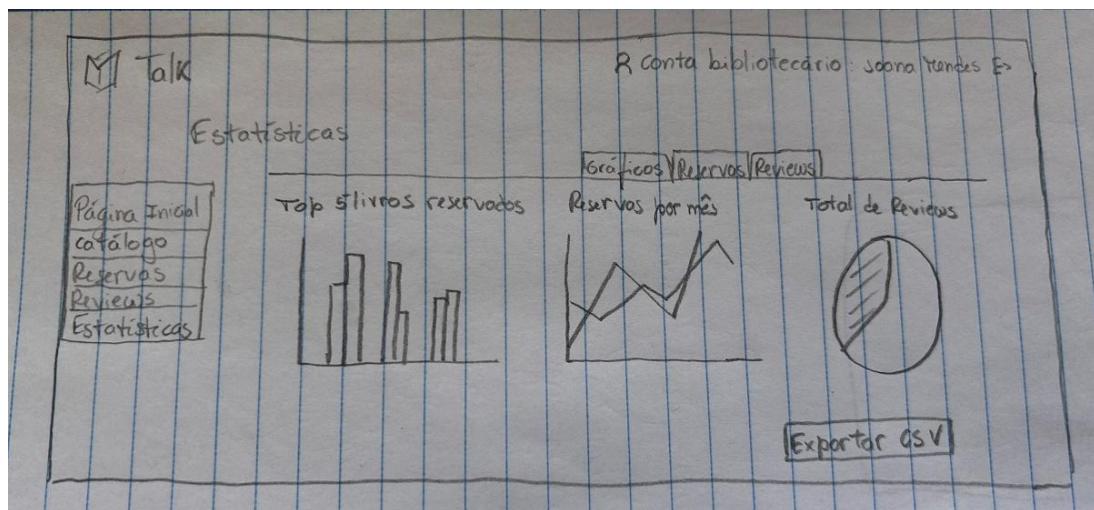


Figura 23 - Página Estatísticas - Gráficos

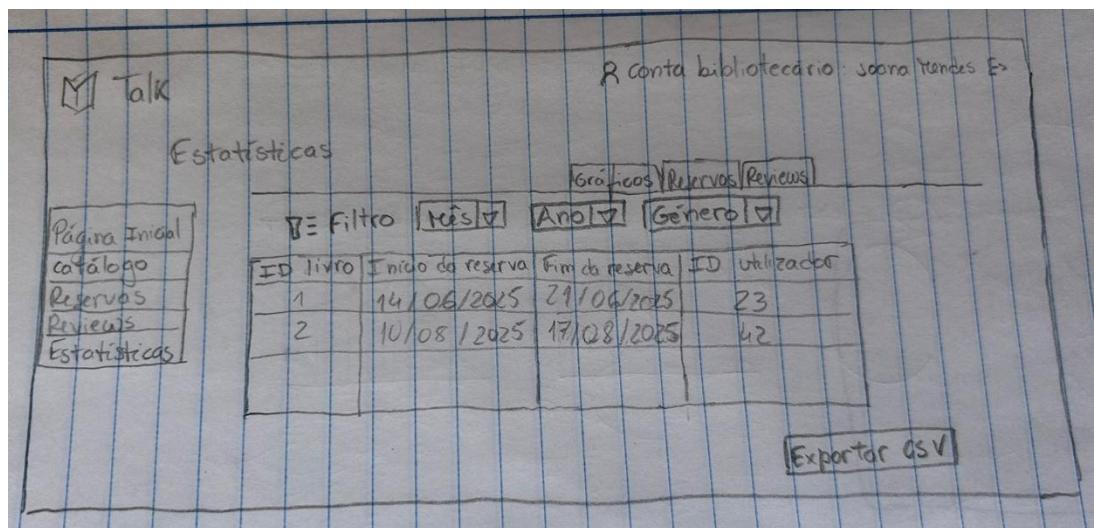


Figura 24 - Página Estatísticas - Reservas

The screenshot shows the 'Talk' application interface. At the top right, it says 'R conta bibliotecário: Joana Mendes >'. On the left, there's a sidebar with links: 'Página Inicial', 'Catálogo', 'Reservas', 'Reviews', and 'Estatísticas'. The main area is titled 'Estatísticas' and has a 'Filtro' section with dropdown menus for 'Mês' (set to 'Ano') and 'Gênero' (set to 'Todos'). Below this is a table with two rows of data:

ID	Livro	Data da review	Classificação	ID utilizador
4		13/05/2025	4	16
7		12/07/2025	5	22

At the bottom right of the table area is a button labeled 'Exportar CSV'.

Figura 25 - Página Estatísticas - Reviews

### 3.2.2 Wireframes

### 3.2.3 Wireframes

#### 3.2.3.1 Tarefa 1 – Adicionar livro ao catálogo

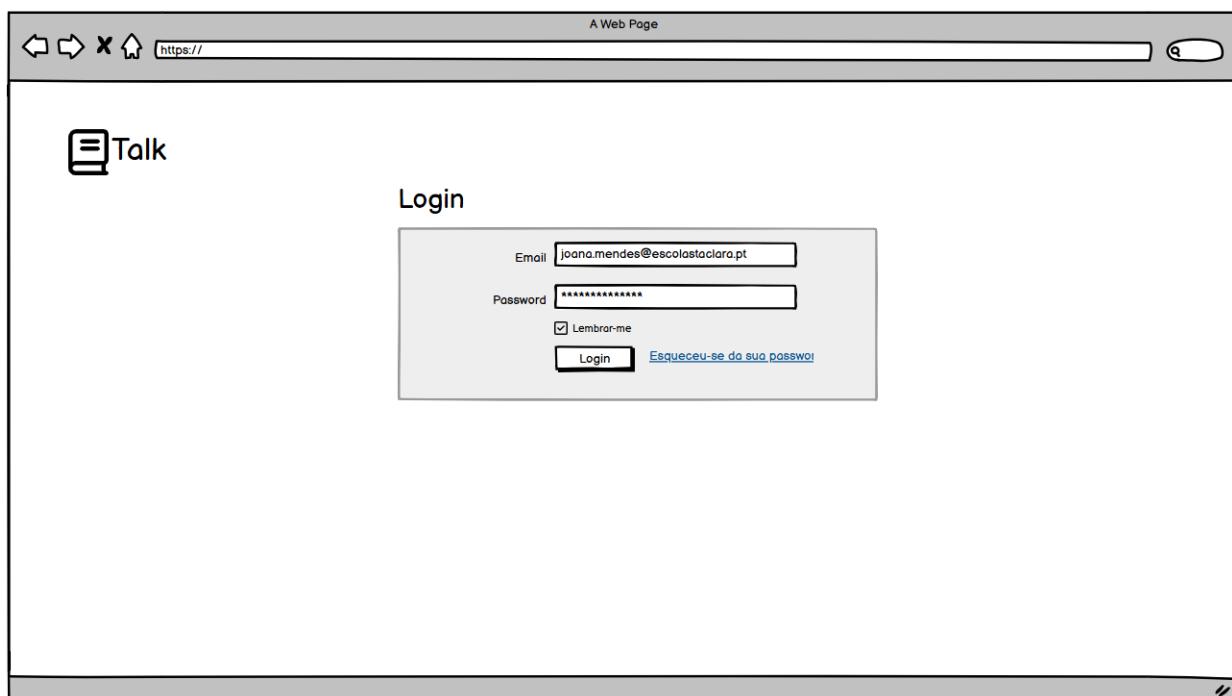


Figura 26 - Login como bibliotecário

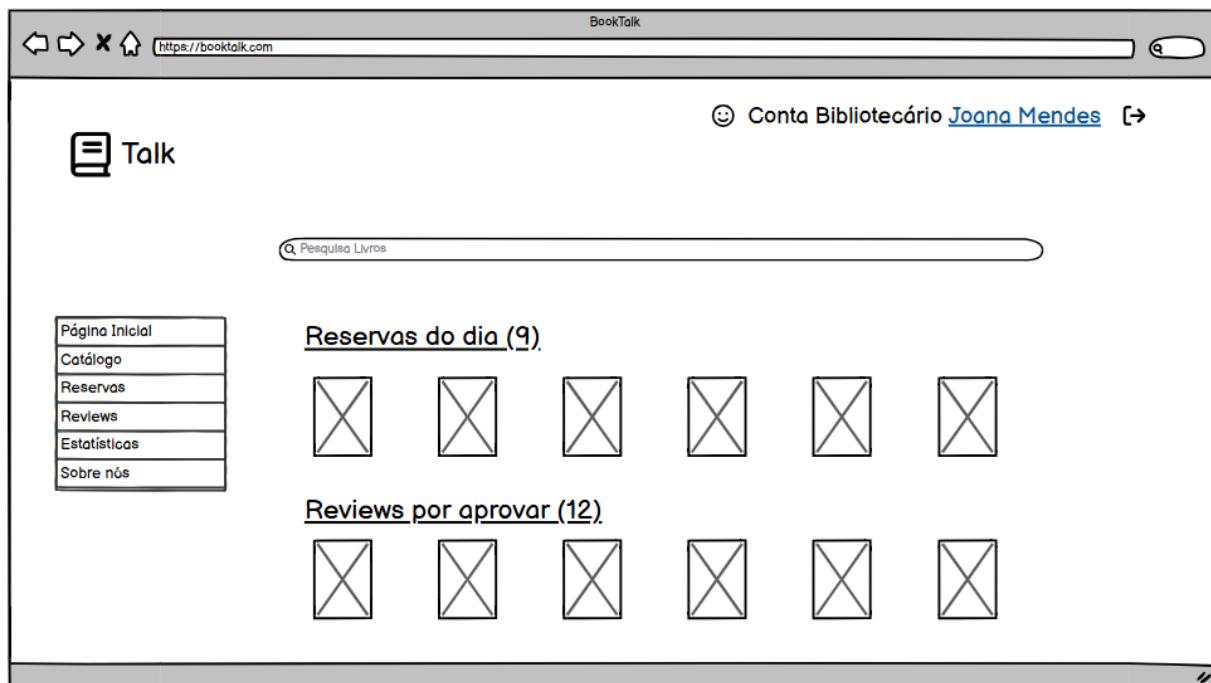


Figura 27 - Página inicial do bibliotecário

This screenshot shows the "Adicionar ao Catálogo" (Add to Catalog) page. On the left is a sidebar with links: "Página Inicial", "Pesquisa de Livros", "Gestão de Catálogo", "Gestão de Reservas", "Gestão de Reviews", "Relatórios Estatísticos", and "Sobre nós". The main form has fields for "Título", "Autores", "Sinopse", "Editora", "Categoria", "Data de Publicação", "Idioma", and "ISBN", each with an associated input box. To the right, there's a "Imagen de Capa" (Cover Image) section with a dashed box for dragging files, containing a cloud icon with an upward arrow and the text "Escolher arquivo ou arrastar e largar aqui". At the bottom right is a large "Adicionar ao Catálogo" button.

Figura 28 - Adicionar título

### 3.2.3.2 Tarefa 2 – Reservar um livro

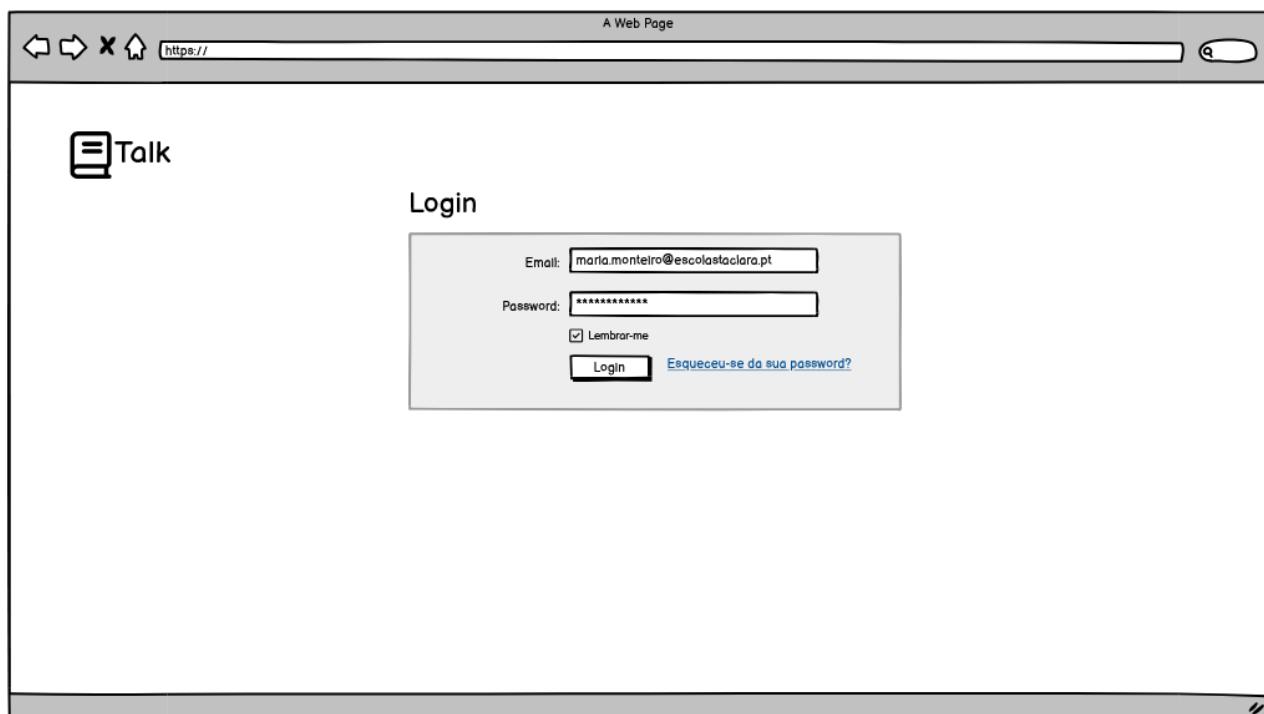


Figura 29 - Página de Login

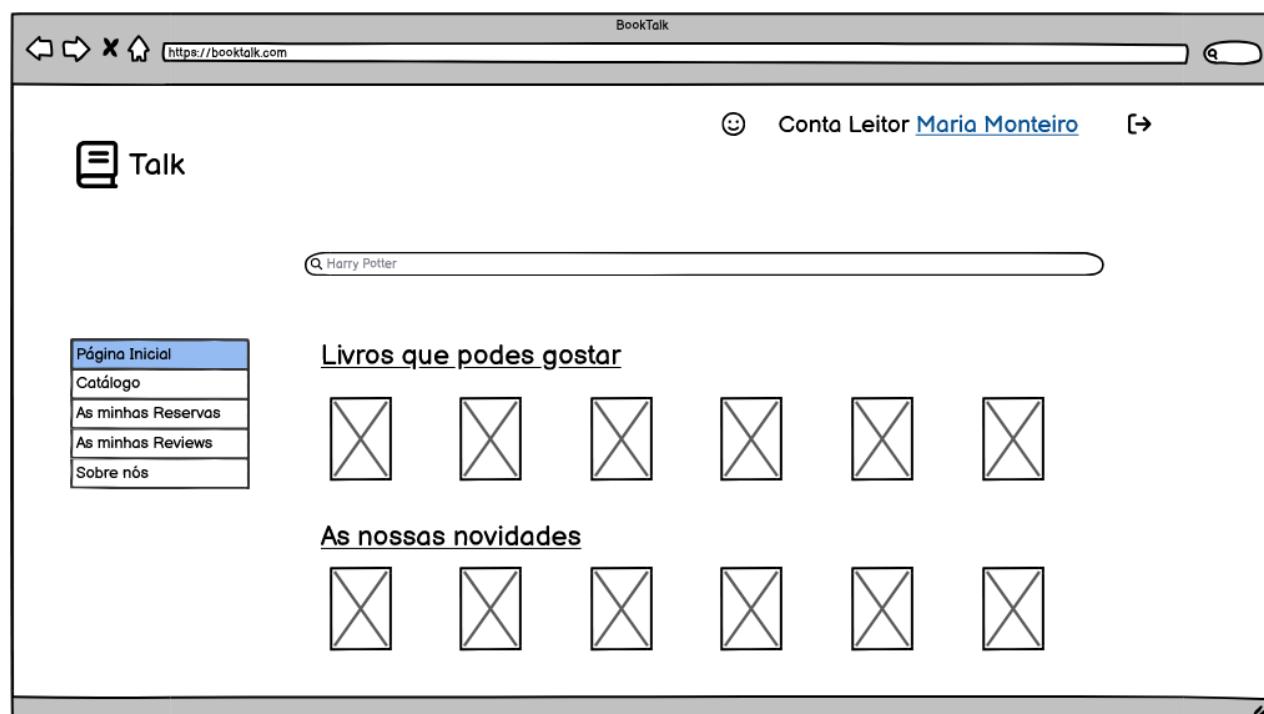


Figura 30 - Página Inicial (Leitor)

The screenshot shows a web browser window for the BookTalk website (<https://booktalk.com>). At the top right, there is a user profile for 'Conta Leitor Maria Monteiro'. Below the header, the word 'Talk' is displayed next to a document icon. A search bar contains the query 'Harry Potter'. On the left, a sidebar menu includes links for 'Página Inicial', 'Catálogo' (which is highlighted in blue), 'As minhas Reservas', 'As minhas Reviews', and 'Sobre nós'. The main content area displays three book entries, each with a small thumbnail image (a square with a large 'X') and the title: 'Harry Potter e a Pedra Filosofal', 'Harry Potter e a Câmara dos Segredos', and 'Harry Potter e o Prisioneiro de Azkaban'. To the right of these titles is a dropdown menu labeled 'Ordenar por'. The bottom right corner of the window has a double arrow icon.

Figura 31 - Resultados da Pesquisa

This screenshot shows a detailed view of the first Harry Potter book entry from Figure 31. The page title is 'BookTalk' at the top, followed by the user profile 'Conta Leitor Maria Monteiro'. The 'Talk' icon is present. A search bar at the top says 'Pesquisa de livros'. The sidebar menu is identical to Figure 31. The main content area shows the book 'Harry Potter e a Pedra Filosofal' by J. K. Rowling. It includes a brief synopsis: 'Harry Potter descobre no seu 11.º aniversário que é um feiticeiro e vai estudar na escola de magia Hogwarts. Lá, faz novos amigos,' and a 'Ler +' button. Below the synopsis are buttons for 'Reservar' and 'Avaliar e Escrever Review'. To the left of the synopsis, there is additional book information: 'Editora: Presença', 'Categorias: Magia, Aventura, Amizade', 'Data de publicação: Outubro de 1999', 'Língua: Português', and 'ISBN: 9789722365598'. At the bottom, there is a section for '2 Reviews' with a rating of 'Maria Cerejeira - 3/5' and a note 'Há 4 mês'.

Figura 32 - Página de detalhes do livro

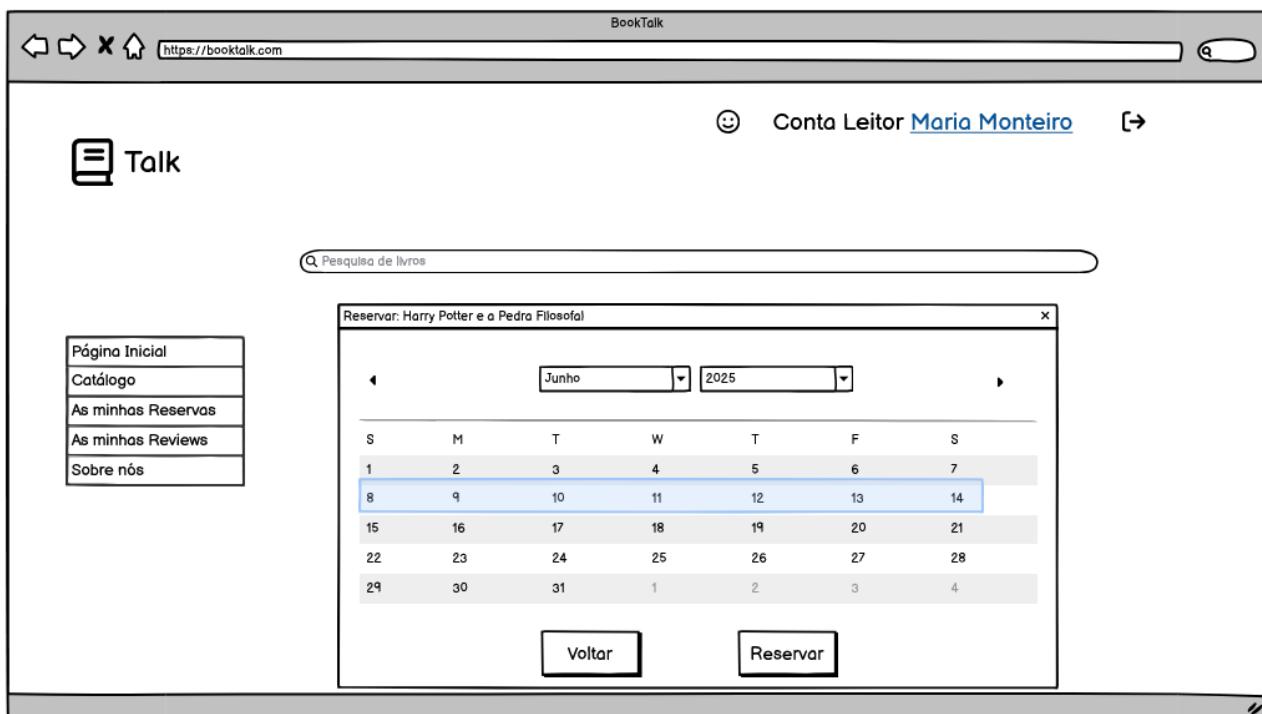


Figura 33 - Página de Reserva com Calendário

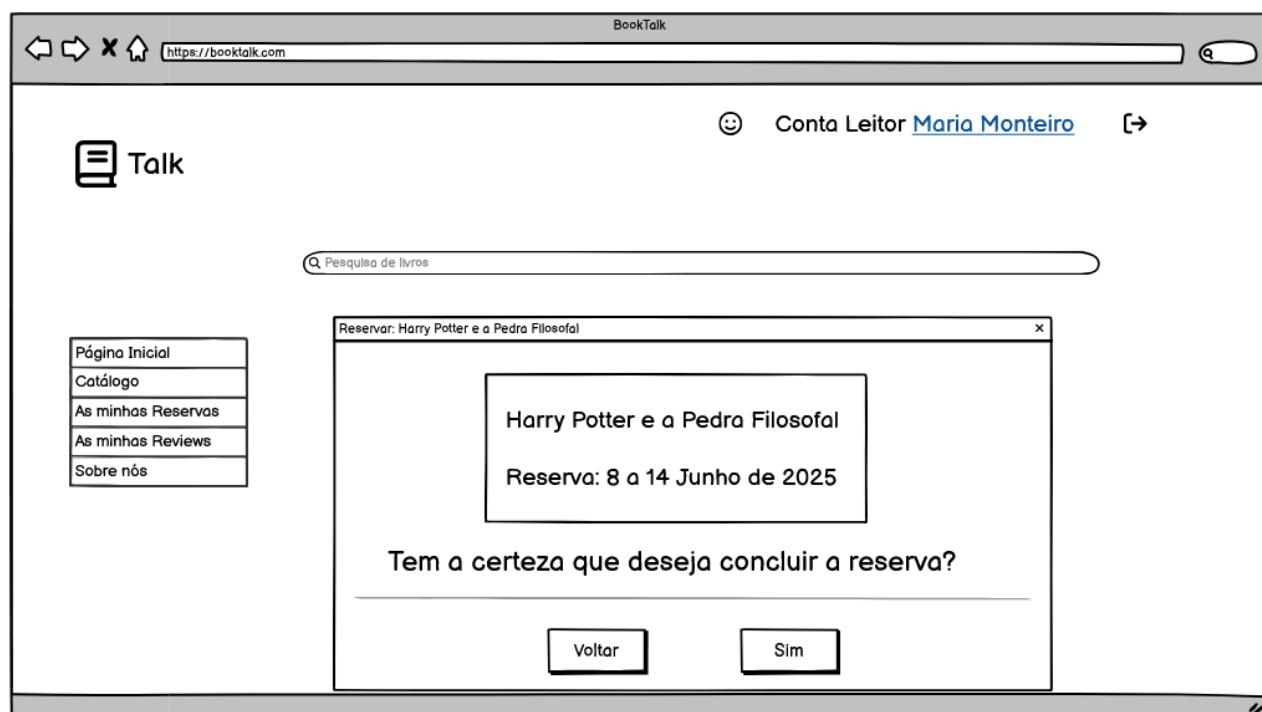


Figura 34 - Confirmação da Reserva

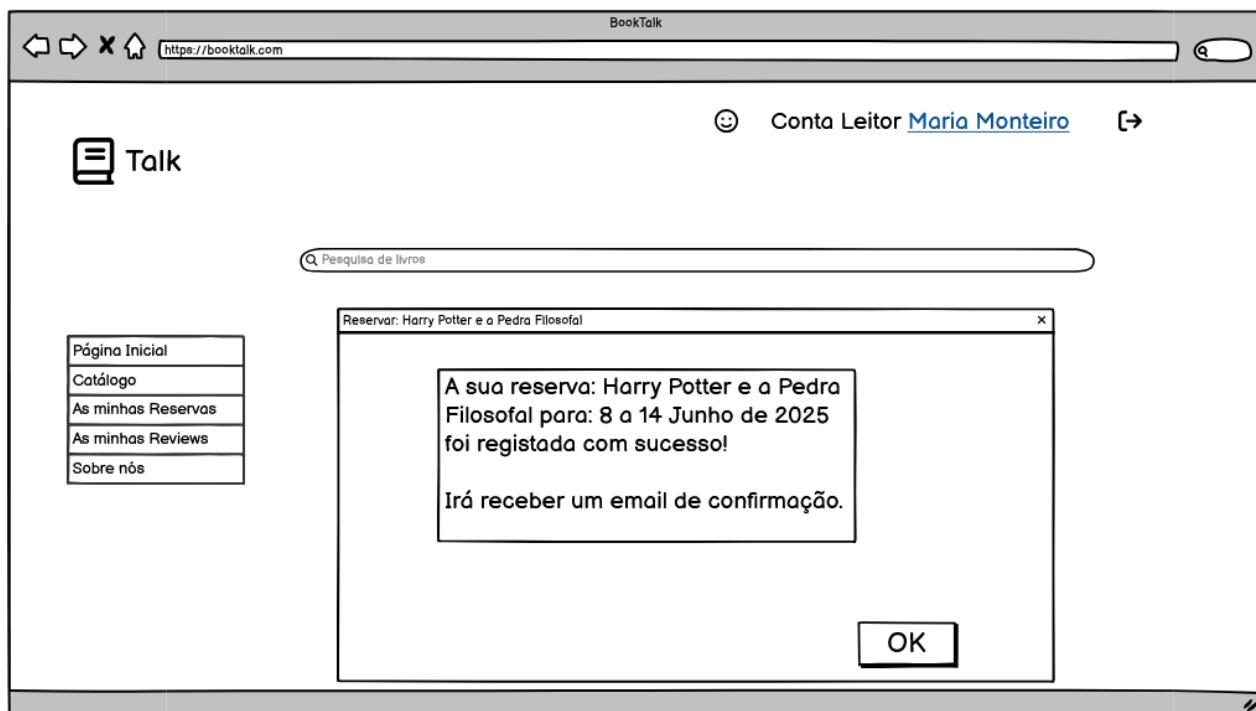


Figura 35 - Resumo da Reserva

The screenshot shows the same 'BookTalk' website interface as Figure 35. The user 'Maria Monteiro' is still logged in. The main content area displays the book 'Harry Potter e a Pedra Filosofal' by J. K. Rowling. The book cover is shown with a large 'X' over it. Below the cover, the synopsis reads: 'Harry Potter descobre no seu 11º aniversário que é um feiticeiro e vai estudar na escola de magia Hogwarts. Lá, faz novos amigos,' followed by a 'Ler +' button. To the left of the book details is a sidebar menu. At the bottom of the page, there is a section for reviews with a heading '2 Reviews' and a review from 'Maria Cerejeira' with a rating of '3/5' and the comment 'Há 4 meses'.

Figura 36 - Detalhes do Livro após Reserva

### 3.2.3.3 Tarefa 3 – Avaliar e escrever uma review sobre um livro

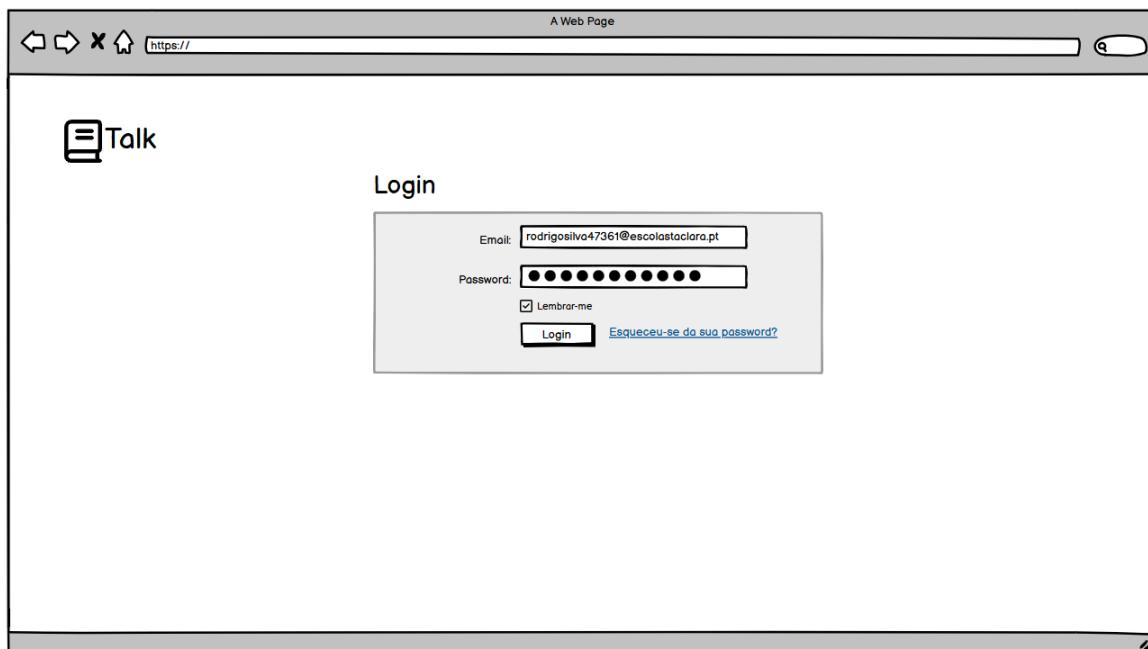


Figura 37 - Login como leitor

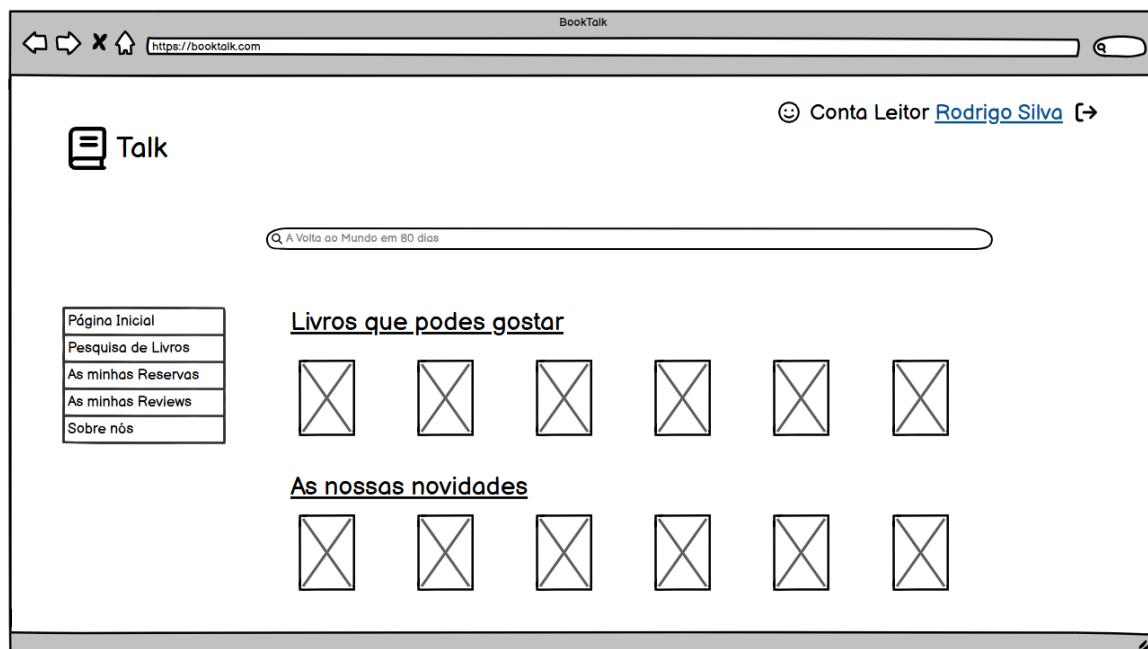


Figura 38 - Página inicial do leitor

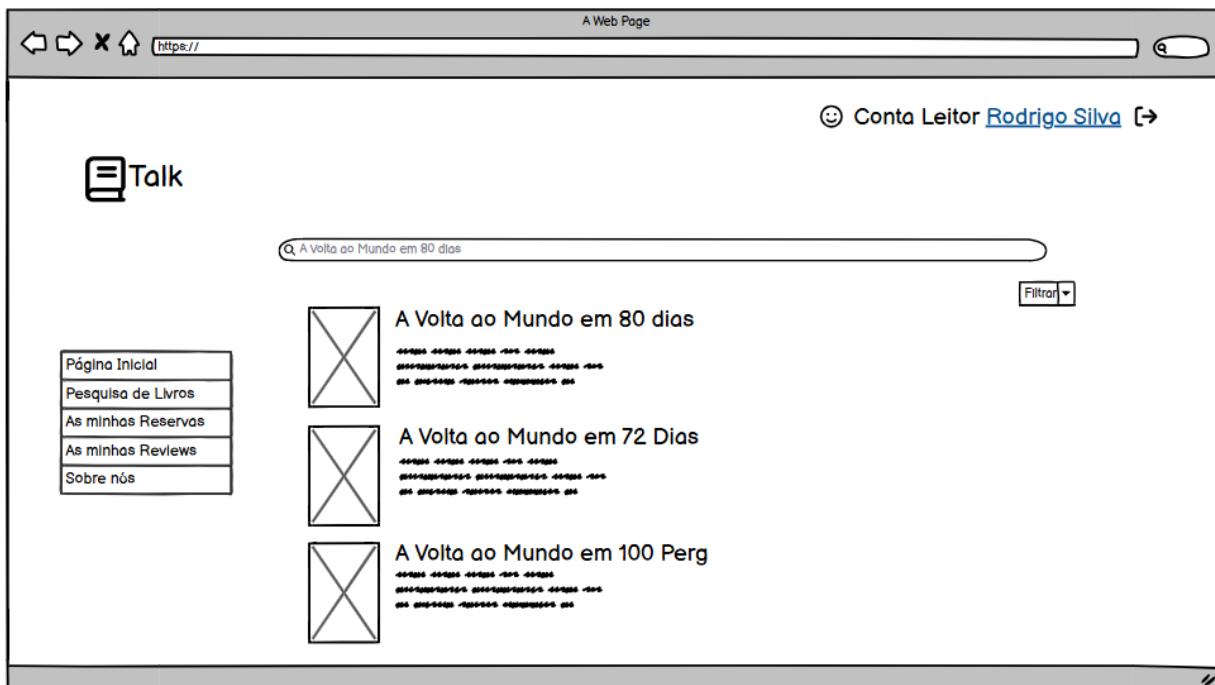


Figura 39 - Pesquisa de um livro

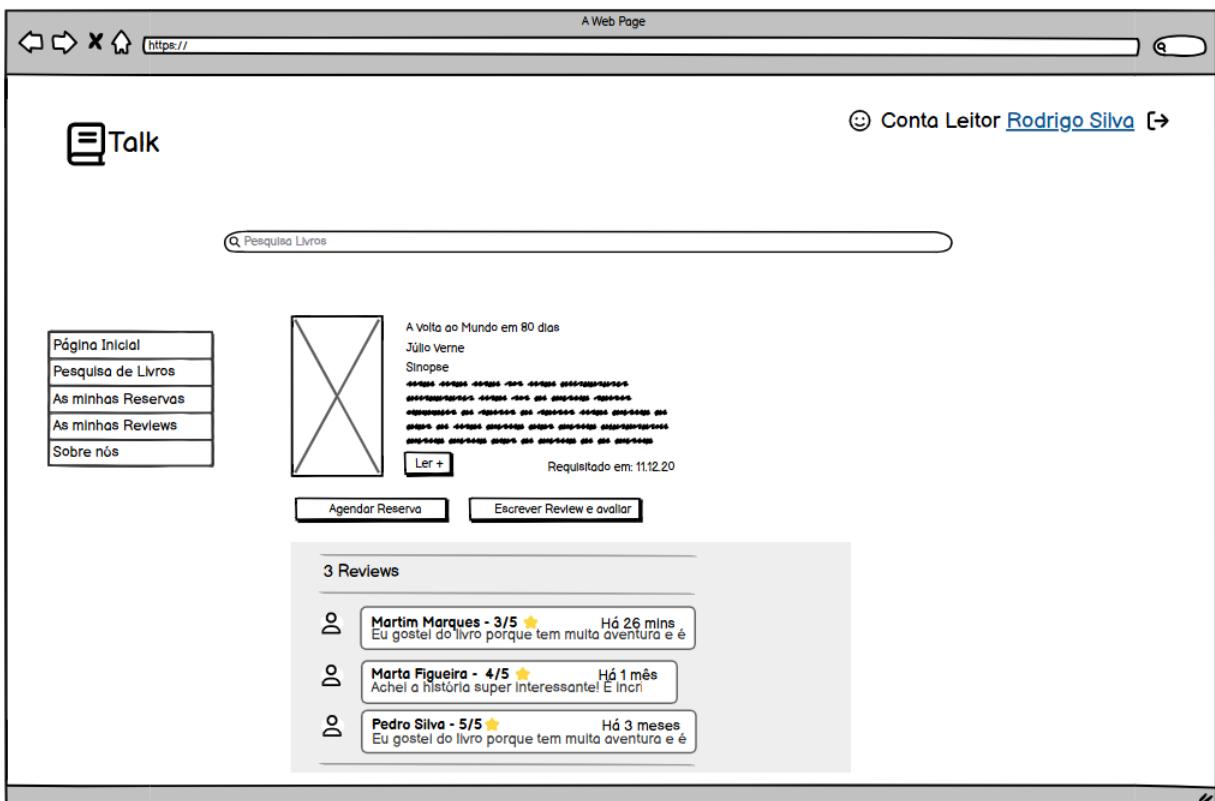


Figura 40 - Detalhes do livro

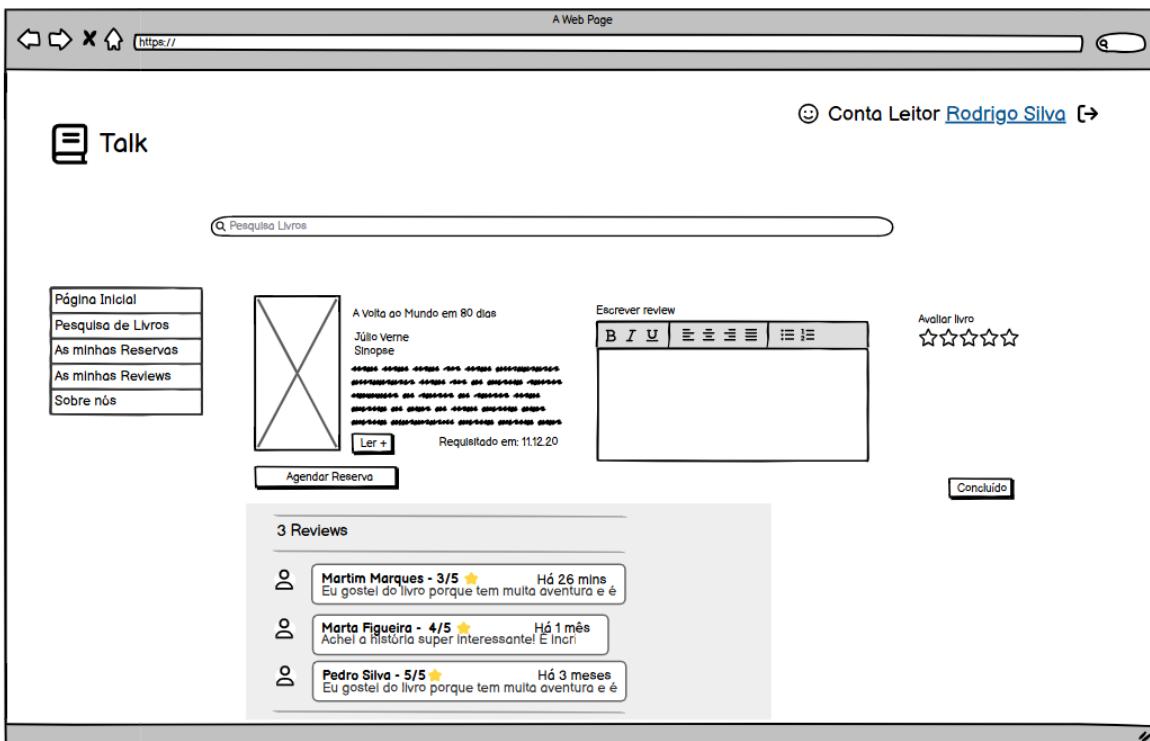


Figura 41 - Escrever review e avaliar um livro

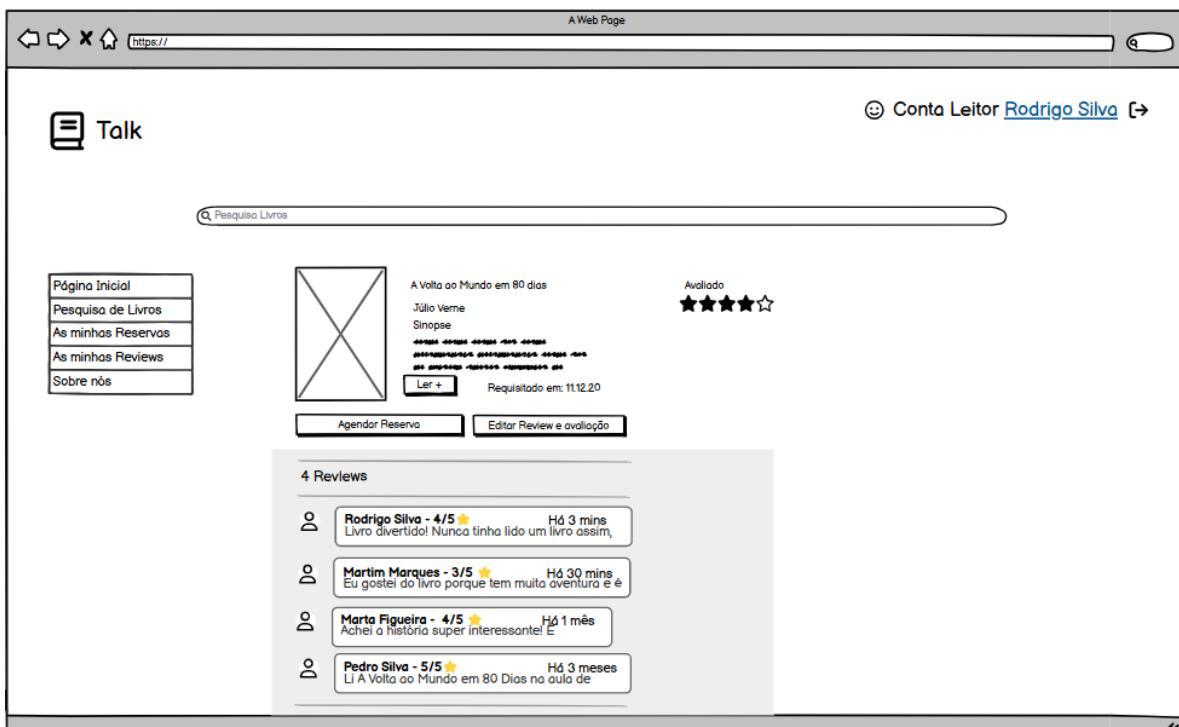


Figura 42 - Após submeter a review e a avaliação

### 3.2.3.4 Tarefa 4 – Validar uma review

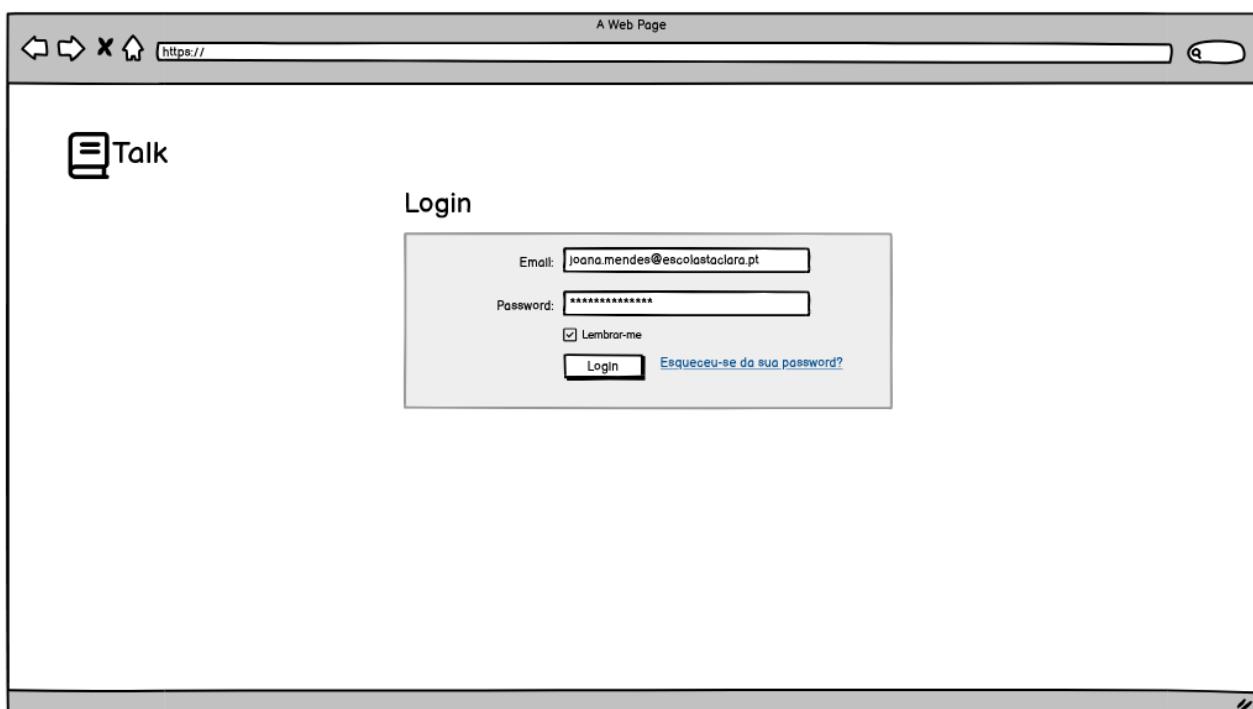


Figura 43 - Página de Login

A screenshot of a web browser window titled "BookTalk". The address bar shows "https://booktalk.com". The header includes the BookTalk logo and a link to "Conta Bibliotecário Joana Mendes". The main content area features a search bar labeled "Pesquisa Livros". On the left, a sidebar menu lists "Página Inicial", "Catálogo", "Reservas", "Reviews", "Estatísticas", and "Sobre nós". Below the sidebar, there are two sections: "Reservas do dia (9)" showing 9 crossed-out boxes, and "Reviews por aprovar (12)" showing 12 crossed-out boxes.

Figura 44 - Página Inicial da Conta Bibliotecário

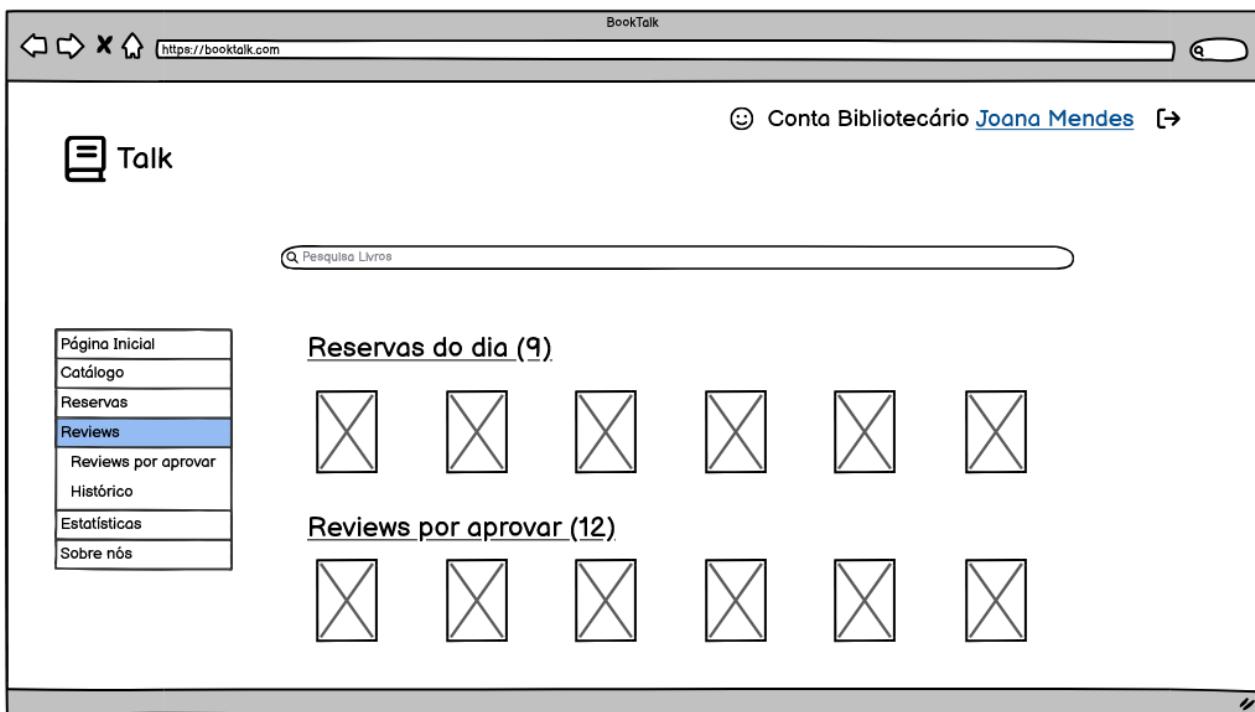


Figura 45 - Acesso à seção "Reviews por aprovar"

The screenshot shows a web browser window for 'BookTalk' at the URL <https://booktalk.com>. The user is logged in as 'Conta Bibliotecário Joana Mendes'. The main menu on the left includes 'Página Inicial', 'Catálogo', 'Reservas', 'Reviews' (selected), 'Reviews por aprovar', 'Histórico', 'Estatísticas', and 'Sobre nós'. The central area displays a list titled 'Reviews por aprovar (12)'. It shows two entries, each with a small square icon with an 'X':

- Volta ao Mundo em 80 dias  
"Livro divertido" por Rodrigo Silva
- Harry Potter e a Pedra Filosofal  
"Adorei este mundo" por Maria Monteiro

A dropdown menu labeled 'Ordenar por' is visible on the right side of the list.

Figura 46 - Lista de "Reviews por aprovar"

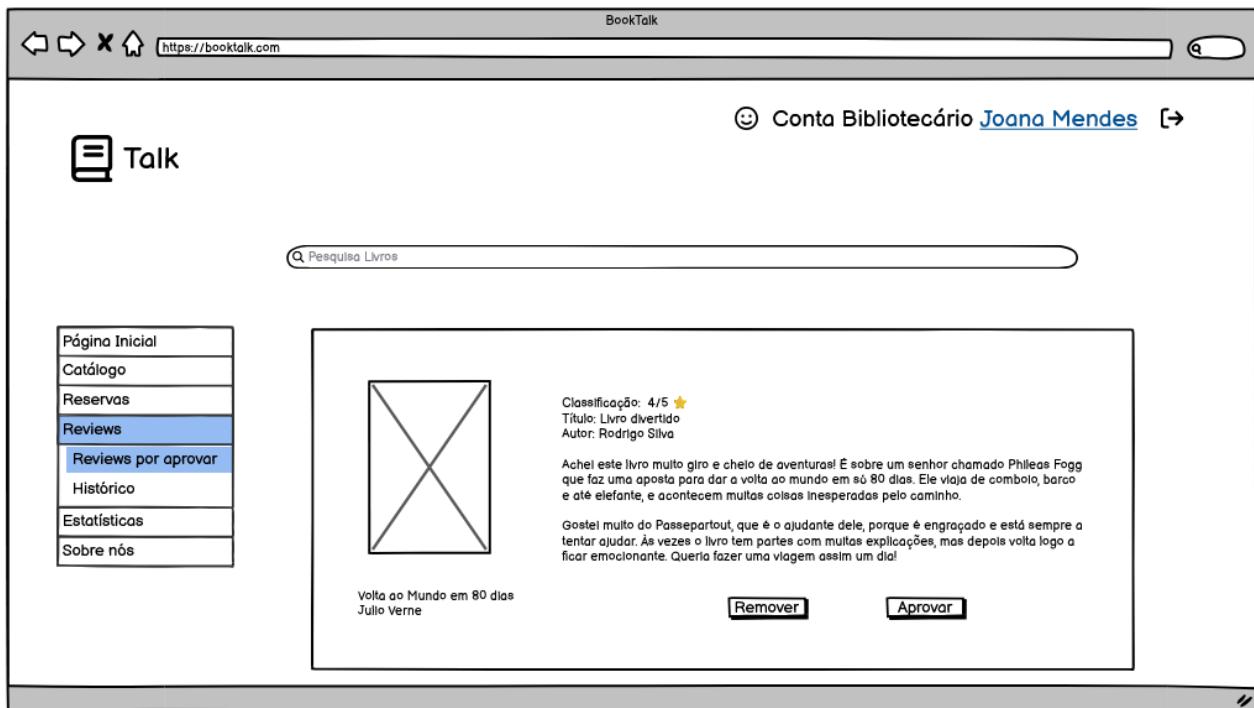


Figura 47 - Moderar e Validar Review (popup da review selecionada)

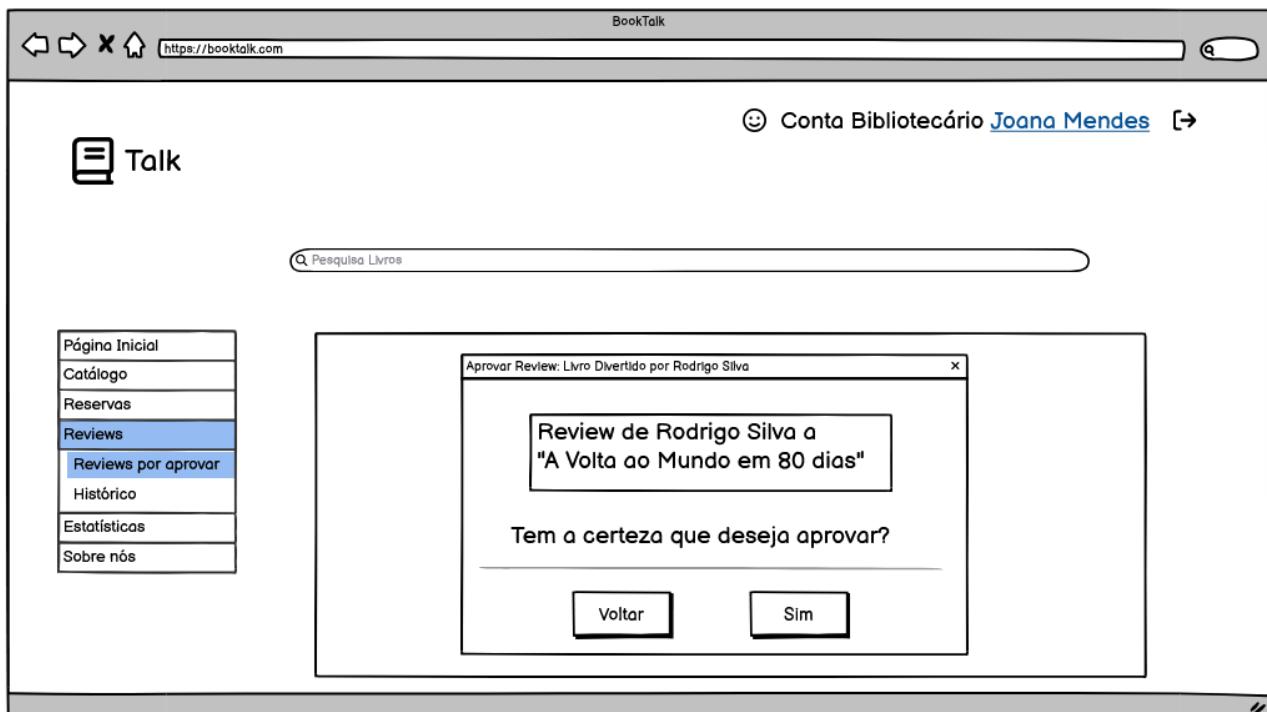


Figura 48 - Confirmação da Aprovação de Review

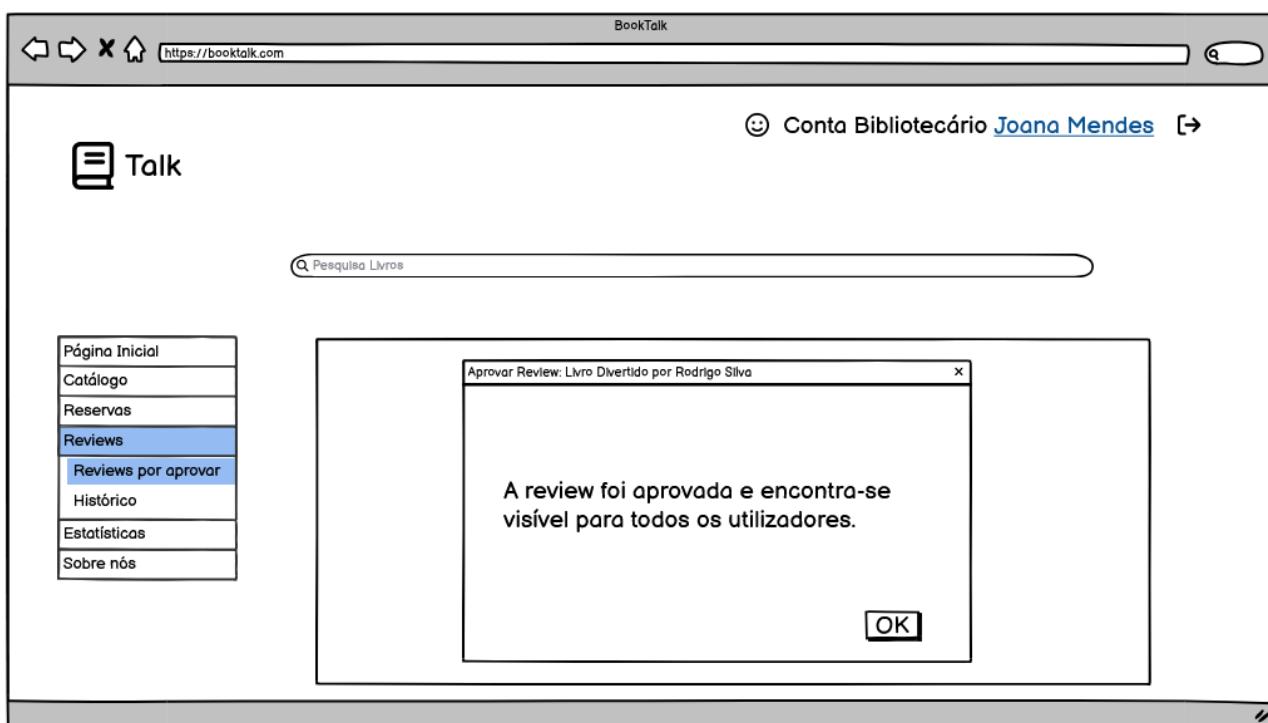


Figura 49 - Resumo da ação

The screenshot shows the 'Reviews por aprovar' section of the BookTalk application. The sidebar menu is identical to Figure 49. The main area displays a list titled 'Reviews por aprovar (11)'. Two reviews are listed with large 'X' icons next to them:

- Harry Potter e a Pedra Filosofal  
Adorei este mundo por Maria Monteiro
- Diário de um Banana 2  
"Diverti-me muito" por Teresa Barros

A dropdown menu labeled 'Ordenar por' is visible on the right side of the list.

Figura 50 - Reviews por aprovar (após ter aprovado uma da lista)

### 3.2.3.5 Tarefa 5 – Consultar painel de estatística (Data Analytics)

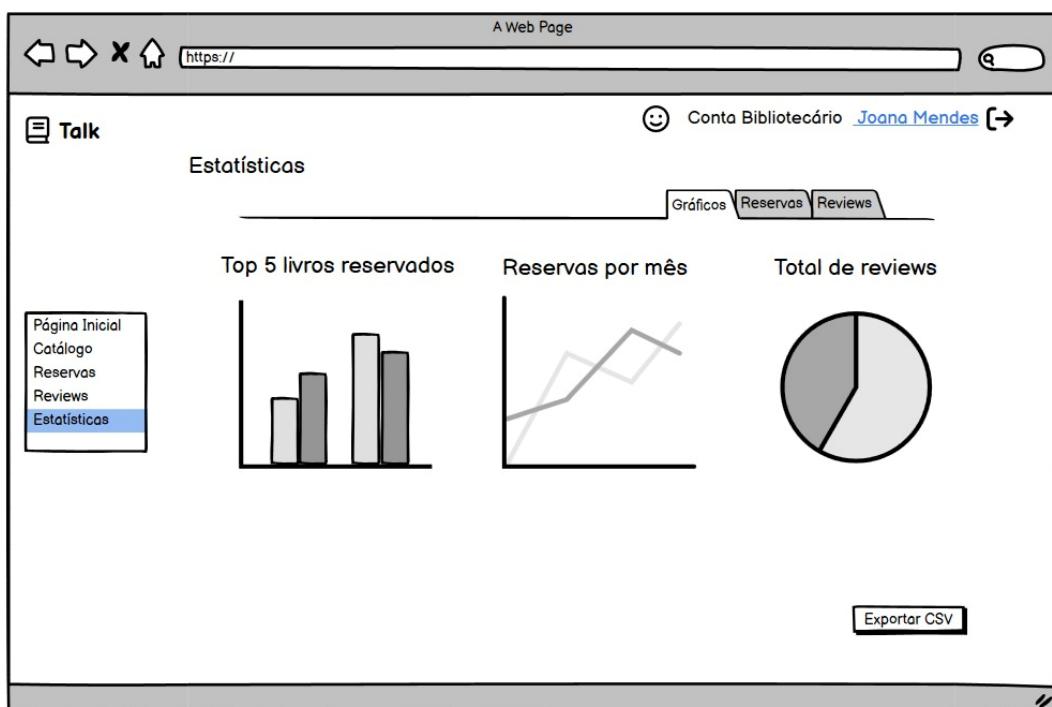


Figura 51 - Página Estatísticas - Gráficos

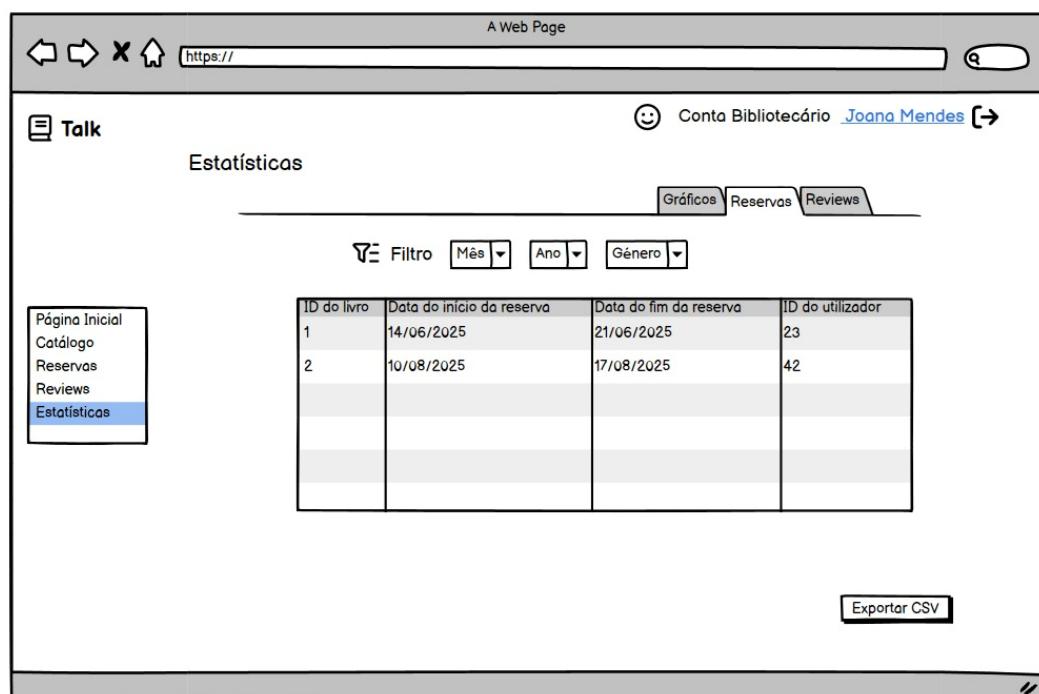


Figura 52 - Página Estatísticas - Reservas

A Web Page

https://

Talk

Estatísticas

Gráficos Reservas Reviews

Filtro Mês Ano Género

ID do livro	Data da review	Classificação	ID do utilizador
4	13/05/2025	4	16
7	12/07/2025	5	22

Exportar CSV

Figura 53 - Página Estatísticas - Reviews

## 4 Melhorias efetuadas na análise e desenho do sistema

Na sequência da apresentação da fase de análise e desenho do sistema ao docente, foram identificadas várias oportunidades de melhoria com o objetivo de tornar o sistema mais completo, flexível e alinhado com um cenário real de utilização. As principais sugestões apresentadas incidiram sobre a modelação dos dados, os fluxos de interação do utilizador e o tipo de informação disponibilizada no módulo de estatísticas, nomeadamente:

- Introdução de novas entidades no modelo de dados, como Género, Editora e Autor, de forma a enriquecer a representação do catálogo
- Alteração do caso de uso de adicionar livro, permitindo associar múltiplos autores e múltiplos géneros a um único livro através da interface
- Reformulação do processo de reservas, passando a permitir uma única reserva com vários livros, recorrendo a um mecanismo de carrinho de reservas e a um passo final de *checkout*
- Ajuste do módulo de estatísticas para apresentar dados simples e diretos, privilegiando indicadores de utilização em detrimento de lógica de negócio complexa

Todas estas sugestões foram analisadas e posteriormente implementadas na fase de desenvolvimento. O modelo de dados foi expandido para incluir as novas entidades propostas, bem como as respetivas relações, permitindo uma gestão mais completa do catálogo. A interface de adição de livros foi adaptada para suportar a criação e associação dinâmica de autores, géneros e editoras, simplificando o trabalho do bibliotecário.

Ao processo de reservas foi acrescentado um sistema de carrinho que permite ao leitor selecionar até três livros antes de confirmar a operação, garantindo uma experiência de utilização mais fluida e coerente. Por fim, o módulo de estatísticas foi ajustado para apresentar indicadores como os livros mais clicados, os livros mais requisitados, os períodos do dia com maior atividade e o tempo total de utilização por cada leitor, indo ao encontro das orientações definidas pelo docente. Estas melhorias resultaram num sistema mais robusto, funcional e alinhado com os objetivos do projeto.

# 5 Implementação

Neste capítulo são descritas as principais decisões e ações desenvolvidas na parte da implementação técnica deste projeto.

## 5.1 Arquitetura do sistema

A implementação do sistema BookTalk assenta numa arquitetura do tipo cliente-servidor, organizada de forma a garantir uma separação clara entre a interface de utilizador, a lógica de negócio e a persistência de dados. Esta abordagem permite uma maior organização do código, facilita a manutenção do sistema e torna mais simples a sua evolução futura.

O acesso ao sistema é realizado através de um browser web, utilizado tanto por leitores como por bibliotecários. Este browser comunica com a aplicação frontend desenvolvida segundo o padrão MVC, responsável pela apresentação da informação e pela interação com o utilizador.

A aplicação frontend não acede diretamente à base de dados, comunicando exclusivamente com a API REST do sistema. Esta API centraliza toda a lógica de negócio, validações e regras associadas às funcionalidades do BookTalk, como a gestão de livros, reservas, reviews e estatísticas.

A persistência dos dados é assegurada por uma base de dados relacional SQL, onde são armazenadas as entidades principais do sistema. A comunicação entre a API e a base de dados é realizada através do Entity Framework Core, garantindo consistência e integridade da informação.

Esta arquitetura, representada no diagrama de implantação apresentado na figura 54, permite uma clara separação de responsabilidades entre os diferentes componentes do sistema, contribuindo para a robustez e organização da solução desenvolvida.

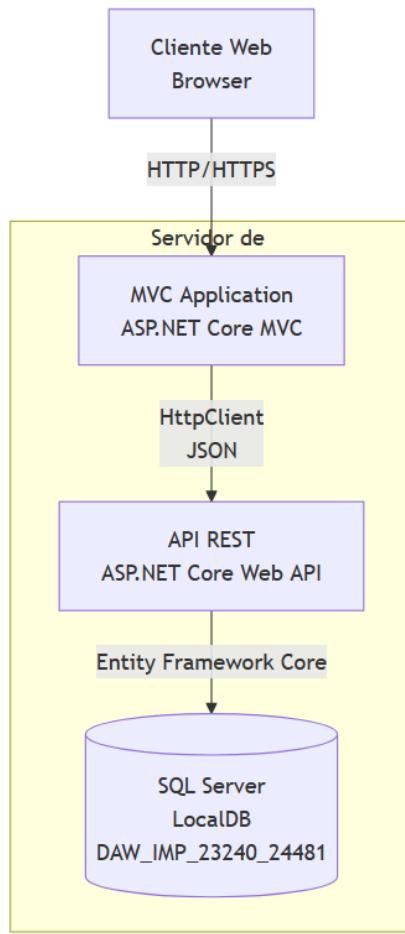


Figura 54 - Diagrama de implantação

## 5.2 Tecnologias usadas

A aplicação foi desenvolvida utilizando a plataforma .NET, servindo de base tanto para a API REST como para a aplicação frontend. A camada de apresentação foi implementada com ASP.NET Core MVC, recorrendo a vistas Razor para a construção da interface gráfica e ao framework Bootstrap para garantir uma apresentação responsiva e intuitiva.

A API REST foi desenvolvida com ASP.NET Core Web API, sendo responsável pela exposição dos endpoints e pela gestão da lógica de negócio do sistema. Para o acesso e persistência de dados foi utilizado o Entity Framework Core, adotando a abordagem Code-First, o que permitiu uma evolução controlada do esquema da base de dados através de migrations.

A base de dados utilizada foi o SQL Server LocalDB, adequada ao contexto académico e de desenvolvimento local.

O desenvolvimento do sistema BookTalk recorreu às seguintes tecnologias e frameworks:

### **ASP.NET Core MVC**

- **Descrição:** Framework web para construção de aplicações MVC (Model-View-Controller)
- **Versão utilizada:** 10.0.2
- **Uso no projeto:** Desenvolvimento da aplicação frontend com vistas Razor
- **Referência:** [ASP.NET Core MVC Documentation](#)

### **ASP.NET Core Web API**

- **Descrição:** Framework para construção de APIs RESTful
- **Versão utilizada:** 10.0.2 (incluída no .NET 10.0)
- **Uso no projeto:** Desenvolvimento da API REST que expõe os endpoints do sistema
- **Referência:** [ASP.NET Core Web API Documentation](#)

### **Entity Framework Core**

- **Descrição:** ORM (Object-Relational Mapping) para acesso a bases de dados
- **Versão utilizada:** 10.0.2
- **Uso no projeto:**
  - Gestão de acesso à base de dados SQL Server
  - Migrations para criação e atualização do esquema da base de dados
  - Gestão de relações entre entidades
- **Referência:** [Entity Framework Core Documentation](#)

### **SQL Server LocalDB**

- **Descrição:** Versão local e leve do SQL Server para desenvolvimento
- **Versão utilizada:** LocalDB (incluída no Visual Studio)
- **Uso no projeto:** Base de dados relacional para persistência de dados
- **Referência:** [SQL Server LocalDB Documentation](#)

## Bootstrap

- **Descrição:** Framework CSS para desenvolvimento de interfaces responsivas
- **Versão utilizada:** 5
- **Uso no projeto:** Estilização da interface de utilizador e componentes visuais
- **Referência:** [Bootstrap Documentation](#)

## Bootstrap Icons

- **Descrição:** Biblioteca de ícones para Bootstrap
- **Versão utilizada:** 1.11.3
- **Uso no projeto:** Ícones na interface de utilizador
- **Referência:** [Bootstrap Icons Documentation](#)

## 5.3 Desenvolvimento da API

A API REST foi desenvolvida seguindo os princípios RESTful e boas práticas de desenvolvimento de APIs. Esta secção descreve a estrutura, especificação e principais decisões de implementação.

### 5.3.1 Especificação da interface

A API REST do sistema BookTalk permite a comunicação entre o frontend e a base de dados de forma estruturada e segura.

A API segue os princípios REST, utilizando os métodos HTTP adequados para cada operação, nomeadamente GET, POST, PUT e DELETE. Os endpoints encontram-se organizados por recursos, tais como livros, reservas, reviews e utilizadores, facilitando a compreensão e manutenção da interface.

Todas as respostas da API são devolvidas em formato JSON e utilizam códigos de estado HTTP normalizados, permitindo ao frontend interpretar corretamente o resultado de cada operação. A especificação completa dos endpoints encontra-se documentada no Anexo I. No Anexo II encontra-se a documentação Swagger, que pode ser acedida através da rota /swagger da API (e.g: <https://localhost:7148/swagger/index.html>).

### 5.3.2 Decisões de implementação

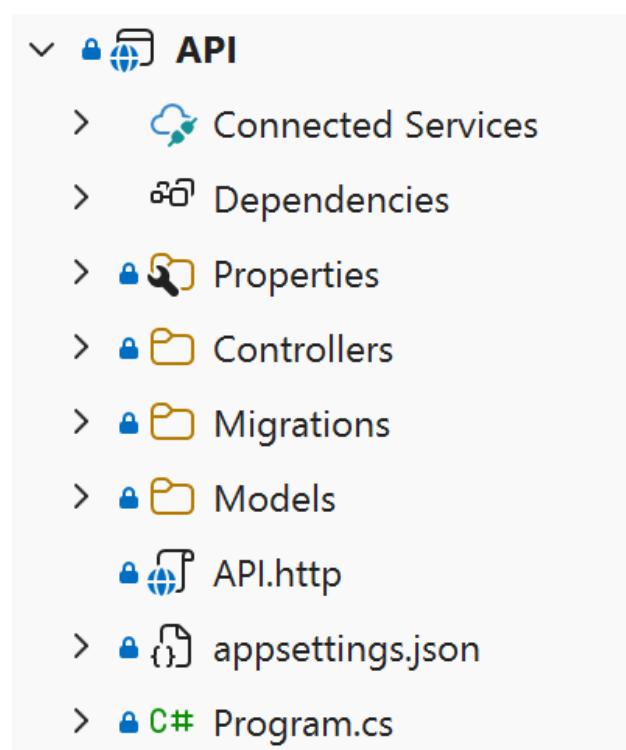


Figura 55 - Estrutura da API

#### Abordagem Code First

Foi utilizada a abordagem Code-First do Entity Framework Core para a gestão da base de dados. Esta abordagem revelou-se adequada ao contexto do projeto, uma vez que permitiu um desenvolvimento mais ágil, facilitando a criação e alteração dos modelos de dados diretamente no código, sem necessidade de recorrer a scripts SQL manuais. Além disso, a utilização de migrations possibilitou que todas as alterações ao modelo fossem refletidas de forma automática, garantindo a sincronização contínua entre os modelos e base de dados.

#### Definição de rotas

A API segue a convenção padrão do ASP.NET Core, utilizando o prefixo `api/` seguido do nome do controlador. Esta opção permite uma organização clara dos endpoints, alinhada com os princípios REST, tornando a estrutura do projeto mais intuitiva e fácil de compreender. Em situações específicas, em que as operações não se enquadram diretamente nas ações CRUD tradicionais, foram definidas rotas personalizadas, como a confirmação de reservas, a aprovação ou rejeição de reviews e a consulta de estatísticas do sistema.

## **Controladores da API**

Os controladores da API foram implementados de forma uniforme, seguindo um padrão comum em todo o projeto, no qual todos herdam de ControllerBase, uma vez que a API não necessita de funcionalidades associadas à apresentação de vistas. Cada entidade principal do sistema possui o seu próprio controlador, o que contribui para uma melhor organização e separação de responsabilidades.

Os métodos definidos em cada controlador são públicos e assíncronos, permitindo uma gestão mais eficiente das operações de acesso a dados. A nomenclatura dos métodos foi escolhida de forma clara e consistente, refletindo diretamente a funcionalidade que executam, como a obtenção, criação, atualização ou remoção de recursos. Esta abordagem facilita a leitura e compreensão do código.

## **Modelos de dados**

No que diz respeito aos modelos de dados, foram utilizadas validações através de Data Annotations, permitindo definir regras diretamente nas entidades do sistema. Esta abordagem assegura que apenas dados válidos são persistidos na base de dados, reforçando a integridade da informação. Foram aplicadas validações como obrigatoriedade de campos, limitação do tamanho de strings, definição de intervalos válidos para determinados valores e especificação explícita de relações entre entidades.

Por fim, as relações do tipo *many-to-many* existentes no sistema, nomeadamente entre livros e autores, livros e géneros, e reservas e livros, foram geridas automaticamente pelo Entity Framework Core através de tabelas de junção. Esta solução simplificou a implementação, evitando a necessidade de criar manualmente entidades intermédias, e garantiu uma gestão consistente das associações entre os diferentes elementos do sistema.

### **5.3.3 Principais casos relevantes de programação**

Explicação dos principais e mais complexos casos de codificação, exemplificando com pequenos trechos de código que permitam a um outro programador no futuro perceber esses casos.

#### **Gestão de Relações Many-to-Many**

A criação e atualização de livros requer gestão cuidadosa das relações *many-to-many* com autores e géneros. O excerto de código abaixo ilustra a forma como são geridas as relações *many-to-many* entre livros, autores e géneros no momento da criação de um novo livro. Inicialmente, o livro é criado na base de dados sem associações para obter o seu id. De seguida, são carregadas as entidades relacionadas existentes (autores e géneros) e estabelecidas as respectivas associações.

```
LivrosController.cs API API.Controllers.LivrosController PostLivro(Livro livro)

72
73     // Handle many-to-many relationships
74     var autoresIds = livro.Autores?.Select(a => a.IdAutor).ToList() ?? new List<int>();
75     var generosIds = livro.Generos?.Select(g => g.IdGenero).ToList() ?? new List<int>();
76
77     if (livro.Autores != null) livro.Autores.Clear();
78     if (livro.Generos != null) livro.Generos.Clear();
79
80     context.Livros.Add(livro);
81     await context.SaveChangesAsync();
82
83     // Add autores
84     if (autoresIds.Any() && livro.Autores != null)
85     {
86         var autores = await context.Autores.Where(a => autoresIds.Contains(a.IdAutor)).ToListAsync();
87         foreach (var autor in autores)
88         {
89             livro.Autores.Add(autor);
90         }
91     }
92
93     // Add generos
94     if (generosIds.Any() && livro.Generos != null)
95     {
96         var generos = await context.Generos.Where(g => generosIds.Contains(g.IdGenero)).ToListAsync();
97         foreach (var genero in generos)
98         {
99             livro.Generos.Add(genero);
100        }
101    }
102
103    await context.SaveChangesAsync();
104
```

Figura 56 - Gestão de many-to-many

## **Registo de visualizações de livros**

Sempre que um livro é consultado através do endpoint GET /api/livros/{id}, o sistema incrementa o valor da propriedade “Clicks” associado ao respetivo registo.

Esta informação é posteriormente utilizada na página de Estatísticas, permitindo identificar os livros mais visualizados pelos utilizadores e apoiar a análise da utilização do sistema. Esta abordagem garante que os dados estatísticos são recolhidos de forma transparente e automática, sem necessidade de intervenção adicional por parte do utilizador.

The screenshot shows the code editor for the `LivrosController.cs` file. The code increments a click counter for a book and saves changes.

```
// Increment click counter
livro.Clicks += 1;
context.Update(livro);
await context.SaveChangesAsync();
```

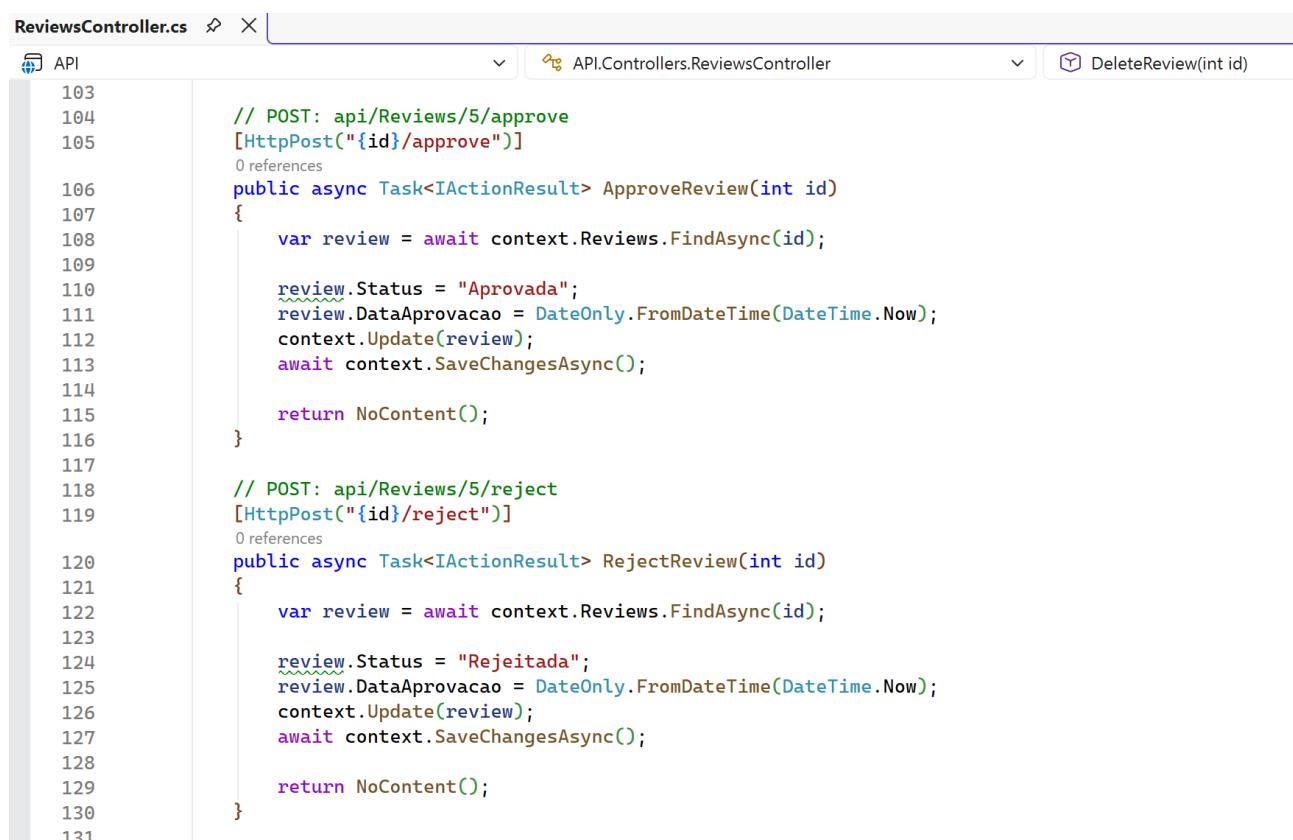
*Figura 57 - Incremento de visualizações do livro*

## Moderação de Reviews

As reviews têm três estados possíveis: "Pendente", "Aprovada" e "Rejeitada". A transição entre estados é feita através de endpoints específicos.

O endpoint POST api/Reviews/{id}/approve recebe o id de uma review e altera o campo "Status" para "Aprovada" e o campo DataAprovacao para a data atual na base de dados. À semelhança do anterior, o endpoint POST api/Reviews/{id}/reject recebe o id de uma review e altera o campo "Status" para "Rejeitada" e o campo DataAprovacao para a data atual na base de dados.

Apenas as reviews com o estado "Aprovada" serão exibidas na página de detalhes do livro.



```
103
104
105    // POST: api/Reviews/5/approve
106    [HttpPost("{id}/approve")]
107    public async Task<IActionResult> ApproveReview(int id)
108    {
109        var review = await context.Reviews.FindAsync(id);
110
111        review.Status = "Aprovada";
112        review.DataAprovacao = DateOnly.FromDateTime(DateTime.Now);
113        context.Update(review);
114        await context.SaveChangesAsync();
115
116        return NoContent();
117    }
118
119    // POST: api/Reviews/5/reject
120    [HttpPost("{id}/reject")]
121    public async Task<IActionResult> RejectReview(int id)
122    {
123        var review = await context.Reviews.FindAsync(id);
124
125        review.Status = "Rejeitada";
126        review.DataAprovacao = DateOnly.FromDateTime(DateTime.Now);
127        context.Update(review);
128        await context.SaveChangesAsync();
129
130        return NoContent();
131    }

```

Figura 58 - Endpoints para aprovar e rejeitar reviews

## Estatísticas Agregadas

O endpoint GET api/Home/statistics é responsável pela recolha, agregação e disponibilização dos dados estatísticos apresentados na página de Estatísticas da aplicação. Este endpoint centraliza toda a lógica de cálculo estatístico, permitindo que o frontend obtenha, numa única chamada, a informação necessária para a análise da utilização do sistema.

Em primeiro lugar, o sistema identifica os livros mais visualizados, ordenando os regtos pelo número de cliques associados a cada livro e limitando o resultado aos dez títulos mais consultados. Estes dados permitem perceber quais os livros que despertam maior interesse por parte dos utilizadores.

De seguida, são calculados os livros mais requisitados, com base nas reservas efetuadas. Para esse efeito, o sistema analisa todas as reservas existentes, agrupa os livros associados a cada reserva e contabiliza o número de ocorrências de cada título. O resultado é novamente ordenado de forma decrescente, permitindo identificar os livros mais frequentemente reservados.

Relativamente à análise de acessos, o endpoint processa os regtos de acesso dos utilizadores, considerando apenas eventos de início de sessão associados a utilizadores do tipo leitor. Os acessos são agrupados por hora e posteriormente agregados em períodos do dia, possibilitando a identificação dos momentos com maior atividade no sistema.

Para além da frequência de acessos, é também calculado o tempo total de utilização da aplicação por cada leitor. Este cálculo baseia-se na análise sequencial dos eventos de início e fim de sessão registados, permitindo determinar a duração efetiva das sessões de utilização. Utilizadores que não correspondem ao perfil de leitor, são excluídos desta análise, garantindo a relevância dos dados apresentados.

Toda a informação agregada é devolvida num único objeto de resposta, estruturado de forma a facilitar a sua apresentação gráfica e tabular no frontend. Esta abordagem permite ao bibliotecário obter uma visão global e detalhada da utilização do sistema, apoiando a análise do comportamento dos utilizadores e a tomada de decisões informadas.

## 5.4 Desenvolvimento da App frontend/MVC

A aplicação frontend do sistema BookTalk é responsável por apresentar a informação ao utilizador, recolher os seus inputs e encaminhar as operações para a API, sem aceder diretamente à base de dados. Esta app foi implementada em ASP.NET Core MVC, permitindo uma separação clara entre a lógica de apresentação, o controlo das interações e a gestão dos dados. Esta abordagem contribui para uma maior organização do código, facilita a manutenção da aplicação e torna mais simples de identificar onde está todo o código por trás de cada página.

### 5.4.1 Decisões de implementação

#### MVC Tradicional

A principal decisão arquitetural na implementação da aplicação frontend foi a adoção do padrão Model-View-Controller. Neste modelo, os controladores são responsáveis por receber os pedidos do utilizador, coordenar a comunicação com a API REST e selecionar a vista adequada a apresentar. As vistas, implementadas através de ficheiros Razor, constituem a interface gráfica da aplicação, enquanto os modelos e view models representam os dados utilizados nas interações com o utilizador.

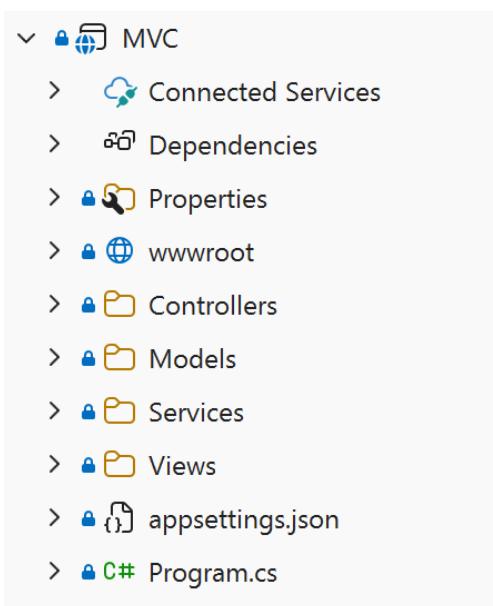


Figura 59 - Estrutura da app MVC

#### Comunicação com a API

A comunicação com a API REST é realizada de forma centralizada através de um serviço dedicado chamado ApiService. Este serviço recorre ao mecanismo IHttpClientFactory e responsável por encapsular toda a lógica de chamadas HTTP.

O ApiService inclui ainda mecanismos de tratamento de erros e de transformação dos dados em formato JSON, com registo de eventos relevantes através de logging, permitindo uma deteção mais fácil de falhas durante a comunicação com o backend. Para facilitar a reutilização e reduzir duplicação de código, foram implementados métodos genéricos para as operações mais comuns, nomeadamente pedidos do tipo GET, POST, PUT e DELETE. Esta abordagem contribui para uma maior clareza e simplicidade no código dos controladores da aplicação.

## Estrutura de Rotas

Para a navegação da aplicação, foi adotado o sistema de rotas convencional do ASP.NET Core MVC. Este modelo baseia-se na associação entre o nome do controlador, o método de ação a executar e um parâmetro opcional, permitindo definir URLs simples e facilmente compreensíveis.

A utilização deste padrão resulta numa estrutura consistente em toda a aplicação, facilitando a leitura e compreensão do fluxo de navegação e organização do projeto durante a implementação.

Alguns exemplos de rotas e respectivos métodos do controller:

- / - HomeController.Index()
- /Livros/Details/5 - LivrosController.Details(5)
- /Reservas/Create → ReservasController.Create()

## Definições dos Controladores

Os controladores da aplicação frontend foram implementados de forma consistente, herdando da classe Controller para permitir a utilização das funcionalidades associadas à renderização de vistas. A comunicação com a API REST é realizada através do serviço ApiService, garantindo uma clara separação entre a lógica de apresentação e o acesso aos dados. Estes seguem o padrão MVC com um controlador por página.

## Layouts

A aplicação frontend utiliza um sistema de layouts que garante uma apresentação visual consistente e uma navegação uniforme em todas as páginas. Foi definido um layout principal responsável pela estrutura base da interface, bem como layouts específicos para contextos particulares, como o processo de autenticação.

Os layouts utilizados na aplicação são os seguintes:

- **\_Layout.cshtml** - Layout principal da aplicação, integrando a barra de navegação e a barra lateral
- **\_LoginLayout.cshtml** - Layout simplificado utilizado nas páginas de login

## Partial Views

Para evitar duplicação de código e facilitar a manutenção da interface, foram criados vários componentes reutilizáveis, integrados nas vistas sempre que necessário. Estes componentes são incluídos dinamicamente nas vistas através de partial views, permitindo uma organização mais clara do código e garantindo que alterações visuais ou funcionais possam ser feitas num único local, refletindo-se automaticamente em toda a aplicação.

- **\_Navbar.cshtml** - Barra de navegação superior
- **\_Sidebar.cshtml** - Menu lateral de navegação
- **\_CartPartial.cshtml** - Visualização do carrinho de reservas
- **\_ReviewModal.cshtml** – Modal para criação e edição de reviews
- **\_ModerateReviewModal.cshtml** - Modal para moderação de reviews pelo bibliotecário

### 5.4.2 Principais casos relevantes de programação

#### Sistema de carrinho de reservas e criação de várias reservas

A aplicação implementa um sistema de carrinho de reservas que permite ao utilizador selecionar vários livros antes de proceder à criação efetiva das reservas. Esta funcionalidade melhora a experiência de utilização, possibilitando ao leitor planejar as suas reservas de forma conjunta, em vez de as efetuar individualmente.

A gestão do carrinho e do processo de confirmação é realizada pelo controlador responsável pelas reservas. No momento do *checkout*, o sistema recebe a lista de itens selecionados pelo utilizador e executa um conjunto de validações antes de proceder à criação das reservas. Estas validações garantem o cumprimento das regras de negócio definidas e a integridade dos dados.

O fluxo de funcionamento deste processo inclui:

- Validação da existência de itens no carrinho e verificação do limite máximo de reservas permitidas por operação
- Verificação da validade e disponibilidade de cada livro selecionado
- Validação dos intervalos de datas associados a cada reserva
- Deteção de possíveis sobreposições com reservas já existentes
- Criação individual de cada reserva através da API REST
- Devolução da lista de identificadores das reservas criadas com sucesso

Este processo assegura que apenas reservas válidas e consistentes são registadas no sistema, ao mesmo tempo que oferece uma experiência de utilização fluida e controlada. Apesar de as reservas serem criadas individualmente, a lógica de validação centralizada permite tratar o *checkout* como uma operação única do ponto de vista do utilizador.

```

ReservasController.cs  X  |
MVC  |  MVC.Controllers.ReservasController  |  Checkout(List<ReservaDto>
150     // POST: Reservas/Checkout
151     [HttpPost]
152     [Route("Reservas/Checkout")]
153     public async Task<IActionResult> Checkout([FromBody] List<MVC.Models.ReservaDto> items)
154     {
155         var userId = GetCurrentUser();
156
157         if (items == null || items.Count == 0)
158             return BadRequest(new { message = "Nenhuma reserva enviada." });
159
160         if (items.Count > 3)
161             return BadRequest(new { message = "Máximo de 3 reservas por checkout." });
162
163         // Get all reservations from API
164         var allReservas = await apiService.GetAsync<List<Reserva>>("api/reservas");
165
166         // Create a list of created reservations
167         var created = new List<int>();
168
169         foreach (var it in items)
170         {
171             // Get the book from API
172             var livro = await apiService.GetAsync<Livro>($"api/livros/{it.LivroId}");
173
174             // Check if the book is valid
175             if (livro == null || livro.Estado != "ativo")
176             {
177                 return BadRequest(new { message = $"Livro {it.LivroId} inválido ou não disponível." });
178             }
179
180             // Get the start and end dates
181             var start = it.DataInicio;
182             var end = it.DataFim;
183
184             // Check if the end date is before the start date
185             if (end < start)
186                 return BadRequest(new { message = "Data de fim anterior à data de início." });

```

Figura 60 - Método Checkout do carrinho de compras (1)

```

188     // Get the number of days
189     var days = end.DayNumber - start.DayNumber + 1;
190     if (days > 14)
191         return BadRequest(new { message = "Duração máxima de 14 dias." });
192
193     if (allReservas != null)
194     {
195         // Check if the book is already reserved for the selected period
196         var overlapping = allReservas
197             .Where(r => r.IdLivros.Any(l => l.IdLivro == it.LivroId))
198             .Any(r => !(end < r.DataInicio || start > (r.DataFim.HasValue ? r.DataFim.Value : r.DataInicio)));
199
200         // Check if the book is already reserved for the selected period
201         if (overlapping) {
202             return BadRequest(new { message = $"Livro '{livro.Titulo}' já reservado nesse período." });
203         }
204     }
205
206     // Create the reservation input
207     var reservaInput = new
208     {
209         IdUtilizador = userId.Value,
210         DataInicio = start,
211         DataFim = end,
212         Confirmada = false,
213         LivroIds = new List<int> { livro.IdLivro }
214     };
215
216     // Create the reservation in the database
217     var createdReserva = await apiService.PostAsync<Reserva>("api/reservas", reservaInput);
218     if (createdReserva != null)
219     {
220         // Add the created reservation to the list
221         created.Add(createdReserva.IdReserva);
222     }
223 }
224
225
226 // Return the created reservations
return Ok(created);

```

Figura 61 - Método Checkout do carrinho de compras (2)

## Criação Dinâmica de Autores/Géneros/Editoras

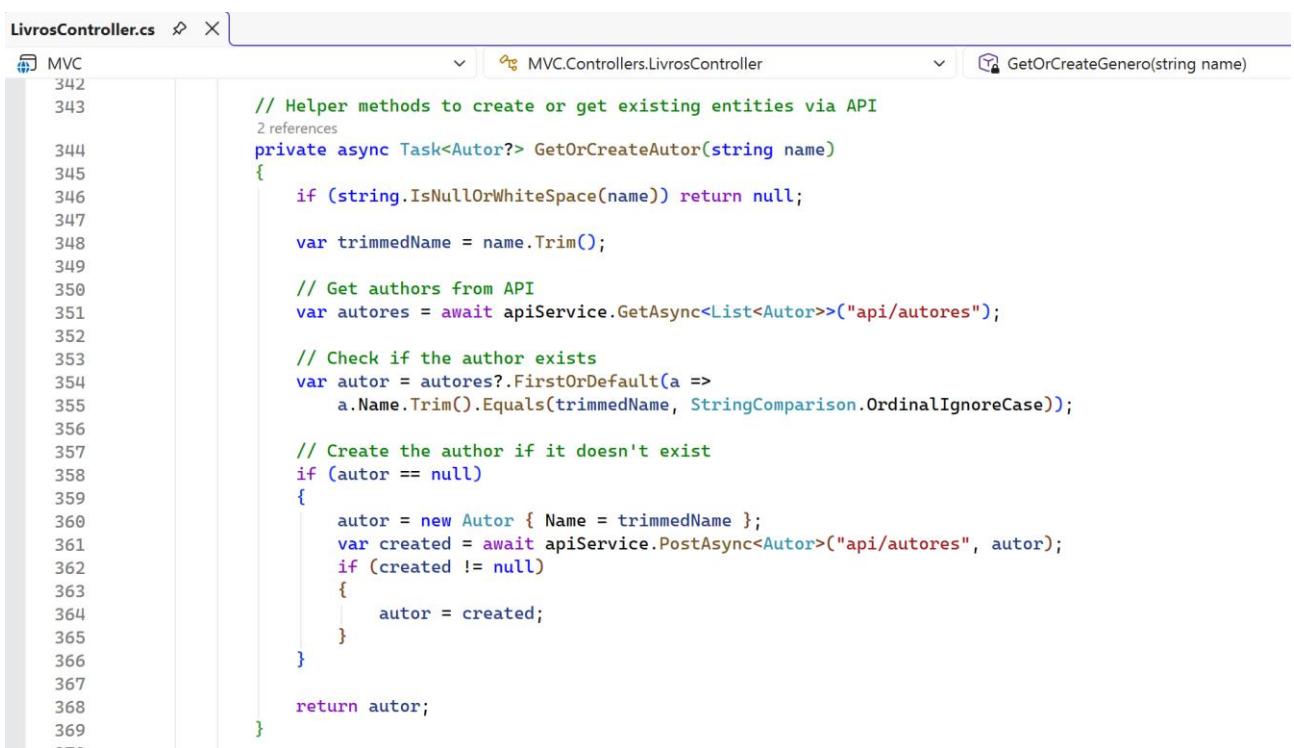
Durante o processo de adição de um novo livro, a aplicação permite a criação automática de autores, géneros e editoras caso estes ainda não existam no sistema. Esta funcionalidade evita que o bibliotecário tenha de criar previamente estas entidades em passos separados, tornando o processo mais simples e eficiente. Para esse efeito, foram implementados métodos auxiliares que verificam a existência da entidade pretendida através da API e procedem à sua criação apenas quando necessário. A comparação é feita de forma insensível a maiúsculas e minúsculas, prevenindo a criação de registo duplicados com grafias diferentes.

O funcionamento deste automatismo segue os seguintes passos:

- Validação do nome introduzido, garantindo que não se encontra vazio
- Obtenção da lista de entidades existentes através da API
- Procura por correspondência do nome, ignorando diferenças de capitalização
- Criação automática da entidade caso não exista

- Retorno da entidade existente ou recém-criada para associação ao livro

Esta lógica é reutilizada para diferentes tipos de entidades, nomeadamente autores, géneros e editoras, assegurando consistência e reutilização de código. A abordagem adotada melhora significativamente a experiência de utilização e contribui para a integridade dos dados armazenados no sistema.



```

    // Helper methods to create or get existing entities via API
    2 references
    private async Task<Autor> GetOrCreateAutor(string name)
    {
        if (string.IsNullOrWhiteSpace(name)) return null;

        var trimmedName = name.Trim();

        // Get authors from API
        var autores = await apiService.GetAsync<List<Autor>>("api/autores");

        // Check if the author exists
        var autor = autores.FirstOrDefault(a =>
            a.Name.Trim().Equals(trimmedName, StringComparison.OrdinalIgnoreCase));

        // Create the author if it doesn't exist
        if (autor == null)
        {
            autor = new Autor { Name = trimmedName };
            var created = await apiService.PostAsync<Autor>("api/autores", autor);
            if (created != null)
            {
                autor = created;
            }
        }

        return autor;
    }

```

Figura 62 - Método GetOrCreateAutor

## Validação de Sobreposição de Reservas

Antes de permitir a criação de uma nova reserva, o sistema valida se o livro pretendido já se encontra reservado para o período selecionado. Esta verificação é essencial para garantir a correta gestão da disponibilidade dos livros e evitar conflitos entre reservas.

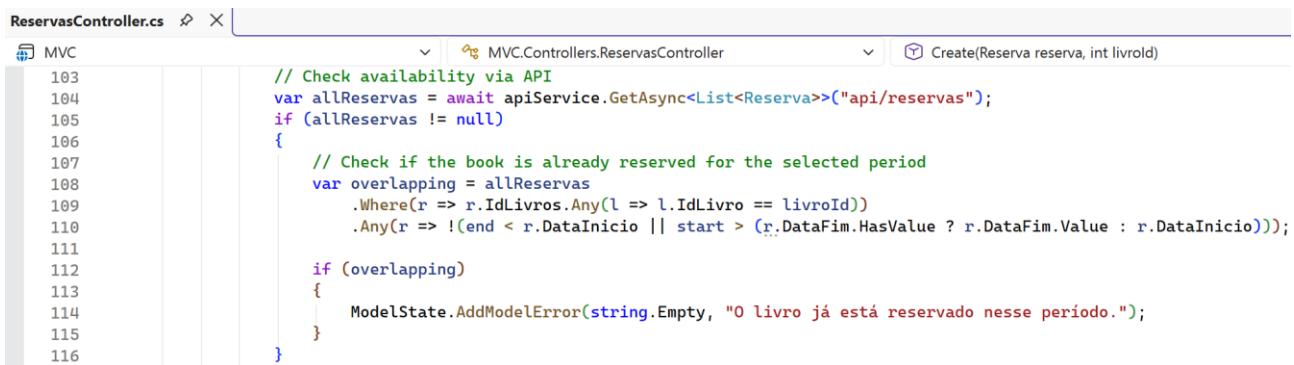
A validação é realizada através da consulta das reservas existentes na API, filtrando apenas aquelas que dizem respeito ao livro em causa. De seguida, é aplicada uma verificação de sobreposição entre o intervalo de datas solicitado pelo utilizador e os intervalos das reservas já registadas.

A lógica de validação segue os seguintes critérios:

- São consideradas apenas as reservas associadas ao livro selecionado

- É verificada a existência de sobreposição entre os intervalos de datas
- Dois intervalos são considerados sobrepostos sempre que o fim de um não ocorre antes do início do outro

Sempre que for detetada uma sobreposição, o sistema impede a criação da reserva e apresenta uma mensagem de erro ao utilizador, garantindo assim a consistência dos dados e a correta utilização do sistema.



```

103 // Check availability via API
104 var allReservas = await apiService.GetAsync<List<Reserva>>("api/reservas");
105 if (allReservas != null)
106 {
107     // Check if the book is already reserved for the selected period
108     var overlapping = allReservas
109         .Where(r => r.IdLivros.Any(l => l.IdLivro == livroId))
110         .Any(r => !(end < r.DataInicio || start > (r.DataFim.HasValue ? r.DataFim.Value : r.DataInicio)));
111
112     if (overlapping)
113     {
114         ModelState.AddModelError(string.Empty, "O livro já está reservado nesse período.");
115     }
116 }

```

Figura 63 - Verificar disponibilidade da reserva

## Dashboard de Estatísticas

A aplicação disponibiliza um dashboard de estatísticas que apresenta informação agregada sobre a utilização do sistema, recorrendo a gráficos e tabelas para facilitar a interpretação dos dados. Esta funcionalidade é direcionada ao bibliotecário e permite analisar padrões de utilização, como os livros mais visualizados, os livros mais requisitados e os períodos do dia com maior atividade.

A obtenção dos dados estatísticos é realizada através de uma chamada à API, que centraliza a lógica de agregação e devolve a informação num único objeto de resposta. No controlador da aplicação MVC, estes dados são mapeados para um *ViewModel* específico, adequado à apresentação na interface gráfica.

Esta separação entre a recolha dos dados e a sua apresentação permite manter a lógica de negócio concentrada na API, garantindo ao mesmo tempo uma interface flexível e facilmente adaptável.

```
HomeController.cs  X | MVC.Controllers.HomeController | Sobre()

73     public async Task<IActionResult> Estatisticas()
74     {
75         // Get statistics from API
76         var statistics = await apiService.GetAsync<StatisticsResponse>("api/home/statistics");
77         if (statistics == null)
78         {
79             return View(new MVC.Models.StatisticsViewModel());
80         }
81
82         var model = new MVC.Models.StatisticsViewModel
83     {
84         TopClicked = statistics.TopClicked.Select(x => new MVC.Models.StatsItem
85         {
86             IdLivro = x.IdLivro,
87             Titulo = x.Titulo,
88             Autor = x.Autor,
89             Value = x.Value
90         }).ToList(),
91         TopRequested = statistics.TopRequested.Select(x => new MVC.Models.StatsItem
92         {
93             IdLivro = x.IdLivro,
94             Titulo = x.Titulo,
95             Autor = x.Autor,
96             Value = x.Value
97         }).ToList(),
98         TimeOfDayCounts = statistics.TimeOfDayCounts
99
100        UserTimes = statistics.UserTimes.Select(kv =>
101            new MVC.Models.UserTimeItem { UserName = kv.Key, Time = kv.Value }).ToList()
102    };
103
104    return View(model);
105 }
```

Figura 64 - Controller da página Estatísticas

## 6 Conclusão e Trabalho Futuro

O desenvolvimento do projeto BookTalk - Biblioteca Escolar Digital permitiu concretizar os objetivos definidos nas fases de análise, desenho e implementação, resultando num sistema funcional, coerente e alinhado com o contexto de uma biblioteca escolar. Ao longo do trabalho foram aplicados conceitos fundamentais de desenvolvimento de aplicações web, nomeadamente a separação de responsabilidades, a utilização de uma arquitetura cliente-servidor e a implementação de boas práticas ao nível da organização do código e da gestão de dados.

Na fase de implementação, foram concretizadas as principais funcionalidades do sistema, incluindo a gestão do catálogo de livros, a associação de autores, géneros e editoras, o processo de reservas com carrinho e *checkout*, a submissão e moderação de reviews e a disponibilização de um painel de estatísticas. A utilização de uma API REST para centralizar a lógica de negócio e de uma aplicação frontend em ASP.NET Core MVC permitiu obter uma solução modular e facilmente extensível. As melhorias introduzidas após o feedback do docente contribuíram para um sistema mais completo, nomeadamente ao nível da modelação dos dados, da experiência de utilização e da relevância da informação estatística apresentada.

Apesar dos resultados alcançados, existem várias possibilidades de evolução futura do sistema. Entre os principais pontos a desenvolver destacam-se a implementação de um mecanismo de autenticação mais robusto, recorrendo a tokens ou identidade federada, a introdução de notificações automáticas para reservas e devoluções, e a melhoria do módulo de estatísticas com análises mais detalhadas e filtros configuráveis. Poderá ainda ser considerada a adaptação da aplicação a dispositivos móveis, bem como a integração com serviços externos para importação automática de catálogos de livros.

Em conclusão, o projeto BookTalk cumpriu os objetivos propostos, permitindo consolidar os conhecimentos adquiridos em ASP.NET Core, nomeadamente na implementação de uma API REST e na sua integração com uma aplicação ASP.NET Core MVC. O trabalho desenvolvido evidenciou a importância da separação entre frontend e backend, da organização do código e da gestão estruturada dos dados. O sistema implementado constitui uma base sólida para futuras melhorias e evoluções, podendo ser facilmente adaptado a contextos reais de utilização em bibliotecas escolares.

# Referências Bibliográficas (normas APA)

Visual Paradigm. (n.d.). Conceptual, Logical and Physical Data Model. Visual Paradigm Online Documentation. Retrieved October 18, 2025, from [https://www.visual-paradigm.com/support/documents/vpuserguide/3563/3564/85378\\_conceptual,l.html](https://www.visual-paradigm.com/support/documents/vpuserguide/3563/3564/85378_conceptual,l.html)

Microsoft. (2025). ASP.NET Core MVC documentation. Microsoft Learn.  
<https://learn.microsoft.com/aspnet/core/mvc>

Microsoft. (2025). *ASP.NET Core Web API documentation*. Microsoft Learn.  
<https://learn.microsoft.com/aspnet/core/web-api>

Microsoft. (2025). *SQL Server LocalDB documentation*. Microsoft Learn.  
<https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-express-localdb?view=sql-server-ver17>

Bootstrap. (2024). *Bootstrap documentation (Version 5)*.  
<https://getbootstrap.com/docs/5.0/getting-started/introduction/>

Bootstrap. (2024). *Bootstrap Icons documentation*.  
<https://icons.getbootstrap.com/>

# Anexo I

A API REST expõe endpoints organizados por recursos principais. A tabela seguinte apresenta a especificação completa de todos os endpoints disponíveis:

Método HTTP	Endpoint	Descrição	Parâmetros de Entrada	Resposta
<b>LivrosController</b>				
GET	/api/livros	Lista todos os livros	Nenhum	200 OK: Array de Livro com Autores, Gêneros, EditoraNavigation
GET	/api/livros/{id}	Obtém detalhes de um livro	id (int) - path parameter	200 OK: Objeto Livro com relações carregadas 404 Not Found: Livro não encontrado
POST	/api/livros	Cria um novo livro	Body: Livro (JSON)	201 Created: Livro criado com ID 400 Bad Request: Dados inválidos
PUT	/api/livros/{id}	Atualiza um livro existente	id (int) - path parameter Body: Livro (JSON)	204 No Content: Atualização bem-sucedida 400 Bad Request: Dados inválidos 404 Not Found: Livro não encontrado
DELETE	/api/livros/{id}	Remove um livro	id (int) - path parameter	204 No Content: Remoção bem-sucedida 404 Not Found: Livro não encontrado

Método HTTP	Endpoint	Descrição	Parâmetros de Entrada	Resposta
<b>ReservasController</b>				
GET	/api/reservas	Lista reservas	userId (int, opcional) - query parameter	200 OK: Array de Reserva com IdUtilizad orNavigation e IdLivros
GET	/api/reservas/{id}	Obtém detalhes de uma reserva	id (int) - path parameter	200 OK: Objeto Reserva com relações
POST	/api/reservas	Cria uma nova reserva	Body: ReservaInp ut (JSON)	201 Created: Reserva criada 400 Bad Request: Dados inválidos
PUT	/api/reservas/{id}	Atualiza uma reserva	id (int) - path parameter Body: Reserva (JS ON)	204 No Content: Atualização bem-sucedida
POST	/api/reservas/{id} /confirm	Confirma uma reserva	id (int) - path parameter	204 No Content: Reserva confirmada
DELETE	/api/reservas/{id}	Remove uma reserva	id (int) - path parameter	204 No Content: Remoção bem-sucedida
<b>ReviewsController</b>				
GET	/api/reviews	Lista reviews	status (string, opcional) - query parameter livroId (int, opcional) - query parameter utilizadorId (int, opcional) - query parameter	200 OK: Array de Review com IdLivroNa vigation e IdUtilizadorNav igation
GET	/api/reviews/{id}	Obtém detalhes de uma review	id (int) - path parameter	200 OK: Objeto Review com relações

Método HTTP	Endpoint	Descrição	Parâmetros de Entrada	Resposta
POST	/api/reviews	Cria uma nova review	Body: ReviewInput (JSON)	201 Created: Review criada 400 Bad Request: Dados inválidos
PUT	/api/reviews/{id}	Atualiza uma review	id (int) - path parameter Body: Review (JSON)	204 No Content: Atualização bem-sucedida
POST	/api/reviews/{id}/approve	Aprova uma review	id (int) - path parameter	204 No Content: Review aprovada
POST	/api/reviews/{id}/reject	Rejeita uma review	id (int) - path parameter	204 No Content: Review rejeitada
DELETE	/api/reviews/{id}	Remove uma review	id (int) - path parameter	204 No Content: Remoção bem-sucedida
<b>AccountController</b>				
POST	/api/account/login	Autentica um utilizador	Body: LoginRequest (JSON) Email (string) Password (string)	200 OK: Objeto Utilizador (sem password) 400 Bad Request: Dados obrigatórios em falta 401 Unauthorized: Credenciais inválidas
<b>UtilizadoresController</b>				
GET	/api/utilizadores	Lista todos os utilizadores	Nenhum	200 OK: Array de Utilizador
GET	/api/utilizadores/{id}	Obtém detalhes de um utilizador	id (int) - path parameter	200 OK: Objeto Utilizador
POST	/api/utilizadores	Cria um novo utilizador	Body: Utilizador (JSON)	201 Created: Utilizador criado

Método HTTP	Endpoint	Descrição	Parâmetros de Entrada	Resposta
PUT	/api/utilizadores/{id}	Atualiza um utilizador	id (int) - path parameter Body: Utilizador (JSON)	204 No Content: Atualização bem-sucedida
DELETE	/api/utilizadores/{id}	Remove um utilizador	id (int) - path parameter	204 No Content: Remoção bem-sucedida
<b>AutoresController</b>				
GET	/api/autores	Lista todos os autores	Nenhum	200 OK: Array de Autor
GET	/api/autores/{id}	Obtém detalhes de um autor	id (int) - path parameter	200 OK: Objeto Autor
POST	/api/autores	Cria um novo autor	Body: Autor (JSON) Name (string)	201 Created: Autor criado
<b>EditorasController</b>				
GET	/api/editoras	Lista todas as editoras	Nenhum	200 OK: Array de Editora
GET	/api/editoras/{id}	Obtém detalhes de uma editora	id (int) - path parameter	200 OK: Objeto Editora
POST	/api/editoras	Cria uma nova editora	Body: Editora (JSON) Name (string)	201 Created: Editora criada
<b>GenerosController</b>				
GET	/api/generos	Lista todos os géneros	Nenhum	200 OK: Array de Genero
GET	/api/generos/{id}	Obtém detalhes de um género	id (int) - path parameter	200 OK: Objeto Genero
POST	/api/generos	Cria um novo género	Body: Genero (JSON) Name (string)	201 Created: Género criado

Método HTTP	Endpoint	Descrição	Parâmetros de Entrada	Resposta
<b>HomeController</b>				
GET	/api/home/statistics	Obtém estatísticas do sistema	Nenhum	200 OK: StatisticsResponse com top clicados, top reservados e contagens por hora do dia
POST	/api/home/accesslog	Regista um acesso ao sistema	Nenhum	204 No Content: Acesso registado

# Anexo II

API 1.0 OAS 3.0  
/swagger/v1/swagger.json

**Account**

`POST /api/Account/login`

**Autores**

`GET /api/Autores`

`POST /api/Autores`

`GET /api/Autores/{id}`

**Editoras**

`GET /api/Editoras`

`POST /api/Editoras`

`GET /api/Editoras/{id}`

**Generos**

`GET /api/Generos`

`POST /api/Generos`

`GET /api/Generos/{id}`

**Home**

`GET /api/Home/statistics`

`POST /api/Home/accesslog`

**Livros**

`GET /api/Livros`

`POST /api/Livros`

`GET /api/Livros/{id}`

`PUT /api/Livros/{id}`

`DELETE /api/Livros/{id}`

Reservas	
GET	/api/Reservas
POST	/api/Reservas
GET	/api/Reservas/{id}
PUT	/api/Reservas/{id}
DELETE	/api/Reservas/{id}
POST	/api/Reservas/{id}/confirm
Reviews	
GET	/api/Reviews
POST	/api/Reviews
GET	/api/Reviews/{id}
PUT	/api/Reviews/{id}
DELETE	/api/Reviews/{id}
POST	/api/Reviews/{id}/approve
POST	/api/Reviews/{id}/reject
Utilizadores	
GET	/api/Utilizadores
POST	/api/Utilizadores
GET	/api/Utilizadores/{id}
PUT	/api/Utilizadores/{id}
DELETE	/api/Utilizadores/{id}