instituto politécnico de gestão e tecnologia
escola superior de tecnologia

LICENCIATURA EM ENGENHARIA INFORMÁTICA

PLATAFORMA WEB AMINHABIBLIOTECA

BRUNO GAMA SILVA - A22108291

PROJETO DE ENGENHARIA INFORMÁTICA

GAIA
SETEMBRO DE 2024

Resumo

Este relatório descreve o desenvolvimento de uma plataforma web de gestão de biblioteca, realizada como projeto final do curso de Engenharia Informática. A plataforma pretende otimizar a interação entre a biblioteca e os seus utilizadores, permitindo pesquisar, requisitar e reservar livros, e fornecer ferramentas de gestão de inventário e atendimento ao cliente para os funcionários.

O projeto incluiu funcionalidades como autenticação segura, pesquisa avançada, gestão de perfis de utilizador, e mecanismos de requisição e reserva de livros. As interfaces foram desenhadas para serem intuitivas e responsivas, garantindo uma experiência de utilização otimizada em diversos dispositivos.

Em conclusão, o sistema cumpre os requisitos estipulados, promovendo uma gestão eficiente da biblioteca e uma experiência de utilizador acessível, assegurando segurança e flexibilidade na gestão de dados.

Índice

Índice de figuras

PARTE I

PROJETO DE ENGENHARIA INFORMÁTICA

1. Introdução

Este relatório descreve o desenvolvimento de uma plataforma web de gestão de biblioteca, realizada como projeto final do curso de Engenharia Informática. O principal objetivo da plataforma é melhorar a interação entre utilizadores e a biblioteca, facilitando a pesquisa, requisição e reserva de livros para utilizadores comuns, e fornecendo ferramentas eficientes para funcionários gerirem o inventário, controlar requisições, responderem a tickets de suporte e controlarem o histórico de livros requisitados. O sistema também oferece funcionalidades específicas para o administrador da biblioteca, garantindo uma gestão centralizada e eficaz.

A plataforma foi desenvolvida com foco na segurança, usabilidade e escalabilidade, oferecendo uma interface intuitiva. Inclui autenticação segura, gestão de perfis, criação de listas de favoritos e pré-reserva de livros. Para os funcionários, há uma interface específica para gestão de inventário, edição de informações e atendimento a utilizadores. O administrador tem acesso a todas as funcionalidades, com permissões para gerir utilizadores e conteúdo.

O relatório apresenta o planeamento e as decisões de design, descreve as funcionalidades implementadas e a interação dos utilizadores com o sistema. Conclui com os resultados, desafios enfrentados, soluções implementadas e sugestões para melhorias futuras.

1.1 Enquadramento

Nos últimos anos, as bibliotecas têm-se modernizado através da digitalização dos seus processos e da adoção de tecnologias que melhoram a gestão de inventários e o atendimento aos utilizadores. Este projeto pretende desenvolver uma plataforma web para otimizar a gestão de bibliotecas, facilitando a pesquisa, requisição e reserva de livros para utilizadores, e fornecendo ferramentas eficientes para os funcionários gerirem o inventário e responderem a tickets de suporte.

Muitas bibliotecas ainda utilizam sistemas obsoletos ou processos manuais, resultando em ineficiências. A plataforma proposta resolve esses problemas ao oferecer uma solução integrada que melhora a interação entre utilizadores e a biblioteca, assegurando a segurança dos dados e a confidencialidade das informações.

A implementação utiliza tecnologias como PHP, MySQL e jQuery, proporcionando uma plataforma ágil e eficiente. Este projeto não só otimiza os processos internos das bibliotecas, como também melhora a experiência dos utilizadores, tornando a gestão de livros mais acessível e prática.

1.2 Objetivo

O principal objetivo deste projeto é desenvolver uma plataforma web eficiente para a gestão de uma biblioteca, oferecendo funcionalidades que facilitem tanto o acesso e a interação dos utilizadores com o inventário de livros, como a gestão interna por parte dos funcionários da biblioteca. A plataforma pretende automatizar processos que, tradicionalmente, são manuais, como a pesquisa, requisição e reserva de livros, bem como a gestão de inventário e o atendimento ao utilizador através de tickets de suporte.

Além disso, o sistema deve garantir uma experiência de utilizador intuitiva, com uma interface clara e acessível, proporcionando funcionalidades específicas para diferentes tipos de utilizadores, como clientes, funcionários e administradores. Cada um destes perfis terá permissões e acessos distintos, adequados às suas responsabilidades e necessidades.

1.3 Organização do Relatório

Este relatório é estruturado em seções que cobrem todas as etapas do desenvolvimento da plataforma web AMinhaBiblioteca, desde o planejamento inicial até a fase de implementação.

O conteúdo deste documento está organizado da seguinte forma:

Parte I: Projeto de Engenharia Informática

1. Introdução: Contexto do Projeto
2. Análise: Plano do projeto
3. Desenho: Paleta de cores, manual de normas e layout
4. Conclusão: Conclusão do plano

Parte II: Projeto Aplicado de Engenharia Informática

1. Desenvolvimento do projeto: Todo o processo de desenvolvimento
2. Base de dados: Funcionamento dinâmico da Base de Dados
3. Instalação Local do Projeto: Manual de instalação local
4. Planeamento Futuro e Melhorias: Planos e ideias para melhorar a plataforma.
5. Conclusões: Conclusão do projeto e resultado

2. Análise

2.1. Levantamentos de requisitos

No levantamento de requisitos deste projeto, identificou-se que a plataforma possui três tipos distintos de utilizadores: clientes, funcionários e administradores. Cada um desses perfis de utilizador tem necessidades e permissões específicas, o que resultou na definição de requisitos funcionais e não funcionais para cada tipo.

Além desses requisitos específicos, foram também identificados requisitos comuns que se aplicam aos três tipos de utilizadores. Estes incluem, por exemplo, a necessidade de autenticação segura, a escolha entre idiomas e a disponibilidade contínua da plataforma. Estes requisitos garantem que todos os utilizadores possam interagir com o sistema de forma coerente e segura, independentemente das suas funções.

Requisitos para utilizadores clientes

Funcionais:

1. O sistema deve fornecer mecanismos seguros para que o utilizador possa iniciar sessão e aceder à plataforma.
2. O utilizador tem a opção de realizar pesquisas seletivas por:
    - Título do livro: basta inserir o título completo ou parcial do livro desejado.
    - Género do livro: escolha entre os diferentes géneros literários disponíveis na plataforma para encontrar livros do seu interesse.
    - Autor: pesquise por livros de um autor específico, digitando o seu nome completo ou parcial.
    - Disponibilidade: verificar se o livro desejado está disponível para requisição no momento da pesquisa.
    - Linguagem: escolher a linguagem do livro mediante aquelas que estão disponíveis.
    - Editora: escolha entre uma lista de editoras disponíveis.
3. O utilizador tem a possibilidade de requisitar até 5 livros em diferentes momentos, definindo datas de entrega distintas para cada requisição.

4. O utilizador pode alterar as suas informações pessoais e dados de acesso através de uma página de perfil do utilizador.
5. Em caso de falha na entrega de um livro, o utilizador estará sujeito ao pagamento de multa e à suspensão do seu direito de realizar novas requisições.
6. Os utilizadores podem colocar-se em fila para um livro que já tenha sido requisitado, fazendo uma pré-reserva.
7. O sistema permite que o utilizador estenda a data de entrega de um livro até duas semanas, desde que o livro não tenha sido reservado por outro utilizador.
8. O número de pré-reservas que cada utilizador pode fazer é de duas.
9. O sistema permite que o utilizador crie e gere uma lista de favoritos, onde pode adicionar livros que lhe interessam.
10. O utilizador tem o direito de solicitar ao administrador a exclusão da sua conta e de todas as suas informações.
11. O sistema oferece um canal de suporte através de tickets, onde o utilizador pode submeter um pedido de ajuda em caso de dúvidas ou problemas na plataforma. Um funcionário ou administrador irá analisar o ticket e responder o mais breve possível.

Não funcionais:

1. Validação de dados de autenticação.
2. O sistema deve fornecer ao utilizador uma página dedicada à gestão de perfil, onde pode visualizar e editar as suas informações pessoais e dados de acesso.
3. A página de pesquisa deve ser capaz de apresentar os livros ou autores que correspondem aos filtros selecionados pelo utilizador.
4. O sistema deve ser capaz de calcular automaticamente o valor da multa para qualquer livro caso o utilizador não o devolva dentro do prazo estabelecido.
5. O sistema deve ter a capacidade de calcular automaticamente o valor da multa por atraso na devolução de livros e suspender a conta do utilizador de imediato, até que a multa seja paga.
6. O sistema permite que o utilizador visualize as seguintes listas de livros:
    - Livros favoritos: lista dos livros que o utilizador marcou como favoritos.
    - Requisições: lista dos livros que o utilizador requisitou, incluindo requisições ativas.
    - Pré-reservas: lista dos livros que o utilizador pré-reservou.

7. O sistema deve disponibilizar uma página dedicada à submissão de tickets, permitindo que o utilizador os envie em diferentes categorias.

Requisitos para utilizadores funcionários

Funcionais:

1. Tem a capacidade de visualizar, editar ou solicitar a exclusão de informações ou contas de utilizadores clientes.
2. Tem a capacidade de gerir todos os livros, incluindo:
   - Editar informações de livros.
   - Adicionar novos livros.
   - Remover livros da base de dados da biblioteca.
   - Marcar livros como danificados.
3. Confirmar todas as requisições e entregas de livros.
4. Validar todos os pagamentos das multas na plataforma.
5. Tem a capacidade de responder a tickets criados por clientes sobre dúvidas sobre o funcionamento geral da biblioteca.
6. Um funcionário da biblioteca com as devidas permissões pode gerir todos os perfis de autores, incluindo:
   - Editar informações de autores: atualizar nome, biografia, foto, data de nascimento, etc. Corrigir erros ou informações desatualizadas. Adicionar ou remover links para websites ou redes sociais.
   - Adicionar novos autores: criar manualmente perfis de autores novos. Vincular autores a livros existentes na biblioteca.
   - Eliminar autores: remover autores da base de dados da biblioteca. Excluir autores com informações irrelevantes ou inapropriadas.

Requisitos para o administrador

Funcionais:

1. O administrador pode controlar qualquer conta dos outros utilizadores, sejam clientes ou funcionários.
2. O administrador pode responder a todos os tipos de tickets submetidos pelos clientes.

3. O administrador pode confirmar e eliminar um utilizador ao receber um pedido para o excluir.

Não funcionais:

1. O sistema deve apresentar uma interface específica para administradores, com opções adicionais em comparação à interface dos outros utilizadores.
2. O sistema deve permitir acesso total ao administrador para entrar em qualquer página da plataforma.
3. O sistema deve ter uma página para gerir todos os utilizadores da plataforma. Esta página deve estar dividida para que haja uma separação entre clientes e funcionários.
4. Permitir acesso às páginas de gestão de tickets e permitir ao administrador acesso total para o tipo de tickets que pode responder.
5. Mostrar uma secção que apresenta uma lista de utilizadores que desejam as suas contas excluídas, bem como uma forma de excluir as suas informações pessoais da plataforma.

Requisitos para todos os utilizadores

Funcionais:

1. A plataforma só pode ser acedida por um utilizador com sessão iniciada.
2. Qualquer utilizador poderá escolher se quer ver a plataforma web em português ou inglês.

Não funcionais:

1. O sistema deve redirecionar qualquer tentativa de entrada na plataforma sem sessão iniciada.
2. O sistema deve saber diferenciar o tipo de utilizador e redirecioná-lo para a sua página inicial exclusiva.
3. A plataforma deve estar disponível 24 horas por dia, 7 dias por semana, para atender às necessidades de qualquer tipo de utilizadores.
4. A interface do utilizador deve ser intuitiva e fácil de usar, para que qualquer pessoa a possa utilizar. Deve ser acessível em diferentes dispositivos e navegadores.
5. O sistema deve garantir a segurança dos dados do utilizador, fazendo a encriptação da password.

6. A plataforma web deve estar apresentada em português ou inglês, sendo a língua portuguesa a linguagem padrão.
7. A interface da plataforma web deve ser pautada por cores leves, seguindo o manual de normas.

2.2.Diagrama de Use Cases

O Diagrama de Use Cases ilustra as interações entre os diferentes tipos de utilizadores e as funcionalidades da plataforma, proporcionando uma visão clara de como os utilizadores interagem com o sistema.

Neste projeto, o diagrama de casos de uso foi desenvolvido para representar as ações possíveis para cada tipo de utilizador, incluindo clientes, funcionários e administradores. As principais funcionalidades destacadas no diagrama incluem:

Figura 1 - Diagrama de Use Cases

- **Clientes**: Podem criar contas, iniciar sessão, pesquisar livros por título, género, autor e disponibilidade, requisitar e devolver livros, gerir o perfil, pagar multas, responder a questionários de feedback, fazer pré-reservas, estender prazos de entrega, criar listas de favoritos, solicitar a exclusão de contas e enviar tickets de suporte.

- **Funcionários**: Têm a capacidade de gerir contas de clientes, editar, adicionar e remover livros, confirmar requisições e entregas, validar pagamentos de multas, responder a tickets e gerir perfis de autores.

- **Administradores**: Podem controlar qualquer conta de utilizador, responder a todos os tipos de tickets, confirmar e eliminar contas de utilizadores.

Este diagrama serve como base para o desenvolvimento do modelo relacional de dados, garantindo que todas as funcionalidades necessárias sejam suportadas de forma eficiente e segura.

2.3. Estrutura Física do Hardware

A arquitetura do sistema selecionada para o projeto segue o modelo de três camadas, assegurando uma separação clara entre a interface do utilizador (View), a lógica de negócios (Controller) e a camada de dados (Model).

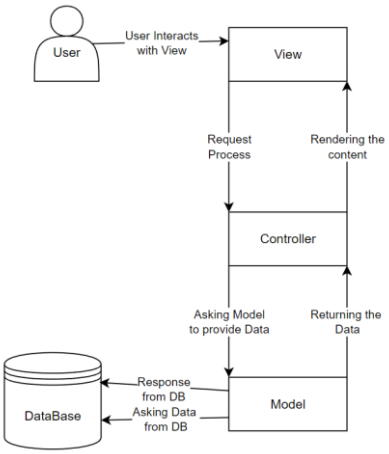


Figura 2 - Estrutura Física do Hardware

2.4. Traduções na Base de Dados

Na base de dados, existiam duas opções para lidar com a tradução dos campos:

1. **Adicionar Campos de Tradução em Cada Tabela**:
    - Esta abordagem envolve adicionar campos específicos para cada idioma em todas as tabelas que necessitam de tradução. Por exemplo, nome_campo_pt e nome_campo_eng.
2. **Criar uma Tabela de Traduções**:
    - Esta abordagem consiste em adicionar uma tabela separada para armazenar as traduções de todos os campos em diferentes idiomas. Esta tabela incluiria o tipo do campo e as traduções correspondentes.

Decisão Detalhada sobre a Tabela de Tradução

Após cuidadosa análise das necessidades de tradução e da estrutura da base de dados, optei por implementar a segunda abordagem, que consiste em criar uma tabela de

tradução separada para gerir os valores de tradução dos géneros de livros. Esta decisão foi tomada com base nos seguintes fatores:

- **Flexibilidade**: Esta abordagem oferece maior flexibilidade para adicionar novos idiomas ou atualizar as traduções existentes sem modificar a estrutura da tabela principal book_genres. Isto garante que a base de dados permaneça adaptável às mudanças futuras nas necessidades de tradução.

- **Manutenção**: Manter os dados de tradução numa tabela separada facilita a gestão e a consulta, tornando mais simples rastrear alterações, garantir consistência e evitar erros. Isto melhora a organização geral da base de dados e facilita a manutenção a longo prazo.

- **Escalabilidade**: À medida que a quantidade de géneros de livros e a necessidade de tradução se expandem, a tabela de tradução separada pode lidar com o crescimento de dados de forma eficiente. Esta abordagem garante que a base de dados permaneça escalável para atender às demandas futuras.

- **Preparação para o Futuro**: Separar a lógica de tradução da estrutura da tabela principal prepara a base de dados para futuras adaptações e integrações. Isto facilita a implementação de novos recursos de tradução ou a adequação a mudanças nos padrões de tradução sem afetar a estrutura central da tabela book_genres.

Conclusão

Acredito que a criação de uma tabela de tradução separada é a melhor decisão para as necessidades atuais e futuras de tradução. Esta abordagem oferece flexibilidade, facilidade de manutenção, escalabilidade e preparação para o futuro, garantindo que a base de dados permaneça robusta e adaptável à medida que os requisitos de tradução evoluem.

2.5.Modelo Relacional de dados

O modelo relacional de dados foi desenhado para suportar todas as funcionalidades da plataforma, com base nos requisitos funcionais e não funcionais identificados.

O modelo é composto por várias tabelas interconectadas, cada uma representando diferentes entidades e suas relações. As principais tabelas incluem:

- **books**: Armazena informações sobre os livros disponíveis na biblioteca, como título, género, autor, data de lançamento, entre outros.
- **authors**: Contém dados sobre os autores, incluindo nome, biografia e outras informações relevantes.
- **users**: Regista informações dos utilizadores, sejam eles clientes, funcionários ou administradores, incluindo dados de autenticação e perfil.
- **requests**: Guarda as informações das requisições feitas pelo utilizador
- **fines**: Contem informações geradas com base em requisições não entregues no tempo estipulado.
- **reserves**: Permite ao utilizador reservar um livro que não está disponível.
- **favorite_books**: Permite que os utilizadores marquem livros como favoritos para fácil acesso futuro.
- **notifications**: Guarda as notificações enviadas aos utilizadores.

As relações entre estas tabelas são indicadas por linhas que conectam os atributos chave, como chaves primárias e estrangeiras, garantindo a integridade referencial e a consistência dos dados. Este modelo relacional assegura que todas as operações, desde a requisição de livros até a gestão de perfis de utilizadores, sejam realizadas de forma eficiente e segura.

Nota: Este modelo sofreu alterações ao longo do desenvolvimento do projeto para melhor atender às necessidades emergentes e otimizar o desempenho da plataforma.

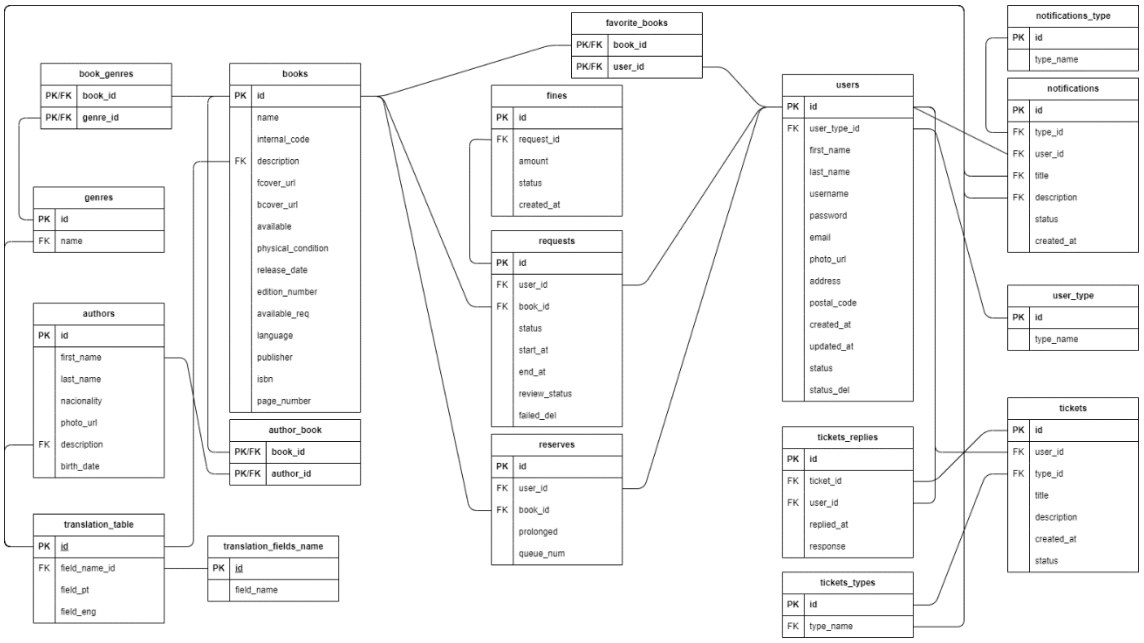


Figura 3 - Modelo Relacional de Dados

2.6. Dicionário de Dados

Tabela	books			
Descrição	Guarda as informações de todos os livros			
Nome da variável	Tipo da variável	Descrição	Tamanho	Restrições
id	int	Id do livro		PK/Identify
title	varchar	Título do livro	100	not null
description	FK	Descrição do livro em português e inglês guardadas na tabela de traduções		not null
internal_code	int	Código interno do livro		not null
fcover_url	varchar	URL da imagem da capa do livro	255	not null
bcover_url	varchar	URL da imagem da contracapa do livro	255	null
available	bool	Disponibilidade do livro	yes/no	not null
physical_condition	int	Um número que corresponde ao estado físico de 0 a 5		not null

				0 a 5
release_date	date	Data de lançamento do livro	YYYY-MM-DD	null
edition_number	int	Número da edição do livro		null
available_req	bool	Disponibilidade do livro para ser requisitado	yes/no	not null
language	varchar	Linguagem do livro	100	not null
publisher	varchar	Editora do livro	100	null
isbn	int	Número Padrão Internacional do Livro		null
page_number	int	Número de páginas do livro		null

Tabela	users			
Descrição	Guarda as informações de todos os utilizadores			
Nome da variável	Tipo da variável	Descrição	Tamanho	Restrições
id	int	ID do utilizador		PK/Identify
user_type_id	FK	Chave estrangeira que especifica o tipo de conta		FK
first_name	varchar	Nome próprio do utilizador	25	not null
last_name	varchar	Apelido do utilizador	25	not null
username	varchar	Nome de utilizador da conta	25	not null/ unique
password	varchar	Palavra-passe da conta	255	not null
email	varchar	E-mail para a conta	100	not null/ unique
photo_url	varchar	URL da imagem para a foto do utilizador	255	not null
address	varchar	Morada do utilizador	255	null
postal_code	varchar	Código postal do utilizador	255	null
created_at	datetime	Data de criação da conta	YYYY-MM-DD hh:mm:ss	not null
updated_at	datetime	Data da última atualização das informações do utilizador	YYYY-MM-DD hh:mm:ss	null
status	bool	Variável que determina se esta conta está ou não bloqueada	yes/no	not null

Tabela	authors			
Descrição	Guarda as informações de todos os autores			
Nome da variável	Tipo da variável	Descrição	Tamanho	Restrições
id	int	Id do autor		PK/Identify
first_name	varchar	Nome próprio do autor	25	not null
last_name	varchar	Apelido do autor	25	
nationality	varchar	Nacionalidade	100	null
photo_url	varchar	URL da imagem para a foto do utilizador	255	null
description	FK	Descrição do autor em português e inglês guardada na tabela de traduções		FK/null
birth_date	date	Data de nascimento do autor	YYYY-MM-DD	null

Tabela	requests			
Descrição	Guarda as informações de todas as requisições feitas pelos utilizadores			
Nome da variável	Tipo da variável	Descrição	Tamanho	Restrições
id	int	Id da requisição		PK/Identify
user_id	FK	Id do utilizador que esta a fazer a requisição		FK
book_id	FK	Id do livro que esta a ser requisitado		FK
status	bool	Determina se a requisição esta ativa ou não	yes/no	
start_at	date	Data do começo da requisição	YYYY-MM-DD	not null
end_at	date	Data do fim da requisição	YYYY-MM-DD	not null
review_status	bool	Determina se a review associada a este livro foi feita ou não	yes/no	not null

Tabela	fines			
Descrição	Guarda as informações de todas as multas dos livros que não foram entregues dentro do tempo limite			
Nome da variável	Tipo da variável	Descrição	Tamanho	Restrições
id	int	Id da requisição		PK/Identify
request_id	FK	Id da requisição associada a multa		FK
amount	double	Valor monetário da multa		not null

status	bool	Determina se o pagamento está pendente ou concluído	yes/no	not null
created_at	date	Data do começo da multa	YYYY-MM-DD	not null

Tabela	reserves			
Descrição	Guarda as informações de todas as requisições feitas pelos utilizadores			
Nome da variável	Tipo da variável	Descrição	Tamanho	Restrições
id	int	Id da requisição		PK/Identify
user_id	FK	Id do utilizador que esta a fazer a reserva		FK
book_id	FK	Id do livro que esta a ser reservado		FK
delayed	bool	Determina se a reserva esta atrasada ou não	yes/no	not null
queue_num	int	Número na fila		not null 1 ou 2

Tabela	notifications			
Descrição	Guarda as informações de todas as notificações dos utilizadores			
Nome da variável	Tipo da variável	Descrição	Tamanho	Restrições
id	int	Id da notificação		PK/Identify
type_id	FK	Id do tipo de notificação		FK
user_id	FK	Id do utilizador que esta a fazer a requisição		FK
title	FK	Título da notificação em português e inglês guardada na tabela de traduções		FK
description	FK	Descrição da notificação em português e inglês guardada na tabela de traduções		FK
status	bool	Determina se a notificação foi vista	yes/no	not null
created_at	datetime	Data da criação da notificação	YYYY-MM-DD hh:mm:ss	not null

2.7. Comportamento do Sistema

Processo de Requisição e Pré-reserva de Livros

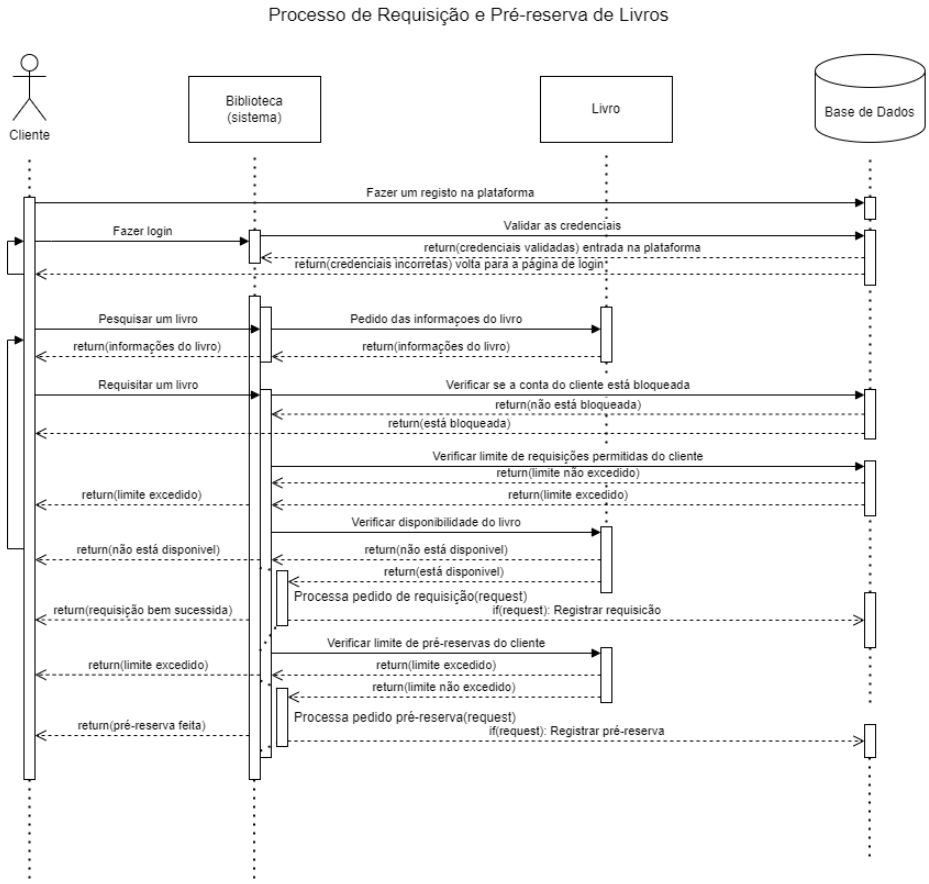


Figura 4 - Processo de Requisição e Pré-reserva de Livros

O comportamento esperado do sistema ao requisitar ou reservar um livro é detalhado no diagrama de fluxo apresentado. Este diagrama ilustra as etapas que um utilizador e o sistema devem seguir para completar essas ações.

1. **Login do Utilizador**:
    - O utilizador deve fazer login na plataforma com as suas credenciais.
    - O sistema valida as credenciais e permite o acesso se forem corretas. Caso contrário, o utilizador é redirecionado para a página de login.
2. **Pesquisa de Livros**:
    - O utilizador pode pesquisar livros por título, género, autor ou disponibilidade.

- O sistema retorna as informações dos livros correspondentes à pesquisa.

3. **Requisição de Livros**:
    - O utilizador seleciona um livro para requisitar.
    - O sistema verifica se a conta do utilizador está bloqueada e se o limite de requisições permitidas não foi excedido.
    - Se o livro estiver disponível e todas as condições forem satisfeitas, o sistema processa a requisição e regista-a na base de dados.
4. **Pré-Reserva de Livros**:
    - Se o livro não estiver disponível para requisição imediata, o utilizador pode optar por fazer uma pré-reserva.
    - O sistema verifica o limite de pré-reservas do utilizador.
    - Se o limite não for excedido, o sistema processa a pré-reserva e regista-a na base de dados.

Este fluxo garante que todas as operações de requisição e pré-reserva sejam realizadas de forma eficiente e segura, atendendo aos requisitos funcionais e não funcionais definidos para a plataforma.

2.8.Lista Completa das Funcionalidades da Plataforma

Funcionalidades:

- Autenticação: O sistema oferece mecanismos seguros de login e acesso à plataforma para todos os tipos de utilizadores.
- Pesquisa: A plataforma oferece pesquisa avançada por título, género, autor e disponibilidade do livro.
- Requisição e Devolução: Os utilizadores podem requisitar até 5 livros, definir datas de entrega e devolver livros.
- Gestão de Perfil: Os utilizadores podem editar suas informações e dados de acesso.

- Multas e Suspensões: Atraso na devolução gera multa e suspensão do direito de requisição.
- Fila de Espera e Pré-Reserva: Permite que os utilizadores se coloquem em fila para livros requisitados e façam pré-reservas.
- Extensão de Prazo: Permite a extensão da data de entrega por até duas semanas, se o livro não estiver reservado.
- Lista de Favoritos: Os utilizadores podem criar e gerenciar uma lista de livros favoritos.
- Exclusão de Conta: Os utilizadores podem solicitar a exclusão da sua conta e dados.
- Suporte por Tickets: Os utilizadores podem enviar tickets de ajuda em caso de dúvidas ou problemas.

Funcionalidades para Funcionários:

- Gestão de Utilizadores: Visualizar, editar e solicitar a exclusão de contas de clientes.
- Gestão de Livros: Editar, adicionar, remover, marcar como danificados ou perdidos.
- Confirmação de Requisições e Entregas: Confirmar todas as requisições e entregas de livros.
- Validação de Pagamentos: Validar pagamentos de multas na plataforma.
- Gestão de Tickets: Responder a tickets criados por clientes.
- Gestão de Autores: Editar, adicionar, eliminar autores e seus perfis.

Funcionalidades para Administrador:

- Controle Total de Contas: Controlar qualquer conta de clientes ou funcionários.
- Resposta a Tickets: Responder a todos os tipos de tickets submetidos pelos clientes.
- Confirmação e Eliminação de Utilizadores: Confirmar e eliminar pedidos de exclusão de contas.

Funcionalidades Gerais:

- Acesso Restrito: A plataforma só pode ser acedida por utilizadores com sessão iniciada.
- Seleção de Idioma: português ou inglês.
- Diferenciação de Tipo de Utilizador: Direcionar para a página inicial específica do tipo de utilizador.
- Disponibilidade 24/7: Plataforma disponível 24 horas por dia, 7 dias por semana.
- Usabilidade: Interface intuitiva e fácil de usar em diferentes dispositivos e navegadores.
- Segurança de Dados: Encriptação de password.
- Idiomas: Plataforma disponível em português e inglês.
- Design: Interface com cores leves, seguindo o manual de normas.

3. Desenho

Nesta secção, iremos detalhar os elementos visuais e estéticos que compõem a plataforma, assegurando uma experiência de utilizador coerente e agradável. A categoria de desenho abrange vários aspetos cruciais, incluindo a paleta de cores, o manual de normas e o layout.

- **Paleta de Cores**: Apresentaremos a seleção de cores utilizada na plataforma, escolhida para proporcionar uma interface visualmente harmoniosa e acessível.
- **Manual de Normas**: Detalharemos as diretrizes e padrões que regem o design da plataforma, assegurando consistência e qualidade em todos os elementos visuais.
- **Layout**: Descreveremos a estrutura e organização das páginas da plataforma, garantindo que a navegação seja intuitiva e eficiente para todos os utilizadores.

Esta abordagem meticulosa ao design visa não só a estética, mas também a funcionalidade e a usabilidade, criando uma experiência de utilizador otimizada e satisfatória.

3.1. Paleta de Cores

A paleta de cores apresentada é composta por cinco cores distintas, cada uma acompanhada pelo seu código hexadecimal. Esta paleta foi utilizada para criar uma interface visualmente harmoniosa e coerente.

Figura 5 - Paleta de Cores

3.2. Manual de Normas

Um manual de normas de um website, também chamado de “guia de estilo da web” ou “manual de identidade online”, é um documento que define diretrizes e padrões para o design, conteúdo e funcionamento de um site. Este manual serve como uma referência essencial para assegurar a consistência, coerência e qualidade em todas as páginas e elementos do website.

1) Logotipo para o Menu:

Figura 6 - Logotipo

2) Logotipo para o Login e Registo

Figura 7 - logotipo em branco

Nota: Fundo preto colocado por ser um logotipo totalmente branco e sem fundo.

3) Cores

Todas as cores foram utilizadas em conformidade com a paleta estipulada, garantindo uma aparência visual coerente e harmoniosa em toda a plataforma.

- Texto (texto): #25476A

- Fundo (background): #FAFAFA
- Primária (primary): #23282D
- Secundária (secundary): #BFBFBF
- Destaque (accent): #AC3E31

4) Tipografia

Tipo de letra escolhidos:

- Poppins
- Roboto

Estilos a cumprir na plataforma

- Título Grande:
  - Poppins
  - 32 pixels
  - Bold
- Título:
  - Poppins
  - 24 pixels
  - Bold
- Subtítulo:
  - Poppins
  - 20 pixels
  - Regular / 400
- Texto Grande:
  - Poppins
  - 32 pixels
  - Regular / 400
- Texto:
  - Roboto
  - 16 pixels
  - Light / 300
- Texto Pequeno:

- Roboto
- 12 pixels
- Regular / 400

Estilos a cumprir na página 404

- Título:
  - Poppins
  - 320 pixels
  - ExtraBold / 800
- Mensagem Grande:
  - Poppins
  - 60 pixels
  - ExtraBold / 800
- Mensagem:
  - Poppins
  - 16 pixels
  - SemiBold / 600

3.3. Layout

O layout da plataforma foi cuidadosamente desenhado no Figma, respeitando todas as normas e diretrizes estabelecidas no manual de normas. Este design visa proporcionar uma experiência de utilizador intuitiva, eficiente e visualmente agradável, garantindo a consistência e a coerência em toda a interface.

Link para visualizar as páginas todas no Figma:

https://www.figma.com/proto/0VHfli6LjalPdIh0lePcq6/Biblioteca?node-id=0-1&t=1eL8FPHd0DYvkzwW-1

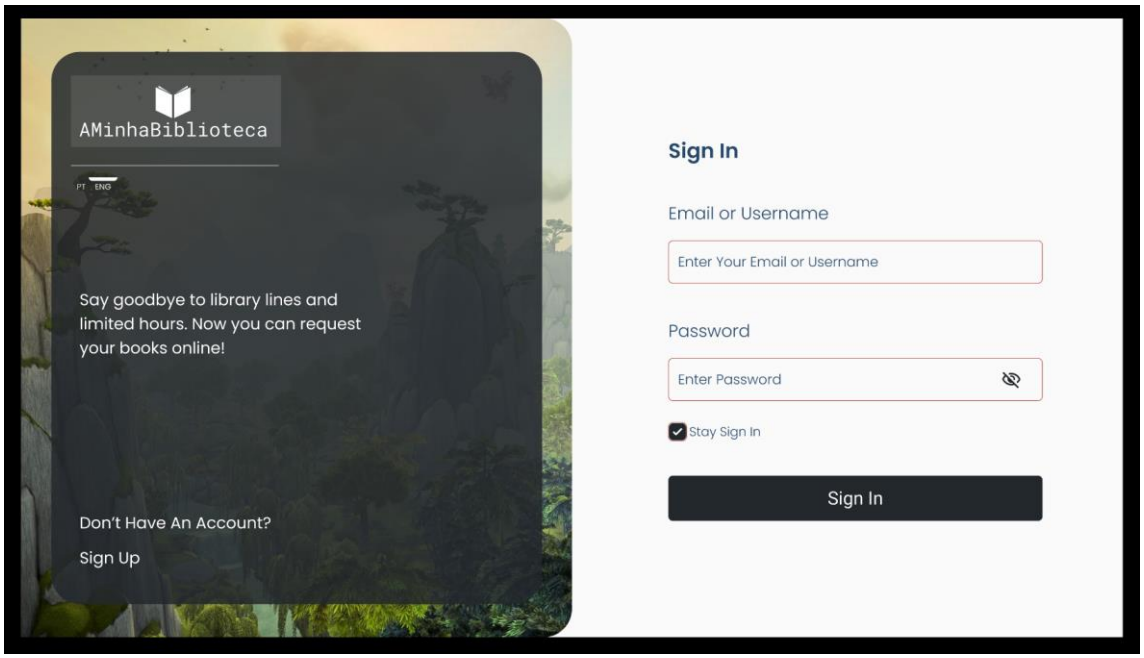


Figura 8 - Página Login

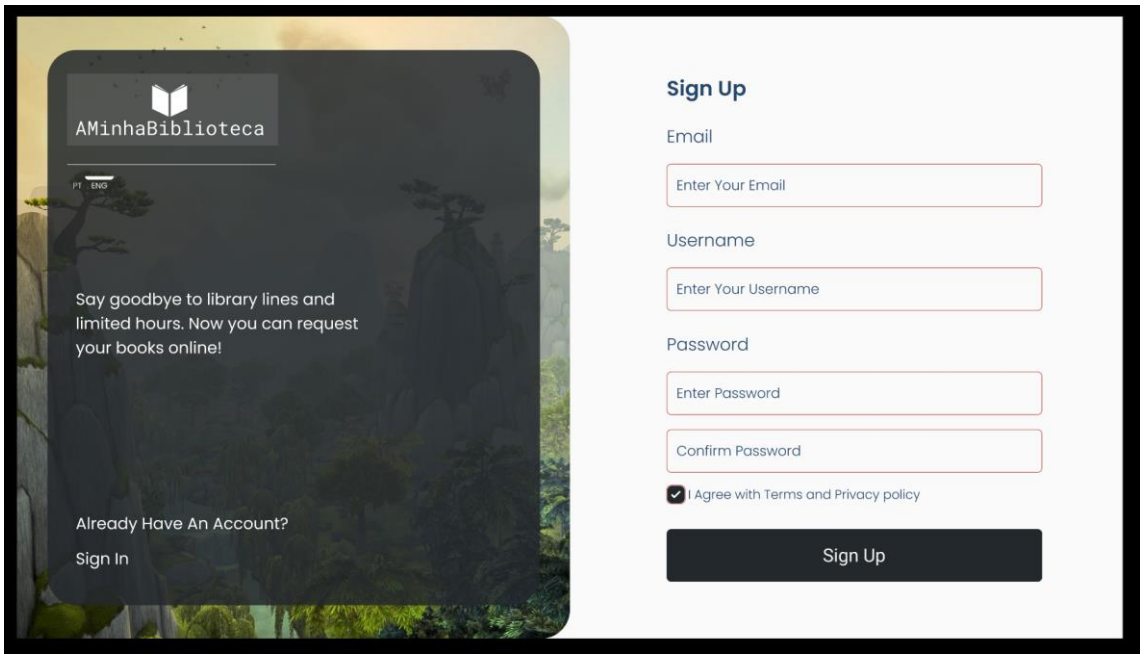


Figura 9 - Página Registo

Figura 10 - Página 404

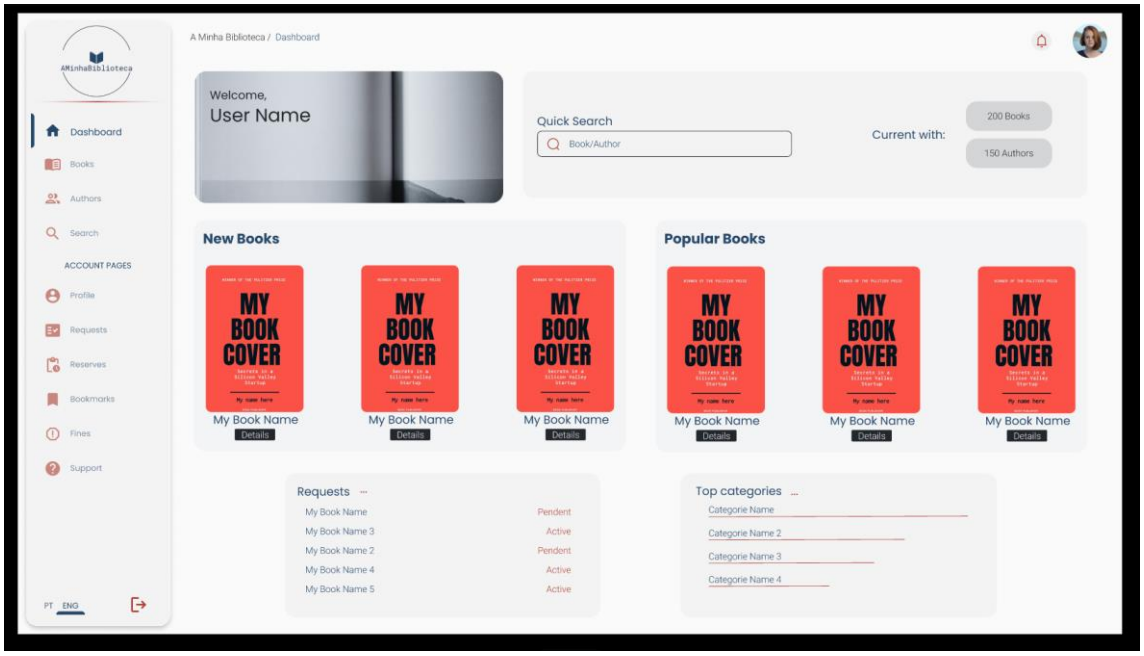


Figura 11 - Página Home

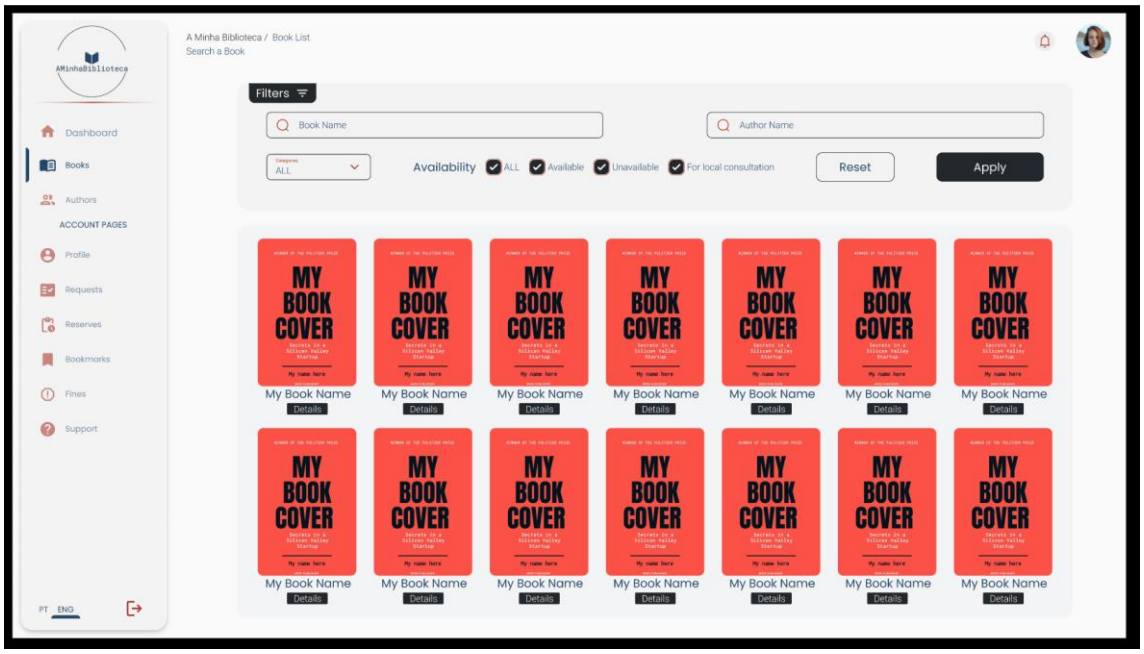


Figura 12 - Página lista de livros

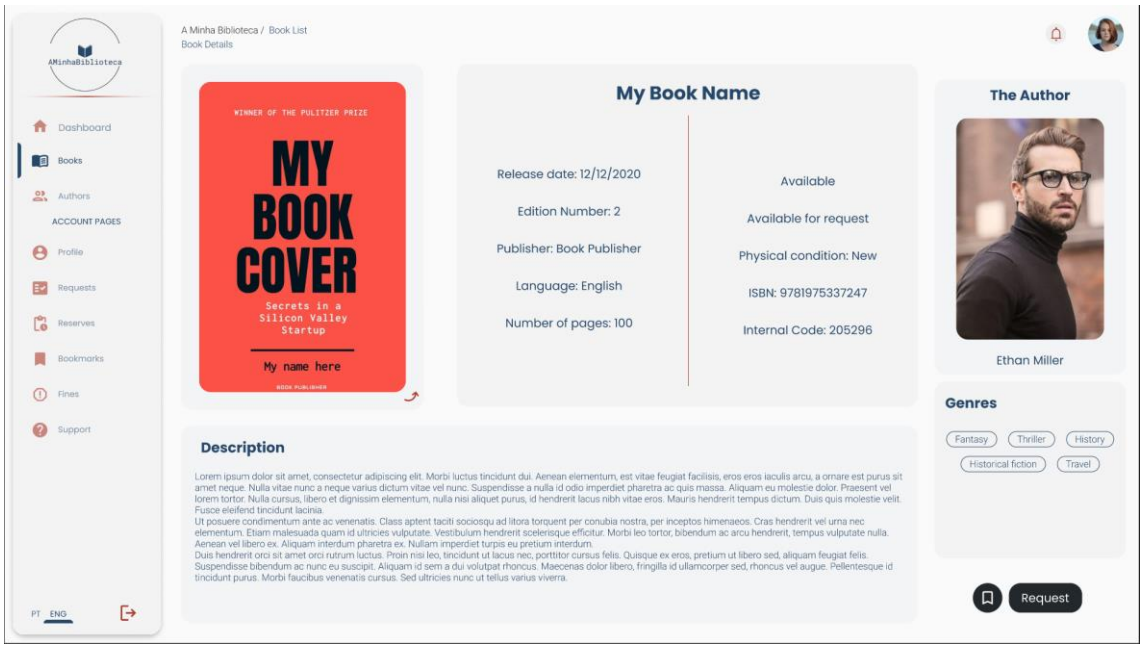


Figura 13 - Página do livro

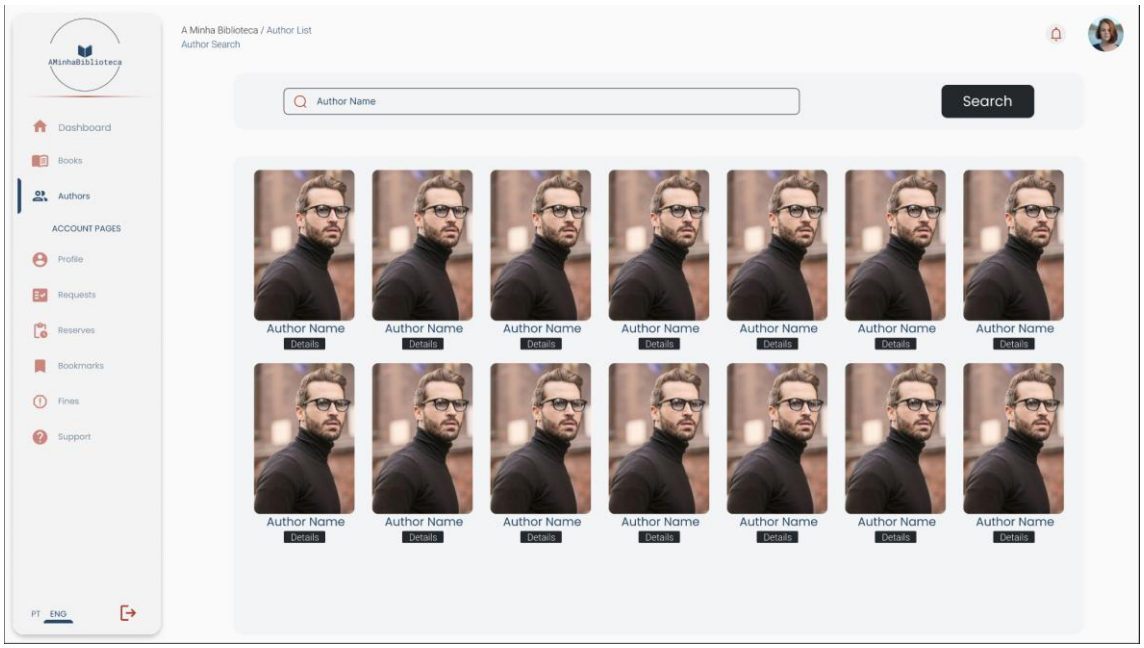


Figura 14 - Página lista de autores

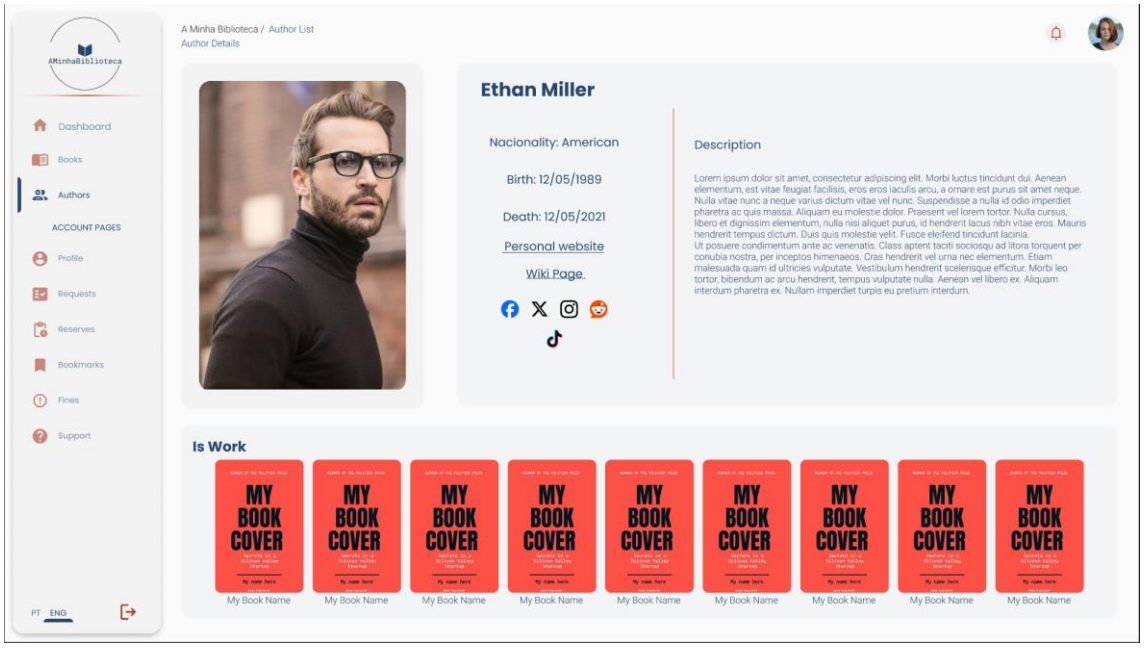


Figura 15 - Página do autor

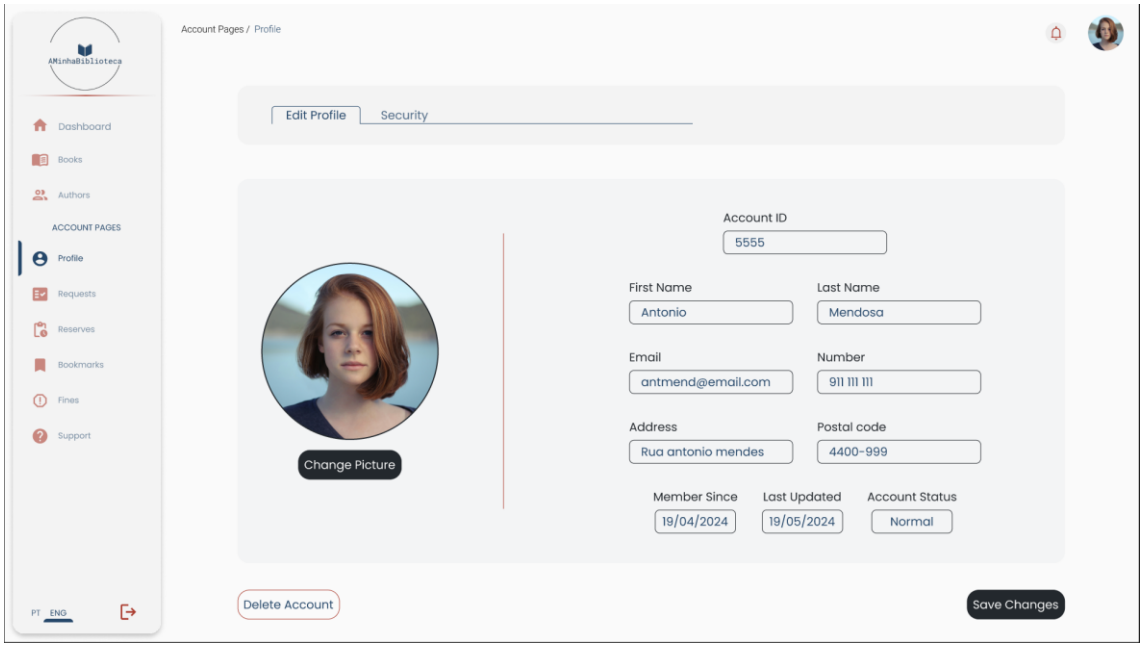


Figura 16 - Página do perfil do utilizador

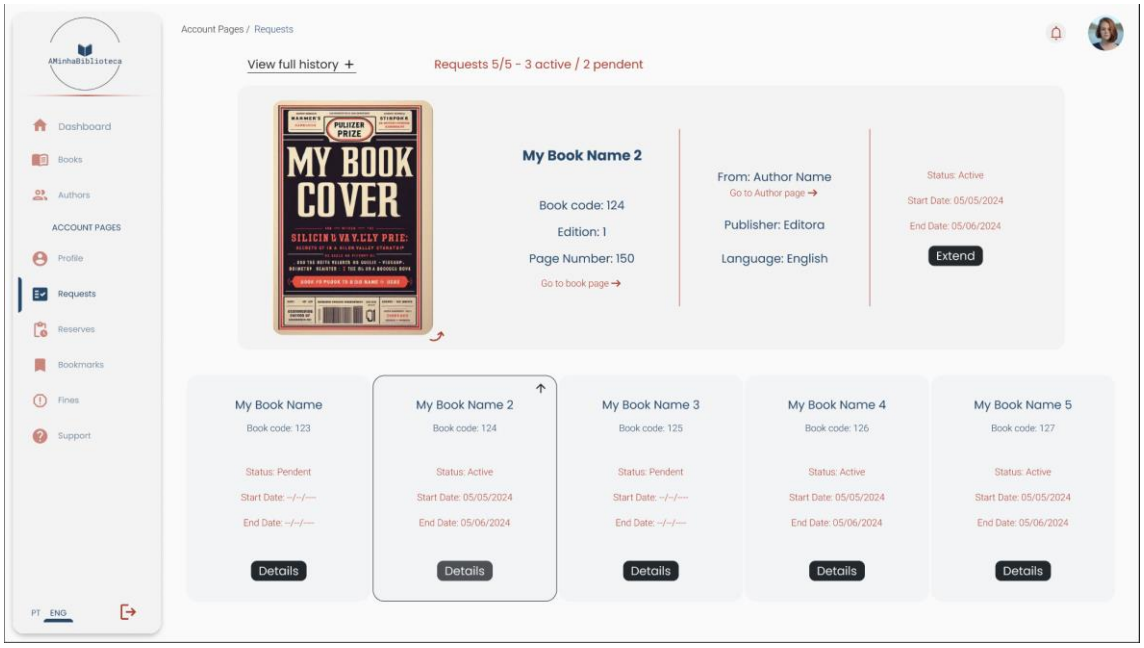


Figura 17 - Página das requisições

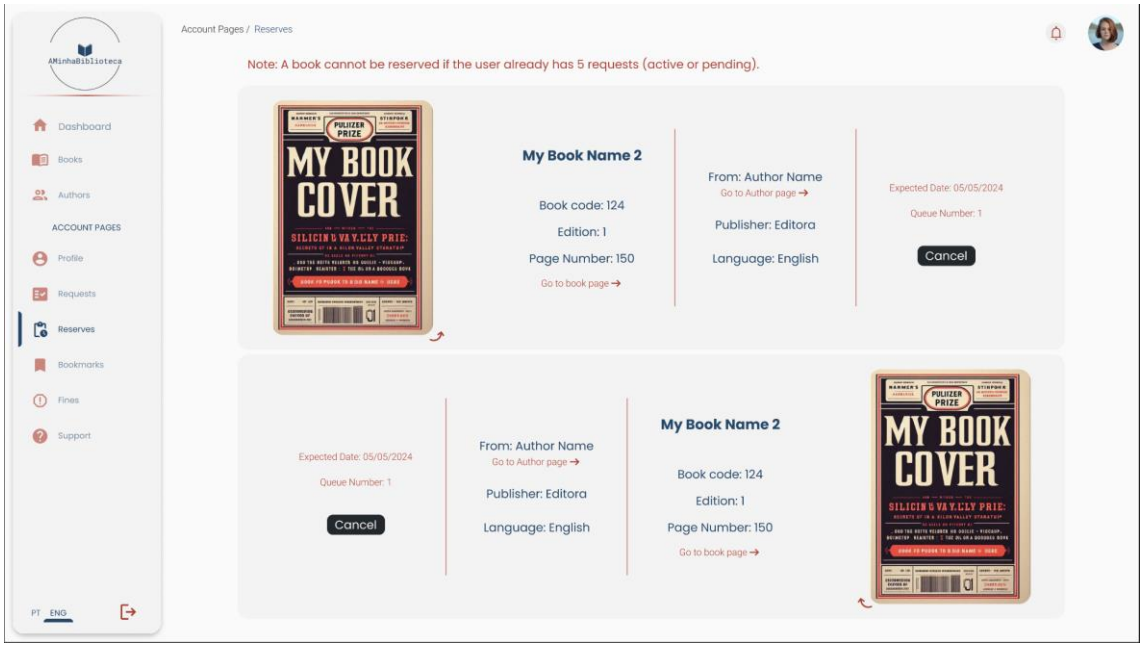


Figura 18 - Página das reservas

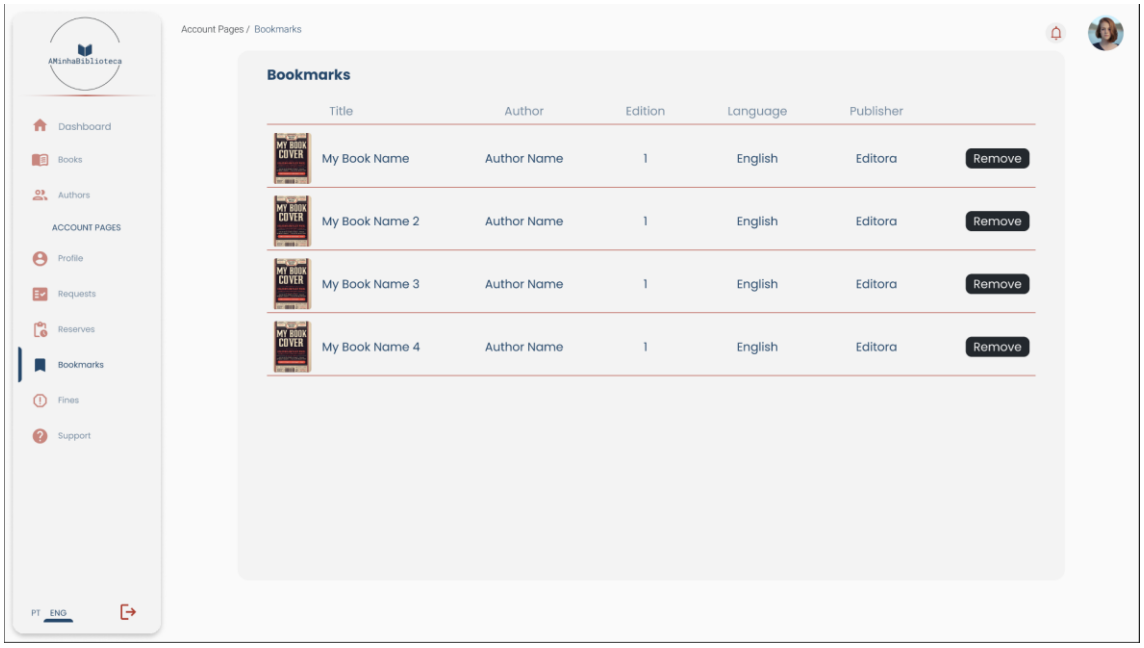


Figura 19 - Página dos favoritos

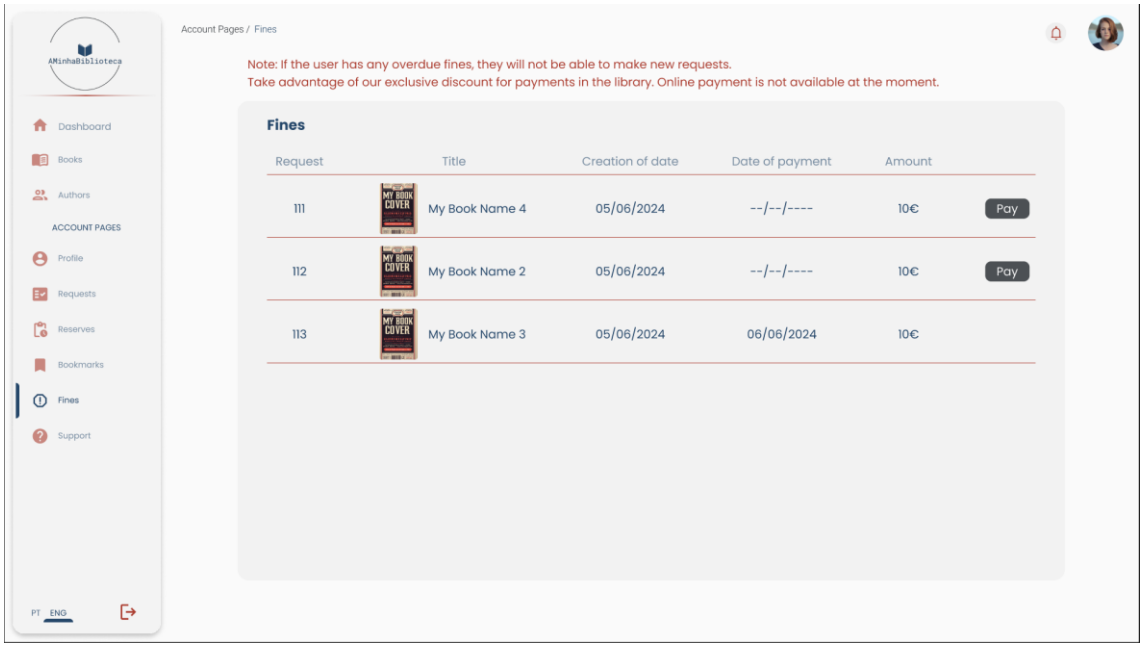


Figura 20 - Página das multas

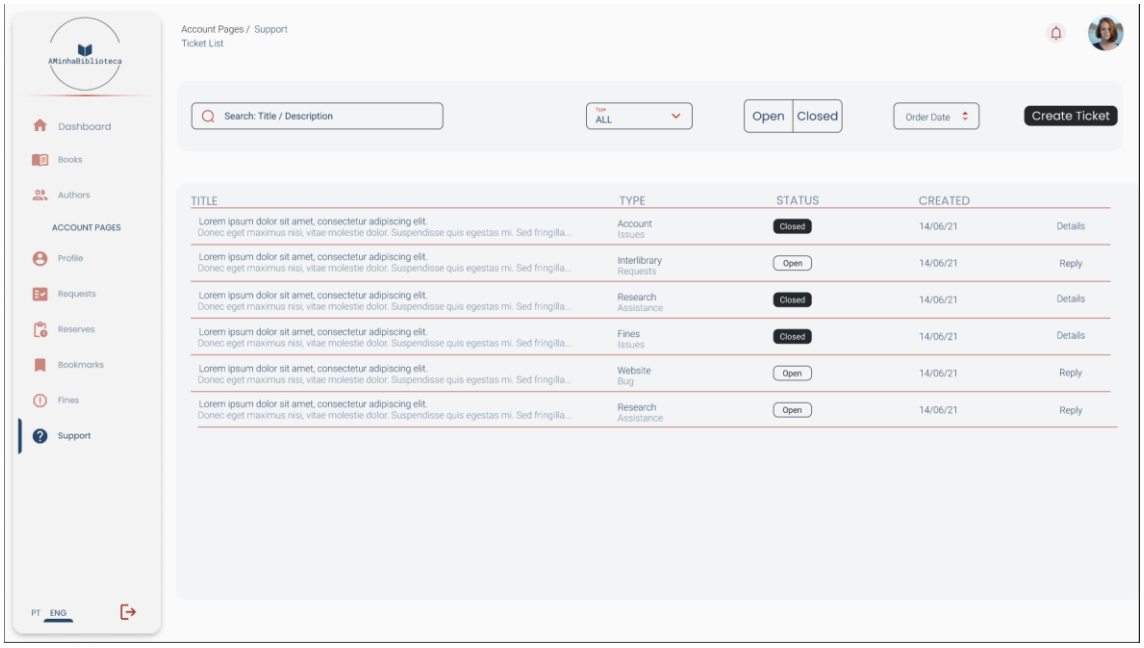


Figura 21 - Página dos tickets

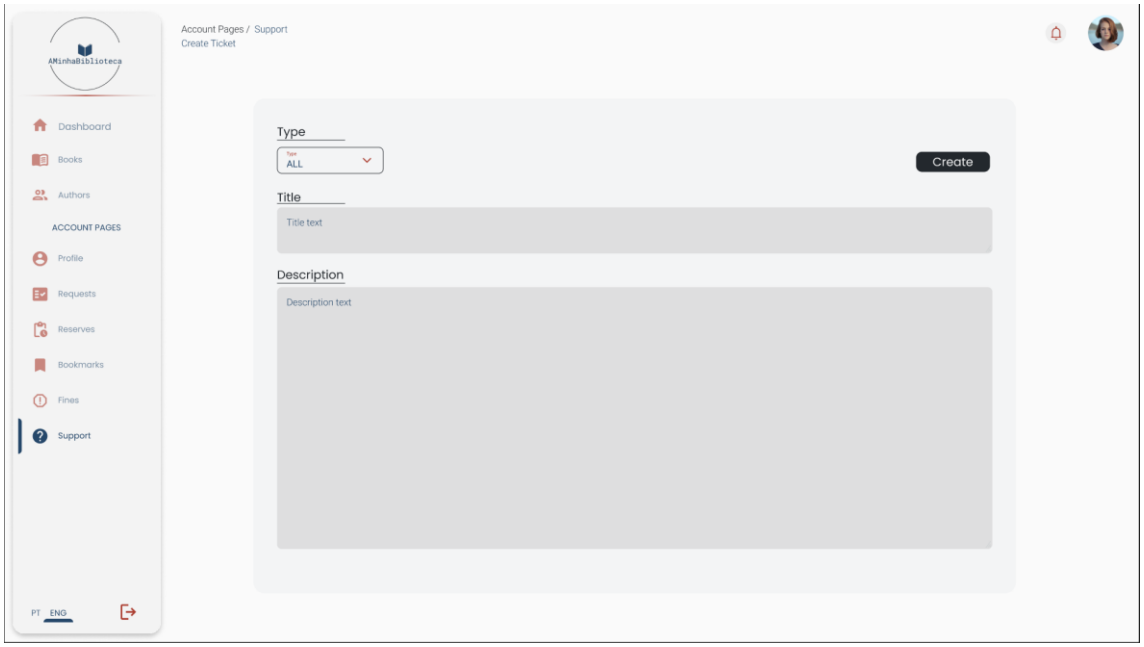


Figura 22 – Página de criar um ticket

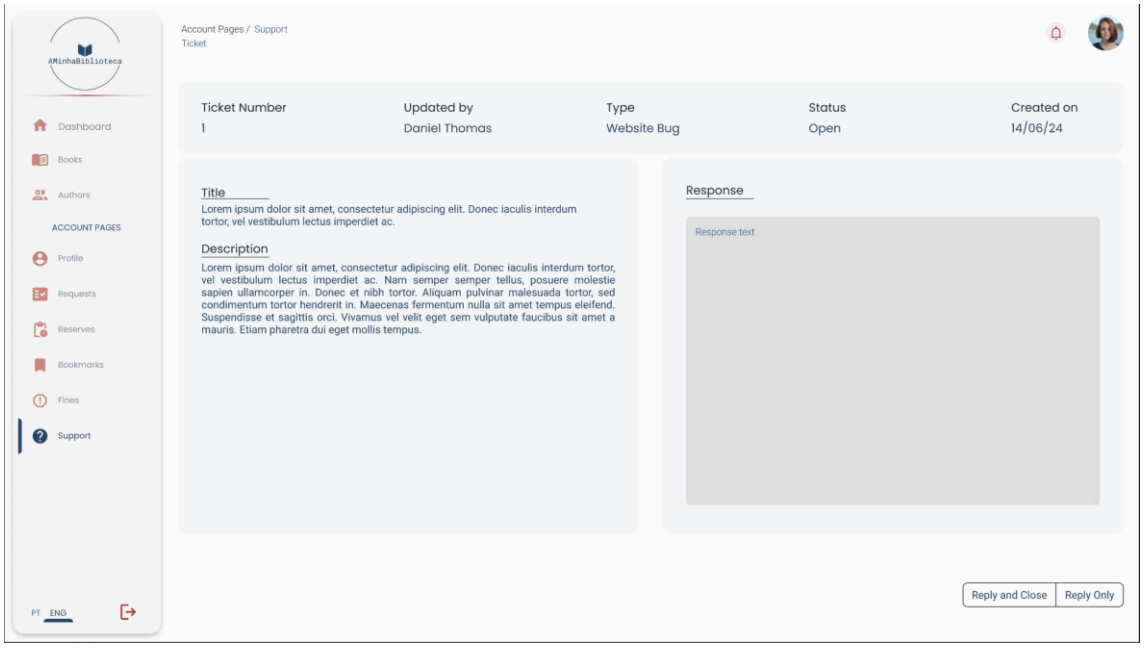


Figura 23 - Página de detalhes do ticket

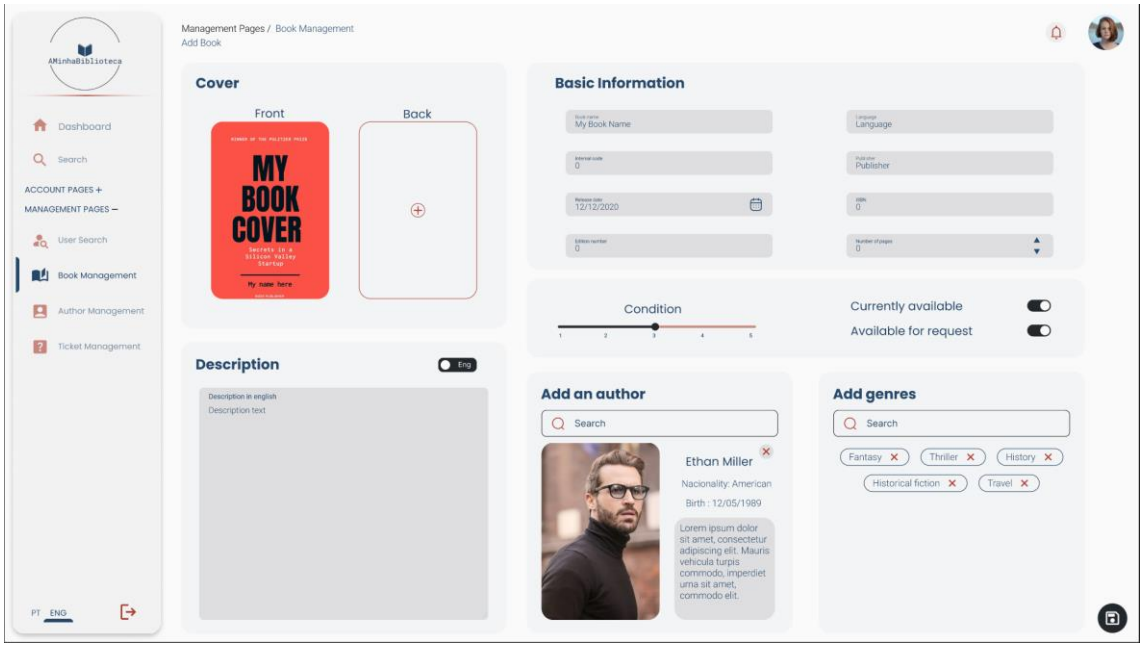


Figura 24 - Página de adicionar um livro

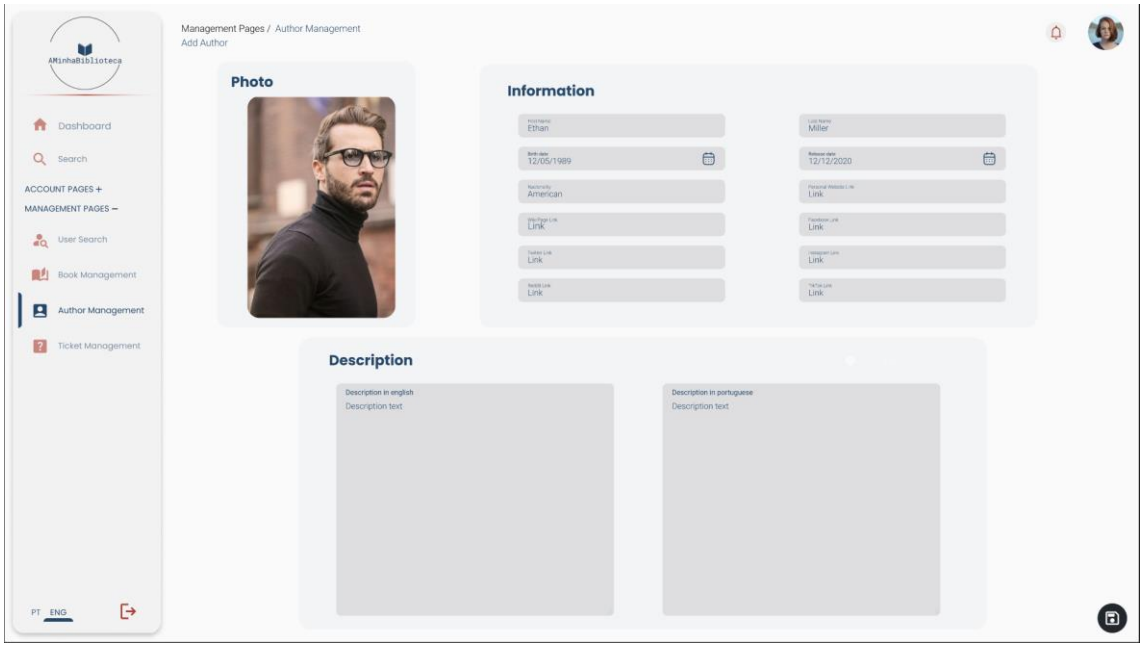


Figura 25 - Página de adicionar um autor

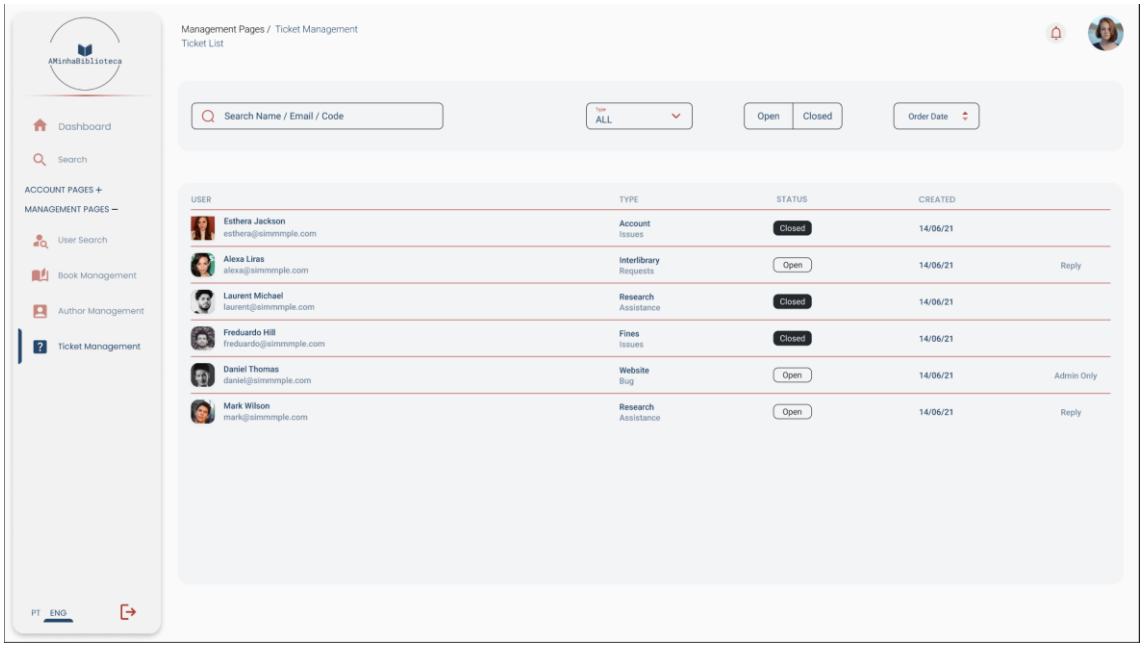


Figura 26 - Página da lista de gerir os tickets

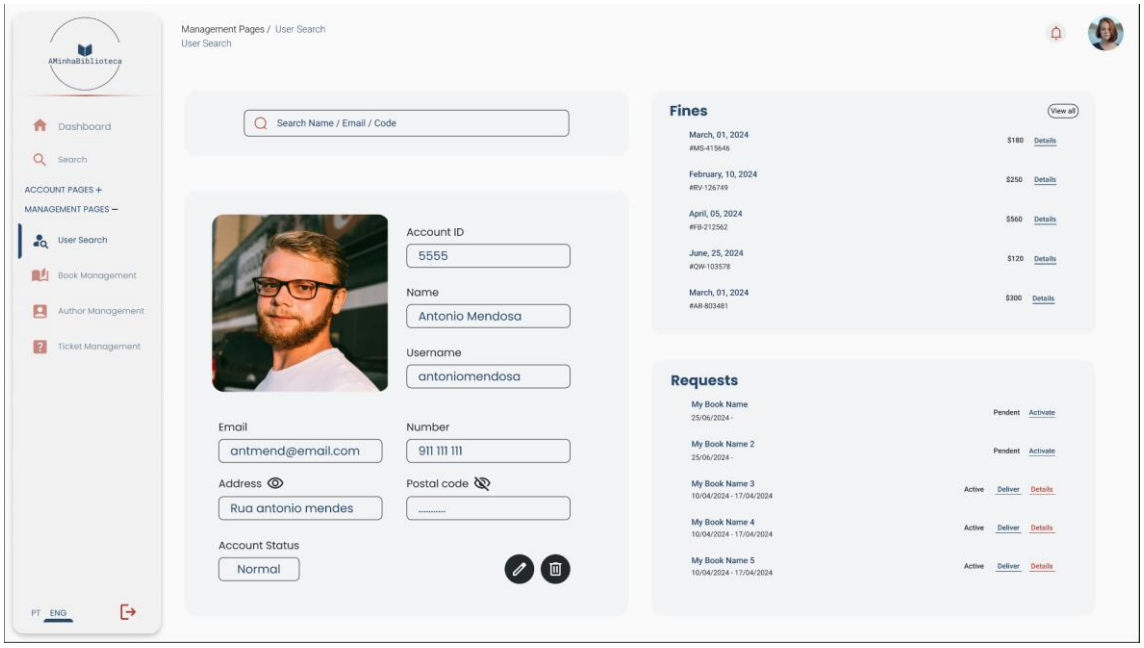


Figura 27 - Página de gerir um utilizador

4. Conclusão

O planeamento deste projeto foi realizado com um enfoque meticuloso em cada detalhe, desde a definição dos requisitos funcionais até a elaboração do layout e da paleta de cores. Cada etapa foi cuidadosamente planejada para garantir que a plataforma atenda às necessidades dos utilizadores de forma eficiente e intuitiva.

Através da análise detalhada dos requisitos, da criação de diagramas de casos de uso e do desenvolvimento de um modelo relacional de dados robusto, estabelecemos uma base sólida para a implementação do sistema. A escolha de uma paleta de cores harmoniosa e a definição de um layout claro e organizado no Figma asseguram uma experiência de utilizador agradável e consistente.

PARTE II

PROJETO APLICADO DE ENGENHARIA INFORMÁTICA

5. Desenvolvimento do Projeto

Nesta seção, será apresentado o processo de desenvolvimento da plataforma, desde a escolha das tecnologias até à implementação final. Serão detalhadas as linguagens de programação utilizadas, as frameworks que suportaram o desenvolvimento e as ferramentas que facilitaram a construção do sistema. Explicarei as decisões técnicas tomadas, as etapas do desenvolvimento e como cada componente foi integrado para criar uma plataforma eficiente, robusta e escalável. Este relato permitirá compreender as bases tecnológicas sobre as quais a plataforma foi construída e o raciocínio por detrás das escolhas feitas ao longo do projeto.

5.1. Escolha das Tecnologias para o Desenvolvimento do Projeto

Neste projeto, optei por uma combinação de tecnologias amplamente utilizadas no desenvolvimento web, visando construir uma solução eficiente, organizada e com alta funcionalidade.

Linguagens de Programação e Tecnologias

HTML e CSS (Sass): Para a estruturação e estilização das páginas, utilizei HTML e CSS. O CSS foi escrito em Sass, uma extensão de CSS que facilita a manutenção e organização do código.

JavaScript (jQuery): A lógica do lado do cliente foi implementada em JavaScript, com o auxílio da biblioteca jQuery. O jQuery simplifica a manipulação do Document Object Model (DOM), a criação de animações e o tratamento de eventos.

PHP: Para o desenvolvimento do backend, escolhi o PHP, uma linguagem de script server-side ideal para a criação de sites dinâmicos e a interação com a base de dados.

MySQL: Para a gestão da base de dados, utilizei MySQL, uma das soluções mais populares e confiáveis para armazenamento e recuperação de dados.

Ambiente de Desenvolvimento

XAMPP: Utilize o XAMPP como servidor local, integrando Apache, MySQL, PHP e Perl, o que permite testar e desenvolver a aplicação localmente.

Visual Studio Code: O código foi escrito no Visual Studio Code (VS Code), um editor de código leve e poderoso, suportado por uma ampla gama de extensões.

Extensões e Ferramentas Adicionais

PHP IntelliSense, PHP Intelephense e PHP Debug: Essas extensões para o Visual Studio Code proporcionam funcionalidades como auto completação de código, análise estática, depuração e refatoração, agilizando o desenvolvimento em PHP.

Live Sass Compiler e Sass (.sass only): Essas extensões simplificam o fluxo de trabalho com Sass, compilando automaticamente os arquivos Sass em CSS e oferecendo recursos como realce de sintaxe e formatação automática.

MySQL e Database Client JDBC: para conectar ao servidor MySQL local diretamente do VS Code, permitindo a manipulação da base de dados sem a necessidade de aceder ao phpMyAdmin.

Prettier - Code formatter by Prettier: para formatar automaticamente o código, mantendo a consistência e legibilidade do projeto.

Frameworks

Bootstrap: Utilizei o Bootstrap para construir layouts responsivos e componentes de interface do utilizador reutilizáveis, acelerando o desenvolvimento e garantindo uma aparência profissional.

Slick.js: A biblioteca Slick.js foi utilizada para criar sliders e carrosséis dinâmicos e responsivos, adicionando interatividade ao site.

A combinação das tecnologias e ferramentas descritas acima permitiu construir uma aplicação web robusta, escalável e de fácil manutenção. A escolha dessas tecnologias foi baseada em sua popularidade, maturidade e capacidade de atender às necessidades do projeto.

5.2. Arquitetura do sistema

A arquitetura em três camadas é uma abordagem amplamente utilizada no desenvolvimento de software, especialmente em aplicações web, por promover a separação de responsabilidades, modularidade e reutilização do código. Esta arquitetura divide a aplicação em três camadas distintas: View (Apresentação), Controller (Controlador) e Model (Modelo). Cada camada tem um papel específico, o que facilita a manutenção, o teste, a escalabilidade e a evolução da aplicação ao longo do tempo.

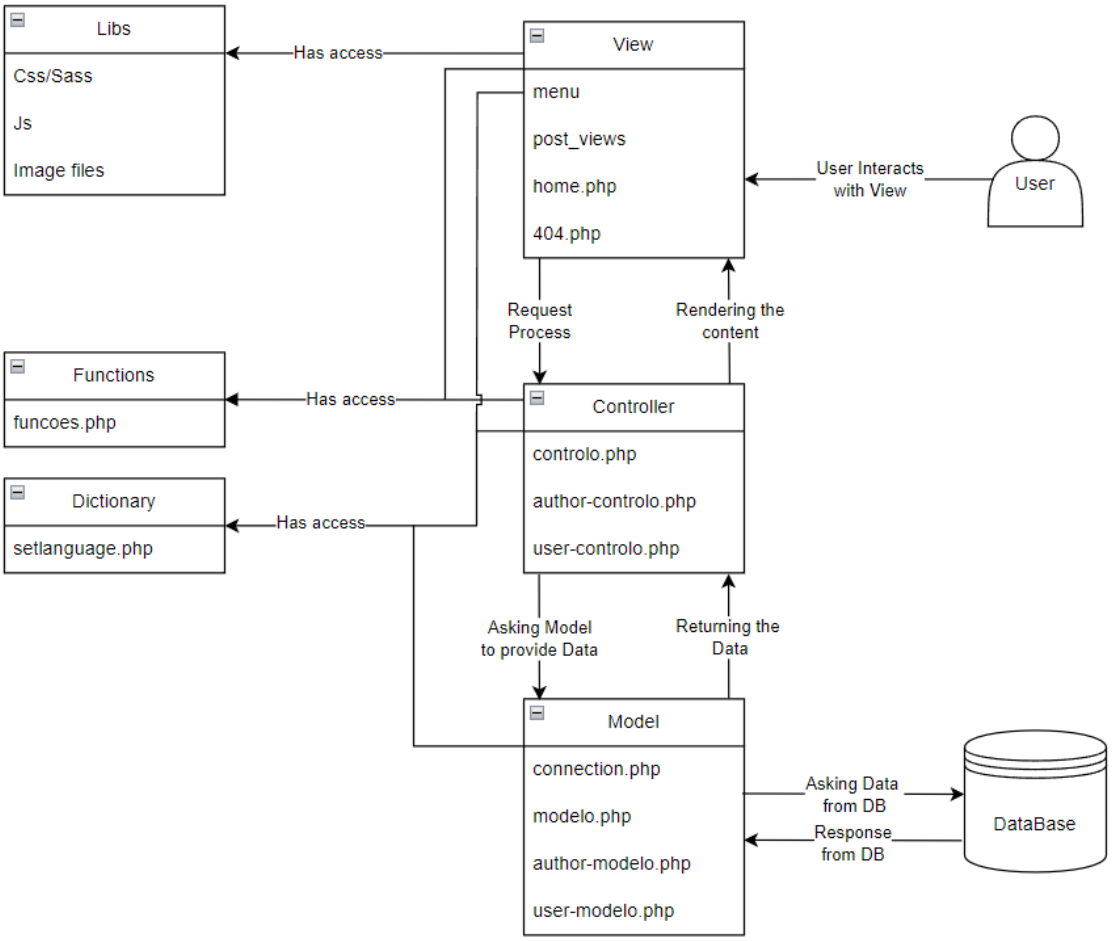


Figura 28 - Arquitetura do sistema

View (Apresentação):

- A camada de apresentação é responsável pela interface com o utilizador. É composta por diferentes arquivos de apresentação, como menu, post_views, home.php e 404.php, entre muitos outros.

- Esta camada vai renderizar o conteúdo que será mostrado ao utilizador final e interage diretamente com o utilizador.

- Recebe as solicitações dos utilizadores e processa as requisições, encaminhando-as para o controlador.

Controller (Controlador):

- O controlador é a camada intermediária que faz a mediação entre a apresentação e o modelo.

- É responsável por processar as requisições recebidas da camada de visão e acionar os modelos apropriados para obter os dados necessários.

- Inclui arquivos como controlo.php, author-controlo.php, e user-controlo.php.

- Depois de receber os dados do modelo, o controlador envia-os de volta à camada de apresentação para serem exibidos ao utilizador.

Model (Modelo):

- A camada de modelo é responsável pela lógica de negócios e interação com a base de dados.

- Contém arquivos como connection.php, modelo.php, author-modelo.php, e user-modelo.php.

- Ela solicita dados da base de dados e retorna esses dados ao controlador, que então os processa e envia de volta ao controlador.

Outras Componentes:

- **Libs (Bibliotecas):** Inclui arquivos de estilo CSS/Sass, scripts JavaScript e outros arquivos de imagem que é acedido pela camada de apresentação.

- **Functions (Funções):** Um arquivo funcoes.php que contém funções reutilizáveis, acedido pela camada de apresentação e controle.

- **Dictionary (Dicionário):** Um arquivo setlanguage.php que lida com a configuração do idioma do sistema, acedido por todas as camadas.

Esta divisão clara entre as camadas de apresentação, controle e modelo, junto com os componentes auxiliares, não só simplifica a manutenção e os testes do sistema, mas também permite uma maior flexibilidade na adaptação e expansão da aplicação, tornando-a uma escolha arquitetural robusta para projetos de software modernos.

5.3. Design da Base de Dados

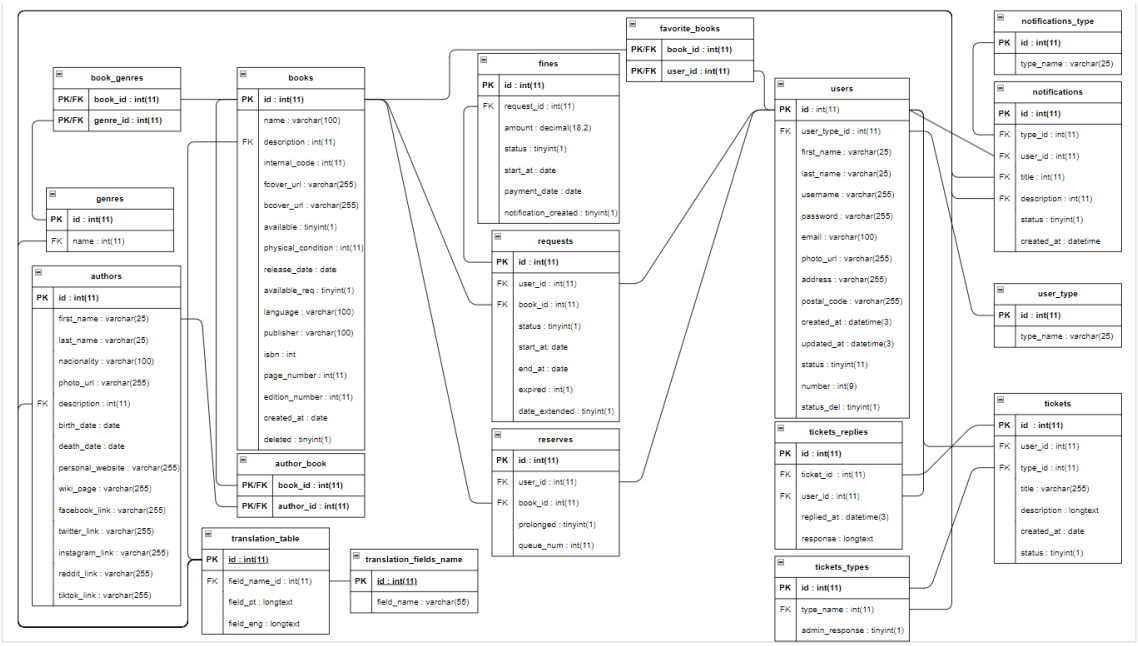


Figura 29 - Design da Base de Dados

O design da base de dados é uma parte fundamental do projeto, pois define como os dados são armazenados, organizados e acedidos. Para este projeto, foi utilizado o MySQL como Sistema de Gestão de Base de Dados (SGBD) devido à sua robustez, facilidade de integração com PHP e ampla documentação.

Estrutura das Tabelas Principais

A base de dados foi desenhada para refletir a estrutura e lógica do sistema, dividindo os dados em várias tabelas inter-relacionadas. Esta organização garante a normalização dos dados, evita redundâncias e assegura que a informação é armazenada de forma eficiente. Abaixo, descreve-se as tabelas principais do esquema:

1. Tabela users:

Armazena as informações dos utilizadores, como nome, apelido, username, password, email, entre outros. A tabela também contém uma chave estrangeira (user_type_id) que se refere à tabela user_type, determinando o tipo de utilizador (admin, staff, user).

2. Tabela books:

Contém os detalhes dos livros disponíveis no sistema, como título, descrição, idioma, código interno, URL das capas, e outras informações relacionadas. Esta tabela possui chaves estrangeiras para gerir géneros e autores através de tabelas auxiliares como book_genres e author_book.

Tabela requests:

Regista os pedidos de livros feitos pelos utilizadores, incluindo o estado do pedido, datas de início e fim, e uma chave estrangeira que liga ao utilizador e ao livro requisitado. Esta tabela é essencial para gerir o fluxo de requisições no sistema.

Tabela reserves:

Similar à tabela requests, esta tabela armazena as reservas de livros feitas pelos utilizadores, com informações sobre a ordem na fila de espera e se a reserva foi prolongada.

Tabela fines:

Regista as multas aplicadas aos utilizadores por atraso na devolução de livros. Contém o valor da multa, estado de pagamento e datas relevantes.

Tabela notifications:

Gere as notificações enviadas aos utilizadores, com detalhes sobre o tipo de notificação, título, descrição e estado.

Tabelas de Apoio (authors, genres, user_type, etc.):

Estas tabelas fornecem informações adicionais que são referenciadas pelas tabelas principais, como a lista de autores, géneros, e tipos de utilizadores.

Tabelas de Tradução (translation_table, translation_fields_name):

Estas tabelas facilitam a tradução de campos específicos para suportar diferentes idiomas no sistema, permitindo uma interface com duas línguas distintas.

Relacionamentos e Funcionalidades Extras

As tabelas estão interligadas por chaves estrangeiras, garantindo a integridade referencial e permitindo consultas complexas entre os diferentes conjuntos de dados. Por exemplo, um user pode ter múltiplas requests ou reserves, e cada book pode estar associado a vários genres. Estes relacionamentos são fundamentais para o correto funcionamento do sistema, permitindo operações como gestão de requisições, cálculo de multas e envio de notificações.

Em comparação com a versão inicial da base de dados, foram adicionados diversos campos às tabelas para suportar funcionalidades adicionais e armazenar informações complementares. Estes campos asseguram que o sistema seja flexível e capaz de evoluir conforme necessário, suportando novas características e integrando dados que enriquecem a experiência do utilizador e a gestão do sistema.

Este design de base de dados foi cuidadosamente planeado para suportar as operações necessárias no sistema, garantindo eficiência, flexibilidade e a capacidade de adaptação às necessidades futuras.

5.4. Organização do Diretório

O diretório principal deste projeto é constituído por diversas pastas e ficheiros essenciais para a construção do website.

A pasta “modelo” é responsável pela comunicação com a base de dados. Dentro dela, encontramos os seguintes ficheiros:

1. connection.php: Contém as configurações para estabelecer a conexão com a base de dados.

2. modelo.php: Agrupa as funções relacionadas à manipulação de dados sobre livros.
3. user-modelo.php: Agrupa as funções relacionadas à manipulação de dados sobre usuários.
4. author-modelo.php: Agrupa as funções relacionadas à manipulação de dados sobre autores.

A pasta “dicionario” armazena todas as strings (ou textos) utilizadas em ambas as línguas do projeto: português e inglês. Cada língua possui sua própria pasta, facilitando a organização e manutenção dos textos.

A pasta “funcoes” contém o ficheiro com funções gerais utilizadas em todo o projeto, como funções utilitárias que não requerem interação com a base de dados.

A pasta “libs” armazena todos os ficheiros de recursos necessários para o funcionamento do site, incluindo imagens, scripts JavaScript, folhas de estilo CSS e arquivos Sass. Os arquivos Sass são pré-processadores CSS que permitem escrever estilos de forma mais eficiente e organizada.

A pasta “controlo” contém os ficheiros que estabelecem a comunicação entre a base de dados e o front-end. Cada ficheiro está associado a um controlador (por exemplo, o user-modelo está associado ao user-controlo). Estes ficheiros são responsáveis por manipular os dados, filtrando-os e formatando-os antes de serem enviados para o front-end ou para a base de dados.

A pasta “views” contém todas as páginas do website. Dentro dela, existe uma pasta "menu" que armazena o template da barra lateral e os ficheiros de configuração dos links do menu. A pasta “views” também contém uma subpasta “post-views” que armazena os templates parciais que são carregados dinamicamente por jQuery para compor o conteúdo de cada página. Estes templates parciais são responsáveis por estabelecer a ligação com os modelos, não só exibindo os dados provenientes da base dados, mas também permitindo a submissão de dados para o servidor.

O ficheiro “.htaccess” é uma ferramenta poderosa para personalizar o comportamento do servidor web Apache. Ao redirecionar todas as requisições para o index.php, ele centraliza a lógica da aplicação, facilitando a gestão e o desenvolvimento do site. Esta

configuração permite a criação de URLs amigáveis e dinâmicas, melhorando a experiência do utilizador e o posicionamento do site nos motores de busca.

O ficheiro "index.php" serve como ponto de entrada único para o projeto. Inclui os templates do cabeçalho e rodapé e, entre eles, requer o ficheiro "router.php".

O ficheiro "router.php" é responsável por determinar qual conteúdo deve ser exibido na página, com base na URL solicitada pelo utilizador, carregando o template correspondente.

O ficheiro "setlanguage.php" define o idioma ativo no website.

O ficheiro "verifica-sessao.php" é chamado num template para verificar se o utilizador tem uma sessão ativa. Caso contrário, é redirecionado.

Os ficheiros "verificar_admin.php" e "verificar_funcionario.php" são chamados num template para verificar se o utilizador com sessão ativa tem autorização para aceder a esse conteúdo. Caso contrário, é redirecionado.

5.5.Desenvolvimento do Backend

No desenvolvimento do backend deste projeto, foram implementadas várias funcionalidades essenciais que garantem o funcionamento eficiente do sistema, desde a gestão de dados à autenticação de utilizadores. Este capítulo detalha a estrutura e implementação do backend, com foco na forma como a camada de modelo e controlo interagem para processar e manipular dados. O backend foi desenvolvido de acordo com uma arquitetura em três camadas: apresentação, controlo e modelo.

Vamos começar pela camada de modelo, onde é estabelecida a conexão com a base de dados e onde são implementados os métodos para interagir com os dados. Esta camada é crucial, pois é responsável por todas as operações de acesso e manipulação de dados no sistema. O ponto de partida para essa camada é o ficheiro de conexão com a base de dados, connection.php.

Ficheiro de Conexão: connection.php

O ficheiro connection.php é responsável por estabelecer e encerrar a conexão com a base de dados. Este ficheiro contém a classe coneccao, que foi projetada para gerir a conexão ao MySQL, utilizando a extensão mysqli. Abaixo está uma explicação detalhada do código:

```php
<?php
class coneccao
{
    // Propriedade pública para armazenar a conexão ao base de dados
    public $condb;
    // Propriedades privadas para as configurações de conexão
    private $host;
    private $user;
    private $pass;
    private $db;

    // Construtor da classe - define as configurações de conexão ao base de dados
    public function __construct()
    {
        // Definir o nome do servidor (normalmente 'localhost')
        $this->host = "localhost";
        // Definir o nome de utilizador para aceder ao MySQL
        $this->user = "root";
        // Definir a palavra-passe para o utilizador do MySQL
        $this->pass =  "";
        // Definir o nome da base de dados à qual se vai conectar
        $this->db = "aminhabiblioteca";
    }

    // Método para abrir a conexão com o base de dados MySQL
    public function open_db()
    {
        // Criar uma nova instância de mysqli para estabelecer a conexão com o base de dados
        $this->condb = new mysqli($this->host, $this->user, $this->pass, $this->db);

        // Verificar se ocorreu algum erro ao tentar conectar
        if ($this->condb->connect_error) {
            // Se houver um erro, exibir uma mensagem e terminar o script
            die("Erro na conexão: " . $this->condb->connect_error);
        }
    }
    // Método para fechar a conexão com o base de dados
    public function close_db()
    {
        // Verificar se a conexão foi estabelecida antes de tentar fechá-la
        if ($this->condb) {
            // Fechar a conexão ao base de dados
            $this->condb->close();
        }
    }
}
?>
```

Figura 30 - Código do connection.php

Estrutura e Funcionalidade

- Propriedades da Classe:

public $condb : Esta propriedade é utilizada para armazenar a instância da conexão com a base de dados, permitindo o seu uso em outros métodos e classes.

private $host, private $user, private $pass, private $db : Estas propriedades privadas armazenam as informações necessárias para a conexão, como o host do servidor, nome de utilizador, palavra-passe e o nome da base de dados.

- Construtor:

O método __construct() é utilizado para definir as configurações básicas de conexão. Ao instanciar a classe, as propriedades são automaticamente configuradas com os valores adequados para o ambiente de desenvolvimento.

- Método open_db():

Este método é responsável por estabelecer a conexão com a base de dados. Utilizando as propriedades configuradas no construtor, o método cria uma nova instância de mysqli e verifica se a conexão foi bem-sucedida. Em caso de erro, a execução é interrompida e uma mensagem de erro é exibida.

- Método close_db():

Este método garante que a conexão seja fechada adequadamente após as operações de base de dados estarem concluídas, liberando os recursos utilizados.

A classe coneccao em connection.php é um componente fundamental para a comunicação entre o sistema e a base de dados. Ao encapsular a lógica de conexão dentro de uma classe, o código torna-se mais modular e fácil de manter, permitindo a reutilização da conexão em diferentes partes do sistema. Este é o ponto de partida para a construção da camada de modelo, onde serão implementadas as interações diretas com as tabelas da base de dados.

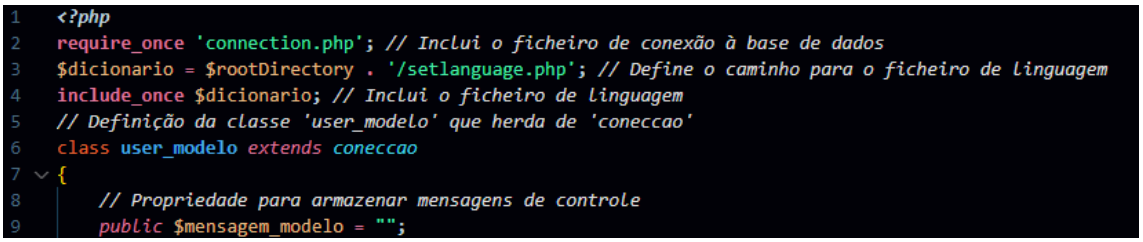
Ficheiro de Conexão: connection.php

Ficheiros modelo

Na arquitetura de três camadas, os modelos são responsáveis por interagir diretamente com a base de dados. Eles contêm a lógica necessária para manipular as informações armazenadas, permitindo que a aplicação insira, atualize, elimine e recupere dados. Todos os modelos seguem uma estrutura comum que facilita a sua manutenção e reutilização.

Estrutura Comum dos Modelos

Cada modelo começa com a inclusão de ficheiros essenciais e a definição de uma classe que herda as funcionalidades da classe de **conexão à base de dados**. Esta abordagem garante que todos os modelos possam estabelecer e fechar a ligação com a base de dados de forma consistente. A seguir está a estrutura comum dos modelos:



```php
<?php
require_once 'connection.php'; // Inclui o ficheiro de conexão à base de dados
$dicionario = $rootDirectory . '/setlanguage.php'; // Define o caminho para o ficheiro de linguagem
include_once $dicionario; // Inclui o ficheiro de linguagem
// Definição da classe 'user_modelo' que herda de 'coneccao'
class user_modelo extends coneccao
{
    // Propriedade para armazenar mensagens de controle
    public $mensagem_modelo = "";
```

Figura 31 - Código da estrutura comum dos modelos

Explicação dos Componentes:

1. require_once 'connection.php';

Este comando garante que o ficheiro connection.php, que contém a lógica para conectar à base de dados, seja incluído no modelo. O uso de require_once assegura que este ficheiro só seja incluído uma vez, prevenindo erros de duplicação.

2. Definição da variável $dicionario

O caminho para o ficheiro de configuração de idioma é definido pela variável $dicionario, que utiliza a variável global $rootDirectory para encontrar a localização correta do ficheiro setlanguage.php. Isto permite que o sistema adapte a linguagem de exibição de acordo com as preferências do utilizador.

3. include_once $dicionario;

A inclusão do ficheiro setlanguage.php garante que todas as variáveis de tradução definidas no sistema estejam disponíveis no modelo. Assim, o modelo pode fornecer textos ou mensagens traduzidas conforme necessário, garantindo que o sistema seja multilíngue.

4. Classe que herda de coneccao

A classe do modelo, como user_modelo, herda da classe coneccao. Esta herança permite que o modelo aceda diretamente aos métodos de conexão à base de dados definidos em connection.php, como open_db() e close_db(). Deste modo, a conexão à base de dados pode ser aberta e fechada facilmente sempre que o modelo precisar interagir com os dados.

5. Propriedade $mensagem_modelo

Esta variável pública é usada para armazenar mensagens que o modelo possa gerar durante o seu funcionamento, como mensagens de sucesso ou erros. Por exemplo, quando o modelo tenta adicionar um novo registo à base de dados, uma mensagem pode ser atribuída a esta propriedade indicando o resultado da operação.

Métodos de Interação com a Base de Dados

Os métodos apresentados a seguir fazem parte da camada de modelo e são responsáveis pela comunicação direta com a base de dados. Cada um deles foi desenvolvido para realizar operações específicas, como verificação de utilizadores, remoção de reservas, pesquisa de livros, entre outras. A estrutura dos métodos é pensada para garantir a segurança e a eficiência, utilizando consultas preparadas para evitar injeções SQL e mantendo uma abordagem robusta no tratamento de erros.

1. Método: verificar_user($username, $email)

```php
// Método para verificar se um utilizador já existe na base de dados
function verificar_user($username, $email)
{
    try {
        // Abre a conexão com a base de dados
        $this->open_db();

        // Prepara a consulta SQL para selecionar utilizadors com o mesmo nome de utilizador ou email
        $query = $this->condb->prepare("SELECT * FROM users WHERE username = ? OR email = ?");
        $query->bind_param("ss", $username, $email);
        $query->execute();
        $query->store_result();

        // Obtém o número de linhas retornadas pela consulta
        $count = $query->num_rows;

        // Fecha a consulta e a conexão com a base de dados
        $query->close();
        $this->close_db();

        // Retorna true se o utilizador existir, caso contrário, retorna false
        return $count > 0;
    } catch (Exception $e) {
        // Em caso de erro, fecha a conexão e lança a exceção
        $this->close_db();
        throw $e;
    }
}
```

Figura 32 - Código do método verificar_user

Propósito: Verificar se já existe um utilizador com o mesmo nome de utilizador ou email no sistema.

Passos principais:

A. Abrir conexão: O método open_db() é chamado para estabelecer a conexão com a base de dados.
B. Consulta SQL: Uma instrução SELECT é preparada para procurar utilizadores com o mesmo nome de utilizador ou email.
C. Ligação de parâmetros: Os parâmetros $username e $email são ligados à consulta através de bind_param("ss", $username, $email), garantindo que a consulta é segura contra SQL injection.

D. Executar e armazenar resultado: A consulta é executada e o número de linhas retornadas (num_rows) é verificado para determinar se existe ou não um utilizador.
E. Fechar consulta e conexão: A consulta e a conexão são fechadas.
F. Retorno: O método retorna true se o utilizador já existir, ou false caso contrário.

2. Método: remover_reserva($reserva_id)

```php
function remover_reserva($reserva_id)
{
    global $bd_erro;
    try {
        $this->open_db();
        $query = $this->condb->prepare("CALL Remover_Reserva(?);");
        $query->bind_param("i", $reserva_id);
        $result = $query->execute();
        $query->close();
        $this->close_db();
        return $result;
    } catch (Exception $e) {
        // Em caso de erro, fecha a conexão e retorna a mensagem de erro
        $this->close_db();
        return $bd_erro;
    }
}
```

Figura 33 - Código do método remover_reserva

Propósito: Remover uma reserva específica através de um procedimento armazenado no MySQL.

Passos principais:

A. Chamada ao procedimento: Utiliza CALL Remover_Reserva(?) para executar um procedimento armazenado que remove uma reserva.
B. Ligação de parâmetro: O identificador da reserva ($reserva_id) é passado como parâmetro para o procedimento.
C. Execução: A consulta é executada, e o resultado é guardado para saber se a operação foi bem-sucedida.
D. Fechar consulta e conexão: Como nos métodos anteriores, a consulta e a conexão são fechadas.

E. Retorno: O método retorna o resultado da execução do procedimento armazenado, indicando sucesso ou falha.

3. Método: extender_requisicao($requisicao_id, $end_date)

```php
function extender_requisicao($requisicao_id, $end_date)
{
    global $bd_erro;
    try {
        $this->open_db();

        $query = $this->condb->prepare("UPDATE requests
            SET
                end_at = ?,
                date_extended = 1
            WHERE id = ?");
        $query->bind_param("si", $end_date, $requisicao_id);
        $result = $query->execute();
        $query->close();
        $this->close_db();
        return $result;
    } catch (Exception $e) {
        // Em caso de erro, fecha a conexão e retorna a mensagem de erro
        $this->close_db();
        return $bd_erro;
    }
}
```

Figura 34 - Código método extender_requisicao

Propósito: Estender a data de fim de uma requisição de livro e marcar a requisição como estendida.

Passos principais:

A. Consulta de atualização: O método usa UPDATE requests para alterar a data de término (end_at) e marca a requisição como estendida (date_extended = 1).

B. Ligação de parâmetros: A nova data de fim ($end_date) e o identificador da requisição ($requisicao_id) são passados como parâmetros para a consulta.

C. Execução e retorno: A consulta é executada, e o resultado da execução (se foi bem-sucedida ou não) é retornado.

4. Método: contador_livros()

```php
function contador_livros()
{
    global $bd_erro;
    try {
        $this->open_db();
        $query = $this->condb->prepare("SELECT COUNT(*) AS count_books
                                        FROM books WHERE deleted = 0;");
        //$query->bind_param("i", $user_id);
        $query->execute();
        $result = $query->get_result();
        $row = $result->fetch_assoc(); // Obtemos apenas uma linha

        $query->close();
        $this->close_db();
        $nlivros = $row['count_books'];
        return $nlivros;
    } catch (Exception $e) {
        $this->close_db();
        return $bd_erro;
    }
}
```

Figura 35 - Código método contador_livros

Propósito: Contar o número total de livros disponíveis (não eliminados) no sistema.

Passos principais:

A. Consulta SQL: Uma instrução SELECT COUNT(*) é usada para contar o número de livros cuja coluna deleted está marcada como 0 (não eliminados).

B. Execução e obtenção do resultado: A consulta é executada e o resultado é extraído com fetch_assoc(), obtendo o número total de livros.

C. Retorno: O método retorna o número total de livros disponíveis.

5. Método: livros_populares_home()

```php
public function livros_populares_home()
{
    global $bd_erro;
    try {
        $this->open_db();
        $query = $this->condb->prepare("SELECT * FROM popular_books");
        //$query->bind_param("s", $user_id);
        $query->execute();

        // Obtém o resultado da consulta
        $res = $query->get_result();

        // Fecha a consulta e a conexão com a base de dados
        $query->close();
        $this->close_db();

        // Retorna o resultado da consulta
        return $res;
    } catch (Exception $e) {
        // Em caso de erro, fecha a conexão e retorna a mensagem de erro
        $this->close_db();
        return $bd_erro;
    }
}
```

Figura 36 - Código método livros_populares_home

Propósito: Obter a lista de livros populares para exibição na página inicial.

Passos principais:

A. Consulta SQL: Simplesmente usa SELECT * FROM popular_books para obter todos os registos da view popular_books.
B. Execução e obtenção do resultado: A consulta é executada e o resultado é obtido com get_result().
C. Retorno: O método retorna o resultado da consulta, que pode ser usado para listar os livros populares.

6. Método: pesquisa_livros($bookname, $authorname, $genero, $linguagem, $editora, $disponibilidade)

```php
public function pesquisa_livros($bookname, $authorname, $genero, $linguagem, $editora, $disponibilidade)
{
    global $bd_erro;
    try {
        $this->open_db();

        $comando_base = "
    SELECT DISTINCT
        b.id,
        b.title,
        b.internal_code,
        b.fcover_url
    FROM
        books b
    LEFT JOIN
        author_book ab ON b.id = ab.book_id
    LEFT JOIN
        author a ON ab.author_id = a.id
    LEFT JOIN
        book_genres bg ON b.id = bg.book_id
    LEFT JOIN
        genres g ON bg.genre_id = g.id
    WHERE 1=1
    AND b.deleted = 0"; // Adiciona uma cláusula sempre verdadeira para facilitar a concatenação de cláusulas

        $params = array();
        $tipos = ''; // String para armazenar os tipos dos parâmetros (s = string, i = integer, etc.)

        if (!empty($bookname)) {
            $comando_base .= " AND b.title LIKE ?";
            $params[] = '%' . $bookname . '%';
            $tipos .= 's';
        }

        if (!empty($authorname)) {
            $comando_base .= " AND (a.first_name LIKE ? OR a.last_name LIKE ?)";
            $params[] = '%' . $authorname . '%';
            $params[] = '%' . $authorname . '%';
            $tipos .= 'ss'; // Dois parâmetros tipo string (first_name e last_name)
        }

        if (!empty($linguagem)) {
            $comando_base .= " AND b.language = ?";
            $params[] = $linguagem;
            $tipos .= 's';
        }
```

Figura 37 - Código método pesquisa_livros parte1

```php
        if (!empty($editora)) {
            $comando_base .= " AND b.publisher = ?";
            $params[] = $editora;
            $tipos .= 's';
        }

        if (!empty($genero)) {
            $comando_base .= " AND g.id = ?";
            $params[] = $genero;
            $tipos .= 'i';
        }

        if ($disponibilidade !== null) {
            if ($disponibilidade == 0 || $disponibilidade == 1) {
                $comando_base .= " AND b.available = ? AND b.available_req = 1";
                $params[] = $disponibilidade;
                $tipos .= 'i';
            } elseif ($disponibilidade == 10) {
                $comando_base .= " AND b.available_req = 0";
            }
        }

        // Prepare a declaração final
        $stmt = $this->condb->prepare($comando_base);
        if (!$stmt) {
            throw new Exception("Erro na preparação da consulta SQL: " . $this->condb->error);
        }

        // Vincule os parâmetros dinamicamente
        if (!empty($params)) {
            $stmt->bind_param($tipos, ...$params);
        }

        // Execute a consulta
        $stmt->execute();
        $res = $stmt->get_result();

        $stmt->close();
        $this->close_db();

        return $res;
    } catch (Exception $e) {
        $this->close_db();
        throw $e; // Aqui estava com um erro de digitação, corrigido para 'throw $e'
    }
}
```

Figura 38 – Código método pesquisa_livros parte2

Propósito: Pesquisar livros com base em vários critérios fornecidos pelo utilizador.

Passos principais:

A. Construção da consulta dinâmica: A consulta SQL começa com uma cláusula SELECT DISTINCT e vai sendo dinamicamente construída conforme os critérios fornecidos (título do livro, autor, género, idioma, editora, disponibilidade).
B. Ligação de parâmetros: Os parâmetros de pesquisa são armazenados em arrays e vinculados à consulta conforme necessário, usando bind_param().
C. Execução e obtenção do resultado: A consulta é executada, e o resultado é retornado como um conjunto de registos.
D. Retorno: O método retorna o conjunto de resultados da pesquisa, que pode ser usado para exibir os livros encontrados.

Todas estes métodos têm várias etapas em comum, como:

Abertura e Fecho da Conexão: Cada método começa com a abertura da conexão (open_db()) e termina com o fecho da conexão (close_db()), garantindo que a base de dados esteja sempre acessível quando necessário e que os recursos sejam liberados corretamente após a operação.

Execução Segura com prepare e bind_param: Para evitar SQL injection, todas as consultas SQL são preparadas com prepare(), e os parâmetros são vinculados com bind_param(). Isso garante que os dados fornecidos pelos utilizadores sejam tratados de forma segura.

Tratamento de Exceções: Em caso de erro, as exceções são capturadas e a conexão com a base de dados é sempre fechada. Em alguns casos, o método também lança a exceção para ser tratada em camadas superiores, garantindo que o sistema lide com erros de maneira robusta.

Esses métodos exemplificam como a camada de modelo interage com a base de dados de forma consistente, segura e eficiente.

Ficheiros controlo

A camada de controlo é responsável por mediar a interação entre a camada de modelo e a visualização, coordenando a lógica da aplicação e a resposta às requisições do utilizador. Todos os controladores no sistema compartilham uma estrutura semelhante, onde são incluídos os arquivos essenciais para a operação, como o modelo de dados, o dicionário de linguagem e funções utilitárias. Cada controlador cria uma instância da classe de modelo, permitindo o acesso aos métodos que interagem com a base de dados.

Estrutura comum dos controladores:

```php
<?php
$rootDirectory = dirname(__DIR__);
$modelo = $rootDirectory . '/modelo/modelo.php';
$dicionario = $rootDirectory . '/setlanguage.php';
$funcoes = $rootDirectory . '/funcoes/funcoes.php';
require $modelo;
include_once $dicionario;
include_once $funcoes;

class controlo
{
    public $mensagem = "";
    public $control;

    function __construct()
    {
        $this->control =  new modelo();
    }

```

Figura 39 – Código estrutura comum dos controladores

Nesta estrutura, observa-se que:

$rootDirectory: Define o diretório raiz do projeto.

$modelo, $dicionario, $funcoes: São os caminhos para os arquivos de modelo, dicionário e funções.

require e include_once: Garantem que esses arquivos sejam carregados para a execução do controlo.

$this->control: Instância a classe do modelo, permitindo o uso dos métodos que manipulam a base de dados.

Métodos de Controle e Lógica

Os métodos que seguem fazem parte da camada de controlo e são responsáveis por coordenar as ações entre o modelo e as visualizações. Estes métodos implementam a lógica da aplicação, invocando funções do modelo para aceder à base de dados, processando os dados e definindo a resposta adequada às interações dos utilizadores. A estrutura visa a clareza e eficiência na gestão de dados e na interação com as funcionalidades do sistema.

1. Método categorias_populares_home

```php
public function categorias_populares_home()
{
    global $setlang;
    $cate_pop = $this->control->categorias_populares_home();
    if ($cate_pop instanceof mysqli_result) {
        $cate_pop_home = [];
        while ($linha = $cate_pop->fetch_assoc()) {
            $categoria = "";
            switch ($setlang) {
                case 'pt':
                    $categoria = $linha['genre_name_pt'];
                    break;
                case 'eng':
                    $categoria = $linha['genre_name_eng'];
                    break;
                default:
                    $categoria = $linha['genre_name_pt'];
                    break;
            }
            $cate_pop_home[] = [
                "categoria" => $categoria,
                "num_requests" => $linha['num_requests']
            ];
        }
        return $cate_pop_home;
    }
}
```

Figura 40 - Código método categorias_populares_home

Este método é utilizado para obter as categorias mais populares de livros com base no idioma selecionado pelo utilizador. O método chama o modelo para recuperar os dados e utiliza o idioma configurado no sistema para determinar se os nomes das categorias devem ser exibidos em português ou em inglês.

Estrutura:

A. Obtém as categorias populares através do método correspondente no modelo.
B. Usa um switch para ajustar o nome da categoria com base no idioma selecionado (pt ou eng).
C. Devolve um array associativo contendo o nome da categoria e o número de pedidos associados a ela.

2. Método pesquisa_rapida_home

```php
public function pesquisa_rapida_home($nome)
{
    $results = [];

    // Pesquisar por livros
    $pesquisa_rapida_livro = $this->control->pesquisa_rapida_livro($nome);
    if ($pesquisa_rapida_livro instanceof mysqli_result) {
        while ($row = $pesquisa_rapida_livro->fetch_assoc()) {
            $row['type'] = 'book';
            $results[] = $row;
        }
    }

    // Pesquisar por autores
    $pesquisa_rapida_autor = $this->control->pesquisa_rapida_autor($nome);
    if ($pesquisa_rapida_autor instanceof mysqli_result) {
        while ($row = $pesquisa_rapida_autor->fetch_assoc()) {
            $row['type'] = 'author';
            $results[] = $row;
        }
    }

    // Ordenar resultados pelo nome
    usort($results, function ($a, $b) {
        return strcmp($a['name'], $b['name']);
    });

    return [
        'success' => true,
        'data' => $results
    ];
}
```

Figura 41 - Código método pesquisa_rapida_home

Este método permite realizar uma pesquisa rápida por livros e autores com base no nome fornecido. Ele retorna os resultados organizados e classificados por nome.

Estrutura:

A. Chama o modelo para pesquisar livros e autores separadamente.
B. Junta os resultados num único array e identifica o tipo de cada resultado (book ou author).
C. Ordena os resultados alfabeticamente pelo nome.

3. Método verificar_favorito

```php
public function verificar_favorito($user_id, $book_id)
{
    $verificar = $this->control->verificar_favorito($user_id, $book_id);
    if ($verificar > 0) {
        return true;
    } else {
        return false;
    }
}
```

Figura 42 - Código método verificar_favorito

Este método verifica se um determinado livro já foi marcado como favorito por um utilizador específico.

Estrutura:

A. Chama o método no modelo para verificar se o utilizador já marcou o livro como favorito.
B. Retorna true ou false dependendo da existência do favorito.

4. Método requisitar_livro

```php
public function requisitar_livro($user_id, $book_id, $user_status)
{
    global $contabloqueada, $operafalhada, $reqs5, $contsuporte, $vermultas, $reqsucesso, $levantarlivro, $reqinfor;
    $verifica_livro_requisitado = $this->control->verifica_livro_requisitado($user_id, $book_id);
    $ver_livro_req = ($verifica_livro_requisitado === $user_id) ? true : false;
    if ($user_status != 0 && !$ver_livro_req) {
        $numero_reqs = $this->control->contador_requisicoes($user_id);
        $numero_revs = $this->control->contador_reservas($user_id);
        $numero_reqs_revs = $numero_reqs + $numero_revs;
        if ($numero_reqs < 5 && $numero_reqs_revs < 5) {
            $start_at = adicionarDoisDiasUteisAPartirDeHoje(); // metodo nas funcoes
            $requisitar = $this->control->requisitar_livro($user_id, $book_id, $start_at);
            if ($requisitar === true) {
                return [
                    'success' => true,
                    'data' => [
                        'reqsucesso' => $reqsucesso,
                        'levantarlivro' => $levantarlivro,
                        'reqinfor' => $reqinfor
                    ]
                ];
            } else {
                return [
                    'success' => false,
                    'data' => [
                        'reqfalhada' => $operafalhada,
                        'reqsucesso' => $requisitar
                    ]
                ];
            }
        } elseif ($numero_reqs == 5 || $numero_reqs_revs >= 5) {
            return [
                'success' => false,
                'data' => [
                    'reqfalhada' => $operafalhada,
                    'reqs5' => $reqs5,
                    'contsuporte' => $contsuporte,

                ]
            ];
        }
    } else {
        return [
            'success' => false,
            'data' => [
                'reqfalhada' => $operafalhada,
                'conta_block' => $contabloqueada,
                'vermultas' => $vermultas,
                'contsuporte' => $contsuporte,
            ]
        ];
    }
}
```

Figura 43 – Código método requisitar livro

Responsável por gerir o processo de requisição de um livro. Verifica o status do utilizador, o número de requisições atuais e efetua a requisição, se possível.

Estrutura:

A. Verifica se o livro já foi requisitado pelo utilizador.
B. Controla o número de requisições ativas e reservas para garantir que o limite permitido não seja excedido.

C. Chama o modelo para realizar a requisição e calcula a data de início, adicionando dois dias úteis à data atual.
D. Retorna mensagens de sucesso ou falha, dependendo do resultado da operação.

5. Método listar_todas_multas

```php
public function listar_todas_multas()
{
    $multas =  $this->control->listar_todas_multas();
    if ($multas instanceof mysqli_result) {
        $lista_multas = [];
        while ($linha = $multas->fetch_assoc()) {
            $lista_multas[] = [
                "id" => $linha['fine_id'],
                "request_id" => $linha['request_id'],
                "user_id" => $linha['user_id'],
                "username" => $linha['username'],
                "title" => $linha['title'],
                "internal_code" => $linha['internal_code'],
                "amount" => $linha['amount'],
                "start_at" => date("d-m-Y", strtotime($linha['start_at']))
            ];
        }
        return [
            'sucess'  => true,
            'data' => $lista_multas,
        ];
    }
}
```

Figura 44 - Código listar_todas_multas

Este método recupera todas as multas associadas às requisições de livros dos utilizadores.

Estrutura:

A. Chama o modelo para listar todas as multas.
B. Constrói um array com detalhes como o ID da multa, título do livro, valor da multa e data de início.
C. Retorna o array de dados organizados.

6. Método entregar_livro

```php
public function entregar_livro($requisicao_id)
{
    $entrega = $this->control->entregar_livro($requisicao_id);
    if ($entrega === true) {
        return true;
    } else {
        return false;
    }
}
```

Figura 45 - Código entregar_livro

Este método marca a entrega de um livro requisitado como concluída.

Estrutura:

A. Chama o método do modelo para atualizar o status da requisição como entregue.
B. Retorna true ou false, dependendo do sucesso da operação.
C. Esses métodos demonstram como o controlo funciona em conjunto com o modelo para gerir a lógica de negócio e manter o sistema em funcionamento.

Esses métodos demonstram como o controlo funciona em conjunto com o modelo para gerir a lógica de negócio e manter o sistema em funcionamento.

Deteção e Configuração Dinâmica do Idioma do Website

Esta secção aborda o processo de deteção e configuração dinâmica do idioma do website. A funcionalidade de escolha de idioma permite aos utilizadores selecionar entre português e inglês, garantindo uma experiência mais personalizada.

O processo de seleção de idioma é controlado por um ficheiro específico, chamado “setlanguage.php”, que é invocado em todas as views, models e controllers do website. Este ficheiro determina o idioma com base numa cookie armazenada no navegador do utilizador. Após a determinação da linguagem, são chamados os ficheiros de tradução correspondentes, que estão localizados dentro da pasta “dicionário”. A escolha do idioma pode ser alterada a qualquer momento através de várias secções do website, como o menu principal, formulários de login e registo, ou até mesmo na página de termos, políticas e privacidade. Este sistema assegura que o website é apresentado na língua preferida do utilizador, proporcionando uma navegação mais acessível e adaptada.

Explicação do Ficheiro "setlanguage.php"

O ficheiro "setlanguage.php" é responsável por gerir a seleção de idioma no website, determinando qual o idioma que deve ser carregado com base nas preferências do utilizador. Abaixo está uma explicação detalhada do fluxo de funcionamento deste ficheiro.

1. Verificação de Mudança de Idioma via Parâmetro *GET*

```php
if (isset($_GET['setlang'])) {
    $setlang = $_GET['setlang'];
    setcookie("lang", "$setlang", 2147483647, "/");
}
```

Figura 46 – Código verificação de mudança de idioma

Descrição: O primeiro bloco de código verifica se foi passado um parâmetro "setlang" através da URL. Este parâmetro é utilizado para indicar uma mudança de idioma, podendo ser definido, por exemplo, através de um botão que o utilizador clica para mudar a língua do website.

Ação: Se o parâmetro "setlang" estiver definido, o código guarda o valor correspondente ('pt' para português ou 'eng' para inglês) numa cookie chamada "lang". Esta cookie tem uma data de expiração muito longa (2147483647), o que faz com que seja praticamente persistente.

Propósito: Isto permite que a escolha do idioma seja lembrada mesmo após o utilizador fechar o navegador ou sair do site.

2. Definição do Idioma Padrão

```php
if (!isset($_COOKIE['lang'])) {
    setcookie("lang", "pt", 2147483647, "/");
}
```

Figura 47 – Código definição do idioma padrão

Descrição: Se a cookie "lang" não estiver definida (ou seja, se é a primeira vez que o utilizador visita o site ou se ele limpou as cookies), o código define o idioma padrão para português ('pt').

Ação: É criada uma cookie “lang” com o valor 'pt', que será utilizada em todas as visitas subsequentes ao site.

3. Leitura do Idioma da Cookie

```php
if(isset($_COOKIE['lang'])){
    $setlang = $_COOKIE['lang'];
}else{
    $setlang = null;
}
```

Figura 48 – Código leitura do idioma da cookie

Descrição: Este bloco verifica se a cookie “lang” está definida. Se estiver, o valor dessa cookie é atribuído à variável “$setlang”, que será utilizada mais tarde para determinar qual o ficheiro de idioma a ser carregado.

Ação: Caso a cookie não esteja definida (o que é improvável neste ponto, devido ao código anterior que define o idioma padrão), a variável “$setlang” será nula.

4. Carregamento do Ficheiro de Idioma Correspondente

```php
switch ($setlang) {
    case 'pt':
        include "dicionario/pt/pt-base.php";
        break;
    case 'eng':
        include "dicionario/eng/eng-base.php";
        break;
    default:
        include "dicionario/pt/pt-base.php";
        break;
}
```

Figura 49 – Código carregamento do ficheiro de idioma correspondente

Descrição: O bloco switch verifica o valor da variável “$setlang” e, com base nesse valor, inclui o ficheiro de idioma correspondente.

Ação: Se “$setlang” for 'pt', o ficheiro de idioma em português (pt-base.php) será incluído. Se for 'eng', será incluído o ficheiro de idioma em inglês (eng-base.php). Se por algum motivo o valor de “$setlang” não for nem 'pt' nem 'eng', o código por defeito inclui o ficheiro de português, garantindo que o site sempre carrega um idioma.

Propósito: Esta estrutura permite que o conteúdo do site seja apresentado na língua escolhida pelo utilizador, carregando as traduções apropriadas de acordo com o idioma selecionado.

Resumo

O ficheiro "setlanguage.php" é central para a funcionalidade multilingue do website. Ele verifica se o utilizador deseja mudar o idioma, guarda essa escolha numa cookie, e carrega o ficheiro de idioma correspondente com base na preferência do utilizador ou nas configurações padrão. Este fluxo garante que o idioma do site é consistente em todas as páginas e sessões, proporcionando uma experiência de utilizador personalizada e contínua.

```php
<?php
// Verifica se existe o parâmetro 'setlang' na URL, indicando uma mudança de idioma
if (isset($_GET['setlang'])) {
    $setlang = $_GET['setlang']; // Armazena o valor do idioma selecionado na variável $setlang
    // Define uma cookie chamada 'lang' com o valor do idioma escolhido, persistente por um longo período
    setcookie("lang", "$setlang", 2147483647, "/");
}

// Verifica se a cookie 'lang' não está definida, ou seja, se é a primeira visita do utilizador ao site
if (!isset($_COOKIE['lang'])) {
    // Define a cookie 'lang' com o valor 'pt' (português) como idioma padrão
    setcookie("lang", "pt", 2147483647, "/");
}

// Verifica se a cookie 'lang' está definida
if (isset($_COOKIE['lang'])) {
    $setlang = $_COOKIE['lang']; // Atribui o valor da cookie à variável $setlang
} else {
    $setlang = null; // Se a cookie não estiver definida, $setlang é nulo (o que não deve acontecer por causa do código anterior)
}

// Carrega o ficheiro de tradução correspondente ao idioma definido
switch ($setlang) {
    case 'pt':
        // Se $setlang for 'pt', inclui o ficheiro de tradução para português
        include "dicionario/pt/pt-base.php";
        break;
    case 'eng':
        // Se $setlang for 'eng', inclui o ficheiro de tradução para inglês
        include "dicionario/eng/eng-base.php";
        break;
    default:
        // Se $setlang tiver um valor inesperado, carrega o ficheiro de tradução para português como padrão
        include "dicionario/pt/pt-base.php";
        break;
}
```

Figura 50 – Código do ficheiro setlanguage.php

Explicação de Como o Utilizador Muda o Idioma

Neste processo, o utilizador pode mudar o idioma do website de forma dinâmica através de botões de seleção. Abaixo está a explicação detalhada de como isso funciona, incluindo a interação entre os botões e o script em jQuery.

1. Botões de Seleção de Idioma

Os botões apresentados ao utilizador são os seguintes:

```
<button class="setCookieBtn btn pe-1 primary nav-link <?php if (!isset($_COOKIE['lang']) || $_COOKIE['lang'] == "pt") {
                                                        echo "active";
                                                    } ?>" name="pt">
    <span class="span-lang">pt</span>
    <div class="bordinhaazulhorizontal position-relative">
        <img src="<?php echo get_link("")?>libs/img/Rectangle_18.png" alt="Rectangle_18.png">
    </div>
</button>
<button class="setCookieBtn btn ps-1 primary nav-link <?php if (isset($_COOKIE['lang']) && $_COOKIE['lang'] == "eng") {
                                                        echo "active";
                                                    } ?>" name="eng">
    <span class="span-lang">eng</span>
    <div class="bordinhaazulhorizontal position-relative">
        <img src="<?php echo get_link("")?>libs/img/Rectangle_18.png" alt="Rectangle_18.png">
    </div>
</button>
```

Figura 51 – Código botões de seleção de idioma

Estrutura dos Botões:

Classes: Cada botão tem a classe “setCookieBtn”, que é usada pelo script jQuery para identificar quando um botão é clicado. Além disso, os botões têm classes adicionais para estilização.

Nome do Idioma: O idioma que cada botão representa é armazenado no atributo name. O botão que representa o português tem name="pt", e o que representa o inglês tem name="eng".

Indicação do Idioma Ativo: A lógica PHP dentro das classes adiciona a classe “active” ao botão correspondente ao idioma atualmente selecionado. Isto é feito verificando o valor da cookie lang para determinar qual idioma está ativo e, em seguida, aplicando a classe “active” ao botão correspondente. Isto permite ao utilizador ver claramente qual idioma está atualmente selecionado.

2. Script em jQuery para Mudar o Idioma

Aqui está o script jQuery responsável por gerir a interação do utilizador com os botões:

```
$(".setCookieBtn").click(function () {
  var setlang = $(this).attr("name"); // Obtém o nome do idioma a partir do botão clicado
  $.ajax({
    type: "GET",
    url: "setlanguage.php", // Faz uma requisição ao ficheiro setlanguage.php
    data: {
      setlang: setlang, // Passa o idioma selecionado como parâmetro
    },
    success: function (response) {
      // Se a requisição for bem-sucedida, recarrega a página
      location.reload();
    },
  });
});
```

Figura 52 - Código script em JQuery para mudar o idioma

Clique no Botão: Quando o utilizador clica em um dos botões de idioma, o evento click é capturado pelo script jQuery.

Obtém o Idioma: O script extrai o valor do atributo name do botão clicado (que será 'pt' ou 'eng') e armazena-o na variável setlang.

Requisição AJAX: O script envia uma requisição AJAX do tipo GET para o ficheiro setlanguage.php, passando o idioma selecionado como parâmetro (setlang).

URL: A URL para a qual a requisição é enviada é setlanguage.php.

Parâmetros: O idioma escolhido é enviado como parâmetro de consulta na URL, por exemplo, setlanguage.php?setlang=pt.

Recarregamento da Página: Se a requisição AJAX for bem-sucedida, a função de callback success é acionada, e a página é recarregada para aplicar o novo idioma.

Explicação dos Ficheiros de Tradução

No processo de internacionalização do website, as traduções são geridas através de ficheiros específicos que se encontram na pasta dicionario, localizada na base do projeto. Para cada idioma disponível no website, existe uma pasta correspondente que contém os ficheiros de tradução. As pastas são nomeadas de acordo com o código do idioma: pt para português e eng para inglês.

Estrutura das Pastas e Ficheiros

Dentro de cada uma destas pastas (pt e eng), encontramos ficheiros que definem as traduções para os diferentes elementos do website. Cada pasta contém dois ficheiros principais:

- pt-base.php (para português)
- eng-base.php (para inglês)

Conteúdo dos Ficheiros de Tradução

Os ficheiros de tradução são responsáveis por definir variáveis que contêm o texto exibido no website. Estas variáveis são as mesmas em ambos os ficheiros, garantindo que o conteúdo do site é consistente, independentemente do idioma selecionado. O que varia entre os ficheiros é o valor atribuído a cada variável, ou seja, o texto que será exibido ao utilizador.

Exemplo de Tradução:

- pt-base.php:

```
$pagina_inicial_titulo = "Pagina inicial";
$login_titulo = "Inciar sessao";
$register_titulo = "Criar conta";
```

Figura 53 – Código exemplo de tradução em português

- eng-base.php:

```
$pagina_inicial_titulo = "Home";
$login_titulo = "Sign in";
$register_titulo = "Sign up";
```

Figura 54 – Código exemplo de tradução em inglês

Funcionamento da Mudança de Idioma

Quando o utilizador seleciona um idioma, o sistema carrega o ficheiro de tradução correspondente (pt-base.php ou eng-base.php). Todas as variáveis definidas nesse ficheiro são então usadas ao longo do website para exibir o conteúdo na língua escolhida. Se o utilizador muda o idioma, apenas o conteúdo das variáveis é alterado, enquanto o nome das variáveis permanece o mesmo, permitindo uma mudança fluida e consistente entre os diferentes idiomas.

Uso do Ficheiro "setlanguage.php"

O ficheiro "setlanguage.php" é fundamental para gerir o idioma do website e assegurar que o conteúdo é exibido na língua correta. Aqui está um resumo do seu funcionamento e aplicação:

1. Inclusão no Header:

O setlanguage.php é incluído no header do website usando include_once "setlanguage.php";. Isto garante que o idioma definido é carregado em todas as páginas do site.

2. Inclusão em Models e Controllers:

É também incluído em models e controllers para que qualquer texto ou mensagem processada por estas partes do sistema seja traduzido conforme o idioma selecionado.

3. Inclusão nas Views de Post:

O ficheiro é incluído nas views de post que geram conteúdos específicos para assegurar que o texto exibido é traduzido corretamente.

Funcionamento

Definição do Idioma: O “setlanguage.php” verifica a cookie lang para determinar o idioma a ser utilizado. Se a cookie não estiver presente, o idioma padrão (português) é aplicado.

Carregamento das Traduções: Baseado no idioma, o ficheiro inclui o ficheiro de tradução correspondente (como pt-base.php para português ou eng-base.php para inglês).

Disponibilidade das Traduções: As variáveis de tradução definidas nos ficheiros de tradução tornam-se disponíveis para uso em todo o site.

Este sistema assegura que o idioma selecionado pelo utilizador é aplicado de forma consistente em todas as partes do website, melhorando a experiência do utilizador com um conteúdo traduzido e adaptado às suas preferências.

Ficheiro funções.php

O ficheiro de funções centraliza vários métodos utilitários que são usados em diversas partes do sistema para evitar repetição de código e facilitar a manutenção. Um dos métodos mais importantes é o get_link, que foi inspirado em sistemas como o WordPress, onde funções similares são usadas para construir links de forma dinâmica com base numa estrutura de URL predefinida.

Este método e suas variações garantem que, ao gerar links dentro da aplicação, qualquer alteração no diretório base (como mudanças no domínio ou na estrutura de diretórios) possa ser facilmente aplicada sem a necessidade de modificar manualmente cada link individual.

Função get_link: Esta função é responsável por gerar um link completo com base no diretório base e no link fornecido. Ela concatena o diretório base, definido globalmente, com o link especificado, produzindo assim o URL final.

Função get_link_completo: Similar à função get_link, esta função também cria um link completo, mas permite a adição opcional de um código ao final do link. Isso é útil quando se precisa gerar URLs que incluem parâmetros dinâmicos.

Função adicionarDoisDiasUteisAPartirDeHoje: Esta função calcula uma data futura adicionando dois dias úteis à data atual. Inclui funções auxiliares para verificar se uma data é um fim de semana e para adicionar dias úteis, ignorando os finais de semana. Esta funcionalidade é útil para agendar eventos ou prazos que devem pular os dias não úteis.

Router

No desenvolvimento de aplicações web, o roteamento é uma parte fundamental do backend, responsável por encaminhar as solicitações do utilizador para os manipuladores apropriados, garantindo que a aplicação responda de forma adequada a diferentes URLs. Neste contexto, explicarei como o roteamento é configurado nesta plataforma utilizando o ficheiro .htaccess e o código PHP do router.php.

Configuração do Ficheiro .htaccess

O ficheiro .htaccess é utilizado para configurar regras específicas do servidor Apache, como redirecionamento e reescrita de URLs. No nosso caso, está configurado da seguinte forma:

```
1   DirectoryIndex index.php
2   RewriteEngine on
3   RewriteBase /aminhabiblioteca/
4
5   RewriteCond %{REQUEST_FILENAME} !-f
6   RewriteCond %{REQUEST_FILENAME} !-d
7   RewriteRule ^(.+)$ index.php?cmd=$1  [QSA,L]
```

Figura 55 – Códigodo ficheiro .htaccess

1) DirectoryIndex index.php: Define index.php como o ficheiro padrão a ser carregado quando um diretório é acedido.
2) RewriteEngine on: Habilita o módulo de reescrita de URLs do Apache.
3) RewriteBase /aminhabiblioteca/: Define a base para a reescrita de URLs, que é a subpasta da aplicação.
4) RewriteCond %{REQUEST_FILENAME} !-f e RewriteCond %{REQUEST_FILENAME} !-d: Estas condições verificam se o ficheiro ou diretório solicitado não existe fisicamente no servidor.
5) RewriteRule ^(.+)$ index.php?cmd=$1 [QSA,L]: Redireciona todas as solicitações que não correspondem a ficheiros ou diretórios existentes para index.php, passando a parte da URL como parâmetro cmd.

Estrutura do index.php

O index.php é o ponto de entrada da aplicação. Inclui o header, carrega o roteador e, por fim, o footer:

```php
<?php
include "views/header.php";
require_once "router.php";
include "views/footer.php";
?>
```

Figura 56 – Estrutura do ficheiro index.php

Lógica de Roteamento no router.php

O router.php é responsável por determinar qual view deve ser carregada com base no URL solicitado:

```php
<?php
$request = $_SERVER["REQUEST_URI"];

$base_path = "/aminhabiblioteca";
$rest = substr($request, strlen($base_path));

switch ($rest) {
    case "":
    case "/":
        require __DIR__ . "/views/home.php";
        break;
    default:
        if (strpos($rest, '/livro/') === 0) {

            $filename = __DIR__ . "/views/livro.php";
            if (file_exists($filename)) {
                require $filename;
                break;
            }
        }
        if (strpos($rest, '/autor/') === 0) {

            $filename = __DIR__ . "/views/autor.php";
            if (file_exists($filename)) {
                require $filename;
                break;
            }
        }
        $filename = __DIR__ . "/views" . $rest . ".php";
        if (file_exists($filename)) {
            require $filename;
            break;
        }
        http_response_code(404);
        require __DIR__ . "/views/404.php";
        break;
}
```

Figura 57 – Código do ficheiro router.php

1) $request = $_SERVER["REQUEST_URI"];: Obtém o URL solicitado.

2) $base_path = "/aminhabiblioteca"; : Define o caminho base da plataforma.
3) $rest = substr($request, strlen($base_path)); : Remove o caminho base do URL para obter o restante do URL.
4) switch ($rest): Utiliza um switch para determinar qual veiew deve ser carregada com base no restante da URL:
    a) case "" ou "/": Carrega a página inicial (home.php).
    b) if (strpos($rest, '/livro/') === 0): Verifica se o URL começa com /livro/ e carrega a view correspondente (livro.php).
    c) if (strpos($rest, '/autor/') === 0): Verifica se o URL começa com /autor/ e carrega a view correspondente (autor.php).
    d) $filename = __DIR__ . "/views" . $rest . ".php"; : Tenta carregar uma view com base no URL restante.
5) http_response_code(404);: Se nenhum dos casos anteriores corresponder, retorna um código de resposta 404 e carrega a página de erro (404.php).

Este processo garante que a plataforma lide corretamente com diferentes URLs e fornece sempre uma resposta ao utilizador, quer seja para carregar uma view existente ou, em caso de erro, uma view apropriada quando a página não é encontrada.

5.6. Desenvolvimento do Frontend

O frontend de uma aplicação é a camada responsável pela interação direta com o utilizador. É onde ocorre a apresentação de dados e funcionalidades de forma visual e acessível. No caso desta plataforma, o frontend foi desenvolvido para garantir uma experiência de uso eficiente e intuitiva. O objetivo principal é proporcionar uma interface agradável, onde os utilizadores possam navegar, realizar ações e interagir com os serviços oferecidos de forma fluida e prática.

O desenvolvimento do frontend envolve a utilização de tecnologias como HTML, CSS e JavaScript para construir as páginas e componentes da aplicação. Além disso, foram utilizados frameworks modernos e bibliotecas para otimizar a performance e facilitar o desenvolvimento de interfaces dinâmicas. Nesta secção, exploraremos as principais funcionalidades implementadas no frontend e a sua integração com o backend, assegurando a consistência e funcionalidade da aplicação como um todo.

Estrutura Básica de Todas as Páginas

No desenvolvimento do frontend da plataforma, a estrutura básica das páginas é padronizada para garantir consistência, modularidade e eficiência. Todas as páginas seguem um formato comum, utilizando um cabeçalho (header) e rodapé (footer) partilhados, com o conteúdo principal a ser carregado dinamicamente através de um roteador (router).

Esta estrutura simplificada permite uma fácil manutenção e expansão do código, além de otimizar o carregamento de recursos essenciais, como bibliotecas externas e folhas de estilo. A seguir, detalho a composição básica das páginas.

Estrutura Base – index.php

A estrutura básica de uma página na plataforma segue o seguinte modelo, exemplificado pelo ficheiro index.php:

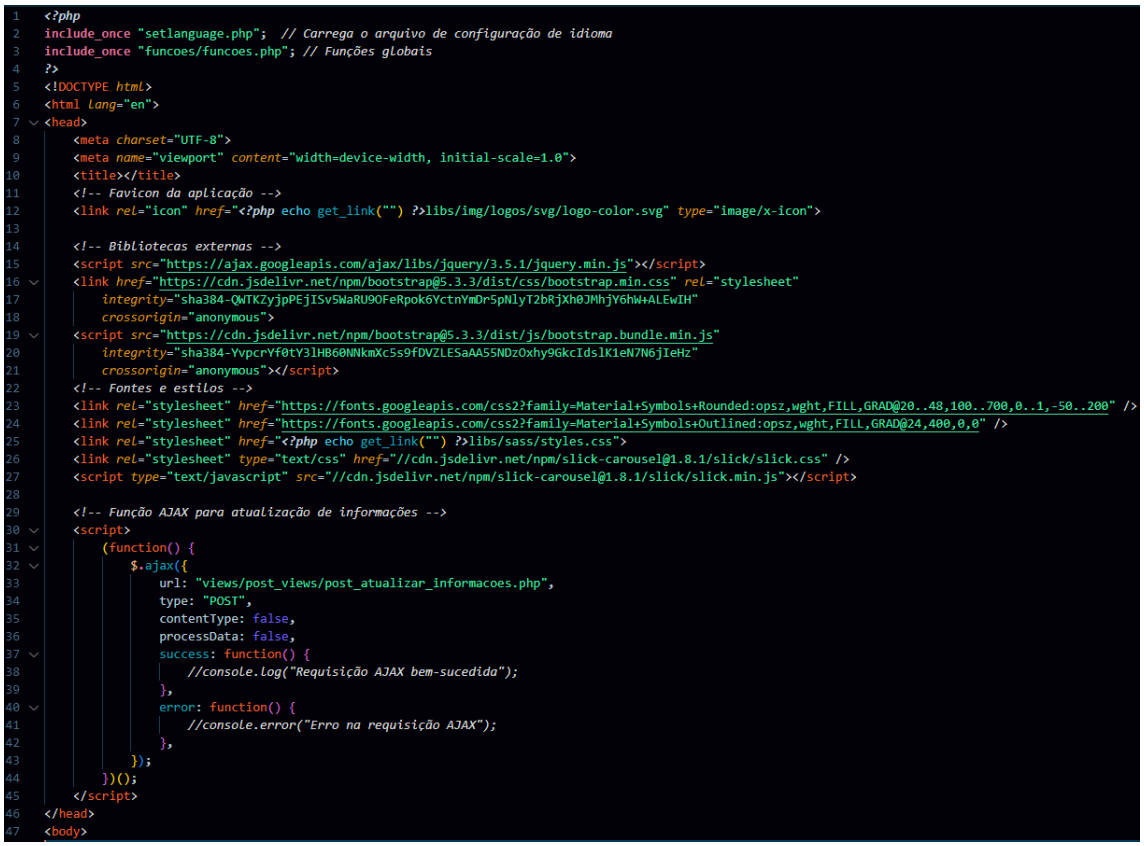
```php
<?php
include "views/header.php";
require_once "router.php";
include "views/footer.php";
?>
```

Figura 58 - Estrutura do ficheiro index.php

A lógica central das páginas reside no uso do roteador (router.php), que determina qual "view" será carregada de acordo com o link acedido pelo utilizador. Esta abordagem separa claramente a lógica do backend e frontend, mantendo o código limpo e organizado.

Header – header.php

Todas as páginas partilham o mesmo header, que inclui definições de idioma, funções globais e a inclusão de recursos essenciais como bibliotecas JavaScript e CSS:



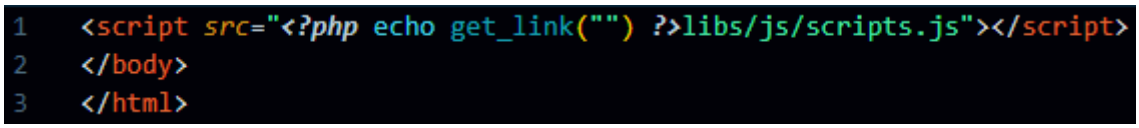
```php
<?php
include_once "setlanguage.php";  // Carrega o arquivo de configuração de idioma
include_once "funcoes/funcoes.php"; // Funções globais
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title></title>
    <!-- Favicon da aplicação -->
    <link rel="icon" href="<?php echo get_link("") ?>libs/img/logos/svg/logo-color.svg" type="image/x-icon">

    <!-- Bibliotecas externas -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
        crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
        crossorigin="anonymous"></script>
    <!-- Fontes e estilos -->
    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Rounded:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />
    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@24,400,0,0" />
    <link rel="stylesheet" href="<?php echo get_link("") ?>libs/sass/styles.css">
    <link rel="stylesheet" type="text/css" href="//cdn.jsdelivr.net/npm/slick-carousel@1.8.1/slick/slick.css" />
    <script type="text/javascript" src="//cdn.jsdelivr.net/npm/slick-carousel@1.8.1/slick/slick.min.js"></script>

    <!-- Função AJAX para atualização de informações -->
    <script>
        (function() {
            $.ajax({
                url: "views/post_views/post_atualizar_informacoes.php",
                type: "POST",
                contentType: false,
                processData: false,
                success: function() {
                    //console.log("Requisição AJAX bem-sucedida");
                },
                error: function() {
                    //console.error("Erro na requisição AJAX");
                },
            });
        })();
    </script>
</head>
<body>
```

Figura 59 – Código do ficheiro header .php

Este header garante que todos os elementos e recursos principais, como scripts e estilos, estejam disponíveis em todas as páginas da plataforma. A função get_link() facilita a geração de URLs dinâmicas.

Footer – footer.php

O footer das páginas inclui os scripts finais que são carregados para manter a interatividade da página:



```php
    <script src="<?php echo get_link("") ?>libs/js/scripts.js"></script>
    </body>
    </html>
```

Figura 60 – Código do ficheiro footer.php

Aqui, o ficheiro de JavaScript principal é incluído, garantindo que as funcionalidades necessárias sejam carregadas em todas as páginas.

Router Dinâmico

O roteador (router.php) é responsável por carregar a "view" correspondente com base na URL acedida. Isto significa que, dependendo da rota, diferentes ficheiros da pasta views/ serão incluídos para exibir o conteúdo correto.

Tipos de Views na Plataforma

No frontend da plataforma, além do header e footer, existem diferentes tipos de views que desempenham funções específicas para garantir a apresentação do conteúdo e a interação do utilizador com a aplicação. A seguir, detalho os principais tipos de views e as suas funções.

Views Normais

As views normais são as páginas principais da plataforma. Elas representam o conteúdo visível e acessível diretamente pelo utilizador, como as páginas de pesquisa, de perfil, de livros, etc. Cada uma dessas páginas é carregada pelo roteador (router.php) e exibe os dados de acordo com a rota acedida.

Estas views são responsáveis por exibir o conteúdo estático e dinâmico da plataforma, integrando com a camada de controlo para apresentar informações em tempo real, como resultados de pesquisa e detalhes de livros.

Views do Menu

A plataforma conta com uma barra lateral (sidebar) dinâmica, que exibe diferentes categorias de links dependendo do tipo de utilizador (utilizador comum, administrador, staff). A estrutura do menu é dividida em ficheiros distintos para manter a organização e facilitar a manutenção.

- menu.php: Este ficheiro contém a estrutura principal da barra lateral da plataforma. Ele é responsável por montar a barra lateral e integrar os diferentes grupos de links. A lógica de exibição de links depende do nível de acesso do utilizador.
- -link-base.php: Contém os links básicos acessíveis a todos os utilizadores da plataforma. Exibe links para funcionalidades comuns, como pesquisa e perfil.
- -link-contas.php: Lista links relacionados à gestão de contas de utilizadores, como opções de login, e visualização de detalhes da conta.

- links-staff.php: Exibe links especiais para utilizadores do tipo "staff", com permissões para aceder a funcionalidades adicionais, como gestão de livros e reservas.
- links-admin.php: Inclui os links exclusivos para administradores, que possuem acesso total à plataforma, permitindo a gestão de utilizadores, administração do sistema, e outras funcionalidades avançadas.
- menu_links.php: Este ficheiro unifica os quatro ficheiros de links anteriores, montando o menu completo com todas as opções categorizadas. Ele verifica o tipo de utilizador e exibe os links corretos com base nas permissões de cada um.

Views de Post (POST Views)

A pasta post_views contém todas as views que são executadas por requisições AJAX feitas pelo jQuery. Estas views não são diretamente acedidas pelo utilizador, mas sim carregadas em segundo plano para realizar operações específicas e interagir com a camada de controlo da aplicação. Algumas operações comuns feitas por estas views incluem:

- Pesquisar livros: A view realiza uma consulta à base de dados, retornando os livros correspondentes aos critérios de pesquisa.
- Atualizar perfil: Permite que os utilizadores modifiquem as suas informações de perfil sem recarregar a página.
- Requisitar livro: Processa a solicitação de requisição de um livro por um utilizador e interage com a camada de controlo para verificar disponibilidade e registar a requisição.

Estas views de post são essenciais para manter a interatividade da plataforma, permitindo que ações sejam realizadas de forma dinâmica e sem a necessidade de recarregar a página. Elas tornam a experiência do utilizador mais fluida e eficiente, integrando diretamente o frontend e backend da aplicação.

Menu

A imagem apresentada representa o menu de navegação de um utilizador normal na plataforma. No topo da sidebar, encontra-se o logotipo da plataforma. Logo abaixo, são exibidos os links básicos, seguidos pelos links relacionados à conta do utilizador. Ao final da sidebar, encontram-se os botões que permitem mudar o idioma da plataforma e efetuar o logout.

Figura 61- Menu da plataforma

Links do menu (menu_links.php)

```php
<?php
include_once "links-base.php";
include_once "links-contas.php";
include_once "links-staff.php";
include_once "links-admin.php";
?>
```

Figura 62 – Código do ficheiro menu_links.php

Este trecho de código PHP é responsável por incluir diferentes conjuntos de links na barra lateral da plataforma. Cada ficheiro incluído representa um grupo de links específico que será exibido no menu de navegação. Abaixo está uma explicação de cada um:

- links-base.php: Contém os links básicos acessíveis a todos os utilizadores, como a página inicial, pesquisa, ou outras funcionalidades comuns.
- links-contas.php: Inclui links relacionados à conta do utilizador, como o perfil, histórico de atividades ou definições de conta.
- links-staff.php: Contém links exclusivos para membros do staff, oferecendo acesso a ferramentas e funcionalidades de administração de conteúdos ou gestão da plataforma.
- links-admin.php: Inclui links específicos para administradores, permitindo o acesso a funcionalidades avançadas de gestão e controlo da plataforma.

Este código junta todos esses ficheiros para gerar a lista completa de links que será apresentada ao utilizador, dependendo do seu nível de permissão.

Estrutura dos ficheiros links

O ficheiro links-admin.php segue uma estrutura básica que define um array de links, onde cada link é representado por um conjunto de atributos. Esta estrutura é comum a todos os ficheiros de links na plataforma (links-base.php, links-contas.php, links-staff.php), e cada um contém links específicos de acordo com a funcionalidade a que pertencem.

Estrutura básica de cada link:

- icon: Define o ícone associado ao link, que será exibido ao lado do rótulo no menu.
- label: Representa o texto que será exibido no menu, traduzido ou personalizado, como por exemplo, $procont, que contém o nome do link.
- url: Define o URL para onde o utilizador será redirecionado ao clicar no link. Este URL é gerado utilizando a função get_link(), que assegura que o caminho seja dinâmico.
- related: Um array que pode conter URLs relacionados ao link principal.

Exemplo com base em links-admin.php:

```php
<?php
$links_admin = [
    [
        "icon" => "person_search",
        "label" => $procont,
        "url" => get_link("procurarconta"),
        "related" => []
    ],
    [
        "icon" => "auto_delete",
        "label" => $listdel,
        "url" => get_link("listapagar"),
        "related" => []
    ],
    [
        "icon" => "help_center",
        "label" => $gertick,
        "url" => get_link("gerirtickets"),
        "related" => [get_link("detalhesticketstaff")]
    ]
];
?>
```

Figura 63 – Código exemplo com base em link-admin.php

Este padrão é repetido nos outros ficheiros de links, alterando apenas os valores do ícone, rótulo e URL de acordo com a categoria de links específica que cada ficheiro trata.

Exemplo de geração dos botões na sidebar

Para gerar a interface que exibe os links administrativos (links_admin) na barra lateral da plataforma, apenas para utilizadores que possuem o tipo de utilizador 1. Esta interface dinamicamente exibe ou oculta os ícones de adicionar ou remover, dependendo do tipo de utilizador, e cria links navegáveis baseados no array $links_admin.

```php
<?php
if ($user_type == 1) {
    $showadminpages = ($user_type == 1) ? "sidebar-modal-open" : "";
    $showsadminiconadd = ($user_type == 1) ? "iconhidden" : "";
    $showadminiconrem = ($user_type == 1) ? "" : "iconhidden";
?>
    <div>
        <div class="d-flex justify-content-center align-items-center">
            <span class="pages-span-text"><?php echo $adminpages ?></span>
            <button class="pages-span-button" type="button" id="adminpages-show">
                <span class="material-symbols-rounded <?php echo $showsadminiconadd ?>" id="iconadd-admin-show">
                    add
                </span>
                <span class="material-symbols-rounded <?php echo $showadminiconrem ?>" id="iconremove-admin-show">
                    remove
                </span>
            </button>
        </div>

        <div class="sidebar-modal <?php echo $showadminpages ?>" id="modal-adminpages">
            <?php
            foreach ($links_admin as $link) {
                $active = ($request == $link['url'] || (is_array($link['related']) &&
                    array_reduce($link['related'], function ($carry, $item) use ($request) {
                        return $carry || strpos($request, $item) !== false;
                    }, false))) ? "active" : "";
                echo '<a class="nav-link my-1 ' . $active . '" href="' . $link['url'] . '">
    <div class="bordinhaazul">
        <img src="' . get_link("") . 'libs/img/Rectangle_17.png" alt="Rectangle_17">
    </div>
    <div class="nav-links-label">
        <span class="material-symbols-rounded sidebaricon icon-30">
            ' . $link['icon'] . '
        </span>
        <span class="ms-2 span-links">' . $link['label'] . '</span>
    </div>
</a>';
            }
            ?>
        </div>
    </div>
    <?php
    }
    ?>
```

Figura 64 – Código exemplo de geração dos botões na sidebar

Explicação detalhada:

1) Condicional de Tipo de Utilizador ($user_type == 1):
   a) O código só é executado se o tipo de utilizador for 1, o que indica que apenas administradores podem visualizar esses links.
2) Controle de Visualização ($showadminpages, $showsadminiconadd, $showadminiconrem):
   a) Estas variáveis controlam a aparência dos ícones e a visibilidade dos links administrativos.
   b) O ícone de "adicionar" (add) será mostrado inicialmente, e o de "remover" (remove) será exibido quando os links estiverem visíveis.
3) Estrutura de Interface:
   a) Um botão é colocado ao lado do título "Páginas Admin" ($adminpages), que alterna a visualização dos links administrativos.

b) O ícone de “adicionar” ou “remover” será trocado de acordo com a ação (expandir ou recolher a lista de links).

4) Impressão dos Links Administrativos:
   a) Dentro do bloco foreach, cada item no array $links_admin é processado para gerar um link (<a>), que contém:
      i) Um ícone (material-symbols-rounded) associado ao link.
      ii) Um rótulo/texto do link.
   b) O link também verifica se está ativo (a página atual coincide com o URL do link ou com os URLs relacionados), e aplica a classe active para destacar o link atual.
5) Classes Dinâmicas:
   a) Dependendo do estado do modal e da interação do utilizador, as classes sidebar-modal-open, iconhidden, e active são aplicadas dinamicamente.

Funcionalidades Chave:

Controle de Acessibilidade: Os links administrativos só são exibidos para utilizadores de tipo 1 (administradores).

Visibilidade Dinâmica: Utilizando botões e ícones, a barra lateral pode ser expandida ou recolhida para exibir ou ocultar os links administrativos.

Links Dinâmicos: Cada link administrativo é gerado a partir de um array e verifica se está ativo, com base na URL atual.

Variações do menu de links

O menu da plataforma apresenta três variações distintas, adaptando-se ao tipo de utilizador que está autenticado no sistema.

A variação do menu para um utilizador normal (user) é projetada para fornecer acesso às funcionalidades e secções básicas da plataforma, de acordo com as permissões padrão.

Figura 65 – Variação do menu para um utilizador normal

A variação do menu para um utilizador funcionário (staff) é ajustada para oferecer acesso às ferramentas e secções necessárias para desempenhar funções específicas e administrativas dentro da plataforma.

Figura 66 – Variação do menu para um utilizador funcionário

A variação do menu para um utilizador administrador (admin) inclui opções avançadas e de gestão, permitindo o controlo total sobre as funcionalidades e configurações da plataforma.

Figura 67 – Variação do menu para um utilizador administrador

Estrutura das páginas

As páginas da plataforma estão divididas em duas secções principais:

- Menu Lateral: Localizado à esquerda, o menu fornece navegação entre diferentes secções e funcionalidades da plataforma, variando conforme o tipo de utilizador (normal, funcionário ou administrador).
- Conteúdo Principal: Situado à direita, o conteúdo é subdividido em duas áreas:
    - Secção Superior: Inclui um marcador de página que muda conforme a página que está a ser visualizada. À esquerda, aparece o marcador de página, e à direita, encontram-se o botão para abrir as notificações e a imagem de perfil do utilizador.
    - Secção Inferior: Exibe o conteúdo principal da página, apresentando as informações e funcionalidades relevantes para o utilizador.

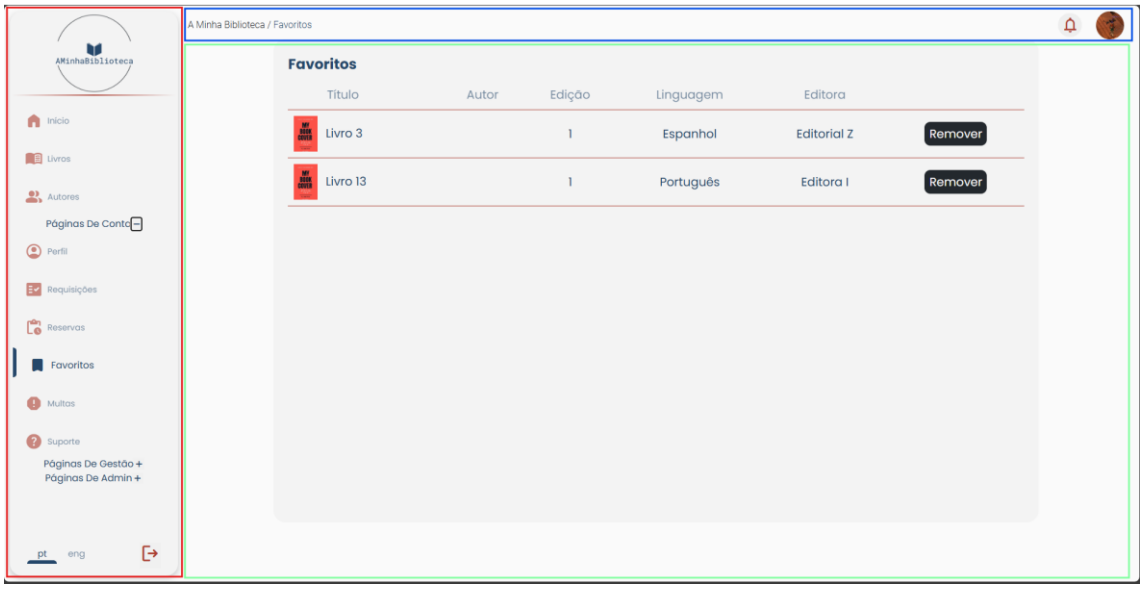


Figura 68 - Estrutura das páginas

De acordo com a imagem apresentada, a secção do menu está rodeada a vermelho, a secção do marcador de página, o botão das notificações e a imagem de perfil estão rodeados a azul, e a secção do conteúdo da página está rodeada a verde.

Exemplo do sistema de grid das páginas

O conteúdo da página está organizado numa estrutura de grelha, utilizando a propriedade display: grid do CSS. Esta abordagem permite a criação de diversos estilos de página na plataforma, desde uma grelha simples, como ilustrado na imagem anterior, até layouts mais complexos com várias linhas e colunas. Além disso, é possível implementar grelhas dentro de grelhas, proporcionando flexibilidade e versatilidade no design das páginas.

Dando o exemplo da próxima imagem, a área rodeada a laranja representa a grelha global, enquanto cada uma das outras cores indica diferentes grelhas dentro da grelha global.

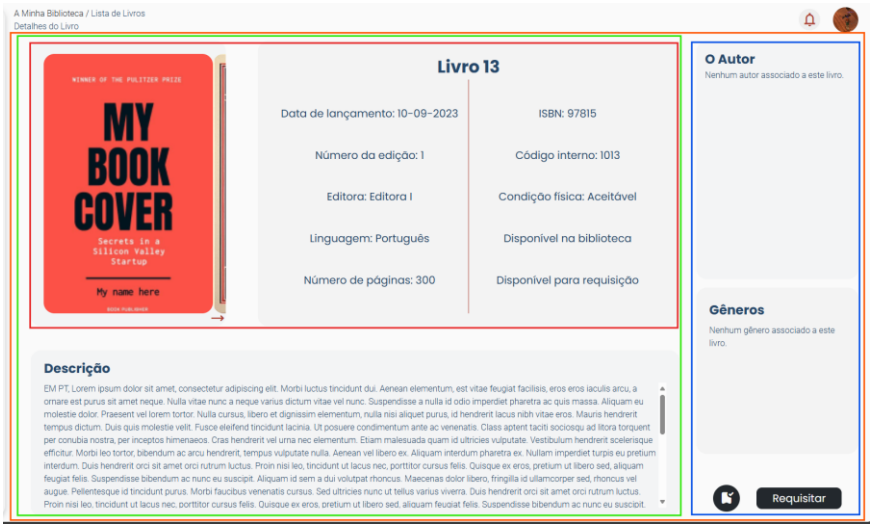


Figura 69 - Exemplo do sistema de grid das páginas

Gestão de Alertas

Em diversas viewa da plataforma, é utilizada a seguinte estrutura para a exibição de alertas:

```
<div id="alert-container" class="fade show position-absolute mt-4 start-50 translate-middle-x top-1 w-auto">
</div> <!-- Container para os alerts -->
```

Figura 70 - Gestão de alertas

Esta div, identificada pelo id="alert-container", serve como o container para os alertas. A sua classe CSS inclui fade e show para controlar a animação de entrada e saída dos alertas, enquanto position-absolute e outras classes de posicionamento garantem que os alertas apareçam na posição correta na página.

O conteúdo e a visibilidade desta div são geridos através de jQuery. A div é “ativada” e atualizada com o conteúdo dos alertas quando o jQuery a modifica, permitindo que os alertas sejam exibidos dinamicamente com base nas ações ou eventos na plataforma.

Exemplo de ligação à camada controlo

O código PHP abaixo é responsável por carregar a lista de livros favoritos de um utilizador numa página específica da plataforma:

```php
<?php
include_once "verifica-sessao.php";
$link = 'controlo/user-controlo.php';
require_once $link;
$controlo = new user_controlo();
$listar_favoritos = $controlo->listar_favoritos($_SESSION['user_dados']['id']);
?>
```

Figura 71 - Código exemplo de ligação à camada de controlo

Inclusão de verifica-sessao.php:

```php
include_once "verifica-sessao.php";
```

Figura 72 – Código de inclusão de verifica-sessao.php

Este ficheiro é incluído para garantir que a sessão do utilizador está ativa e válida. Normalmente, contém a lógica para verificar a autenticidade do utilizador e assegurar que o acesso à página é permitido apenas para utilizadores autenticados.

Definição do Caminho para o Controlador:

```php
$link = 'controlo/user-controlo.php';
```

Figura 73 – Código de definição do caminho para o controlador

Define o caminho para o ficheiro do controlador (user-controlo.php), que contém a lógica para manipulação de dados do utilizador. A variável $link é utilizada para facilitar a inclusão do controlador.

Inclusão do Controlador:

```php
require_once $link;
```

Figura 74 – Código de inclusão do controlador

Inclui o ficheiro do controlador (user-controlo.php). A utilização de require_once assegura que o ficheiro é incluído apenas uma vez, prevenindo erros e duplicações.

Criação da Instância do Controlador:

```
5   $controlo = new user_controlo();
```

Figura 75 – Código de criação da instância do controlador

Cria uma instância nova da classe user_controlo, que é responsável pela gestão das operações relacionadas aos dados do utilizador.

Listagem dos Livros Favoritos:

```
6   $listar_favoritos = $controlo->listar_favoritos($_SESSION['user_dados']['id']);
```

Figura 76 – Código da listagem dos favoritos

Chama o método listar_favoritos da instância user_controlo, passando o ID do utilizador (obtido a partir da sessão) como parâmetro. Este método retorna a lista de livros que o utilizador marcou como favoritos e armazena o resultado na variável $listar_favoritos.

Em resumo, este código é responsável por verificar a sessão do utilizador, incluir o controlador necessário, criar uma instância do controlador, e finalmente recuperar a lista de livros favoritos do utilizador para exibição na página.

Post_views

As post_views são componentes fundamentais do frontend que, através de requisições AJAX, permitem a comunicação eficiente com o backend, possibilitando a atualização dinâmica de dados na interface sem a necessidade de recarregar a página.

Exemplo de uma Post View

Neste exemplo, é explicado como uma post view funciona para atualizar o perfil de um utilizador na plataforma, sendo chamada por um script em jQuery e processada no backend através de uma requisição POST.

Requisição via jQuery:

- O botão com o ID #perfil_salvar dispara um evento click. Quando clicado, ele coleta os valores de diversos campos do formulário de perfil, como nome, sobrenome, e-mail, telefone, endereço, código postal e a URL da foto de perfil.

- O valor da imagem de fundo é extraído do elemento CSS e transformado numa URL válida para enviar via AJAX.

- O script envia os dados coletados para o ficheiro post-atualizar-perfil.php usando uma requisição AJAX do tipo POST. Isto é feito para que a página não seja recarregada, tornando a atualização do perfil mais dinâmica.

```
$("#perfil_salvar").click(function () {
  var prinome = $("#perfil-prinome").val();
  var ultnome = $("#perfil-ultnome").val();
  var email = $("#perfil-email").val();
  var oldemail = $("#perfil-email").attr("name");
  var numero = $("#perfil-numero").val();
  var morada = $("#perfil-morada").val();
  var codigopostal = $("#perfil-codigopostal").val();

  var imagem_fundo = $("#imagem-perfil").css("background-image");
  var caminho = imagem_fundo.match(/^url\(["']?(.+?)["']?\)$/);

  var photo_url = caminho[1]; // Obtém o caminho da imagem
  //alert('Caminho da Imagem de Fundo: ' + photo_url);
  $.ajax({
    type: "POST",
    url: "views/post_views/post-atualizar-perfil.php",
    data: {
      prinome: prinome,
      ultnome: ultnome,
      email: email,
      oldemail: oldemail,
      numero: numero,
      morada: morada,
      codigopostal: codigopostal,
      photo_url: photo_url,
    },
    dataType: "html",
    success: function (response) {
      $("#alert-container").html(response);
    },
  });
});
```

Figura 77 – Código exemplo em JQuery de uma função click

Ficheiro post-atualizar-perfil.php:

- O ficheiro PHP post-atualizar-perfil.php recebe a requisição enviada via POST. Ele inclui ficheiros essenciais de controlo e funções, como user-controlo.php, responsável pela lógica de atualização de utilizadores.
- O código verifica se os dados enviados estão definidos e, em seguida, usa o objeto $controlo para atualizar as informações do utilizador na base de dados através do método update_user.
- Dependendo do sucesso ou falha da operação de atualização, uma mensagem de alerta é retornada e exibida dinamicamente na página.

```php
<?php
$rootDirectory = dirname(dirname(__DIR__));
$link = $rootDirectory . '/controlo/user-controlo.php';
$link2 = $rootDirectory . '/funcoes/funcoes.php';
require_once $link;
include_once $link2;
$controlo = new user_controlo();
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}
$user_id = $_SESSION['user_dados']['id'];
if (isset($_POST['prinome']) && isset($_POST['ultnome']) && isset($_POST['email']) && isset($_POST['oldemail'])
&& isset($_POST['numero']) && isset($_POST['morada']) && isset($_POST['codigopostal']) && isset($_POST['photo_url'])) {
    $first_name = $_POST['prinome'];
    $last_name = $_POST['ultnome'];
    $email = $_POST['email'];
    $oldemail = $_POST['oldemail'];
    $number = $_POST['numero'];
    $address = $_POST['morada'];
    $postal_code = $_POST['codigopostal'];
    $photo_url = $_POST['photo_url'];

    $atualizar = $controlo->update_user($user_id, $first_name, $last_name, $email, $oldemail, $photo_url, $address, $postal_code, $number);

    if ($atualizar["success"]) {
        $_SESSION['user_dados']['photo_url'] = basename($photo_url);;
        echo '
        <div class="alert alert-success alert-dismissible" role="alert">
            ' . $atualizar["data"] . '
            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close" onclick="location.reload();"></button>
        </div>
        ';
    } else {
        echo '
        <div class="alert alert-warning alert-dismissible" role="alert">
            ' . $atualizar["data"] . '
            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
        </div>
        ';
    }
    exit;
}
```

Figura 78 – Código do ficheiro post-atualizar-perfil.php

Explicação do Fluxo:

- O utilizador clica no botão de salvar, e os dados do perfil são enviados via AJAX para o backend sem recarregar a página.
- O backend processa os dados, atualiza as informações do utilizador na base de dados e devolve uma mensagem de sucesso ou erro.
- A resposta é exibida no container de alertas na página, permitindo que o utilizador veja o estado da operação em tempo real.

Função das Post Views

As post views são chamadas via AJAX quando ações específicas são realizadas (como atualizar o perfil) e são responsáveis por conectar a interface com o backend, processando dados e devolvendo o resultado de forma dinâmica.

5.7. Autenticação de utilizadores

A autenticação de utilizadores é essencial para plataformas que lidam com dados sensíveis, como uma biblioteca digital. Este projeto inclui dois processos principais: registo de novos utilizadores e login para utilizadores existentes.

No registo, os novos utilizadores criam contas fornecendo informações como username, e-mail e palavra-passe, que são verificadas e armazenadas de forma segura. No login, utilizadores inserem o nome de utilizador e a palavra-passe para aceder na plataforma. A segurança é garantida através da encriptação da palavra-passe e validação de credenciais, prevenindo acessos não autorizados.

Registo

No processo de registo na plataforma, após o utilizador preencher o formulário com os dados necessários (e-mail, nome de utilizador, palavra-passe, confirmação da palavra-passe) e aceitar os termos e condições, o jQuery é responsável por submeter as informações ao servidor. Isto é feito através de uma requisição POST para o ficheiro post_registo.php.

Este ficheiro PHP faz a ligação com a camada de controlo, chamando o método registo() para efetuar o registo. O método registo() valida os dados fornecidos e segue os seguintes passos:

1. **Validação de Campos**: Verifica se todos os campos foram preenchidos corretamente.

2. **Verificação da Palavra-passe**: Confirma se a palavra-passe e a confirmação da palavra-passe coincidem.

3. **Verificação de Utilizador Existente**: Usa o método verificar_user() para garantir que o nome de utilizador e o e-mail não estão já registados no sistema.

4. **Encriptação da Palavra-passe**: Se todos os dados estiverem corretos, a palavra-passe é encriptada utilizando password_hash() com o algoritmo PASSWORD_BCRYPT, garantindo segurança adicional.

5. **Registo na Base de Dados**: Com a palavra-passe encriptada, a função control->registo() insere os dados do utilizador na base de dados, juntamente com a data de criação da conta.

6. **Confirmação**: Se o registo for bem-sucedido, o utilizador recebe uma mensagem de sucesso. Caso contrário, uma mensagem de erro será exibida.

Se o registo for bem-sucedido, o jQuery encarrega-se de fornecer um feedback imediato ao utilizador, atualizando dinamicamente a interface. Após o processamento da requisição POST no post_registo.php, uma resposta é enviada ao frontend. O jQuery recebe essa resposta e carrega um conteúdo HTML diretamente no elemento , que é o contêiner preparado para exibir alertas.

Neste caso, o alerta exibirá uma mensagem a indicar que o registo foi efetuado com sucesso, garantindo ao utilizador que a operação foi realizada corretamente.

Login

No processo de login, o código PHP contido em post_login.php é responsável por tratar os dados enviados pelo utilizador, como o nome de utilizador e a palavra-passe. Abaixo está uma explicação detalhada de como funciona o processo de login:

Receção de Dados:

O ficheiro recebe os dados do formulário de login através de $_POST, que são:

- username: nome de utilizador inserido.
- password: palavra-passe inserida.
- manter: opção para manter a sessão ativa (caso o utilizador selecione a opção de “lembrar-me”).

Verificação das Credenciais:

A função login($username, $password) do objeto $controlo é chamada para validar as credenciais:

```
$login = $controlo->login($username, $password);
```

Figura 79 – Código de verificação de credenciais

Verificação da Palavra-passe: A função obter_hash_senha($username) busca o hash da palavra-passe armazenado na base de dados para aquele nome de utilizador. O método password_verify($password, $senha_hash) é utilizado para comparar a palavra-passe inserida pelo utilizador com o hash da palavra-passe armazenada.

Verificação das Credenciais: Se a palavra-passe for válida, o sistema chama verificar_credenciais($username) para recuperar as informações do utilizador (como o ID, o tipo de utilizador, o status e a foto de perfil) e armazená-las numa variável.

Atualização de Informações: A função atualizar_apagar_user($username) atualiza certos dados do utilizador na base de dados, como o último login.

Se o login for bem-sucedido, as informações do utilizador são retornadas como um array associativo, indicando sucesso.

Configuração da Sessão e Cookies:

Se o login for bem-sucedido:

```
$_SESSION['user_dados'] = $login['data'];
setcookie("user_data", json_encode($login['data']), time() + (30 * 24 * 60 * 60), "/");
```

Figura 80 – Código de criação das variáveis de credenciais

O sistema inicia uma sessão com session_start() e armazena as informações do utilizador na variável de sessão $_SESSION['user_dados'].

O cookie user_data também é criado para armazenar essas informações. Este cookie é configurado para expirar em 30 dias.

Se o utilizador tiver selecionado a opção de “manter sessão ativa”, outro cookie (manter_sessao) é criado com a validade de 30 dias.

Redireccionamento para a Página Principal:

```
    echo '
    <a id="invisible-link" href="javascript:void(0);" style="display: none;"></a>
    <script>
    $(document).ready(function() {
        $("#invisible-link").trigger("click");
    });

    $("#invisible-link").on("click", function() {
        window.location.href = "' . get_link("") . '";
    });
</script>
';
    exit;
```

Figura 81 – Código de redireccionamento para a página principal

Esse código cria um link invisível que, ao ser acionado, redireciona o utilizador para a página principal da plataforma. Este redireccionamento acontece automaticamente ao carregar a resposta do servidor.

Caso o Login Falhe:

Se as credenciais não forem válidas um alerta de erro é exibido na página com uma mensagem apropriada, informando que o login falhou.

Função login($username, $password) no Controlo:

```php
function login($username, $password)
{
    global $login_falhado;
    $senha_hash = $this->control->obter_hash_senha($username);
    if ($senha_hash && password_verify($password, $senha_hash)) {
        $login = $this->control->verificar_credenciais($username);
        if ($login instanceof mysqli_result) {
            if ($login->num_rows == 1) {
                $linha = $login->fetch_assoc();
                $this->control->atualizar_apagar_user($username);
                $user_info = [
                    "id" => $linha['id'],
                    "type" => $linha['user_type_id'],
                    "status" => $linha['status'],
                    "photo_url" => $linha['photo_url'],
                    "username" => $username
                ];
                return [
                    'success' => true,
                    'data' => $user_info
                ];
            } else {
                // Credenciais inválidas
                return [
                    'success' => false,
                    'data' => $login_falhado
                ];
            }
        } else {
            // Tratar erro de consulta
            return [
                'success' => false,
                'data' => $login // $login contém o erro
            ];
        }
    } else {
        return [
            'success' => false,
            'data' => $login_falhado
        ];
    }
}
```

Figura 82 – Código da função login

A função login() faz a maior parte do trabalho ao verificar se a palavra-passe corresponde ao hash armazenado e recupera as informações necessárias do utilizador:

Esta função verifica a palavra-passe e retorna um array indicando sucesso ou falha, juntamente com as informações do utilizador em caso de sucesso.

Dessa forma, o processo de login é concluído, e o utilizador é autenticado com feedback imediato sobre o estado da operação.

Verificação de sessão

Após o login, o utilizador é redirecionado para a plataforma, onde pode navegar por várias páginas da aplicação. Para garantir que o utilizador está autenticado em todas as páginas e que a sessão permanece válida, a plataforma utiliza o ficheiro verifica-sessao.php, incluído em todas as views.

O ficheiro verifica-sessao.php realiza as seguintes funções:

1. **Verificação da Sessão ou Cookies:**

O sistema verifica se o cookie manter_sessao está ativo ou se a sessão ainda está ativa. O cookie manter_sessao pode ter dois valores:

- 1: O utilizador optou por manter a sessão ativa.
- 0: O utilizador não quer manter a sessão, mas a sessão está ativa enquanto o navegador estiver aberto (controlado pelo cookie sessao_ativa).

Caso nenhum dos cookies seja encontrado, o utilizador é redirecionado para a página de logout.

```
if ((isset($_COOKIE['manter_sessao']) && $_COOKIE['manter_sessao'] == 1) ||
    (isset($_COOKIE['manter_sessao']) && $_COOKIE['manter_sessao'] == 0 && isset($_COOKIE['sessao_ativa']))) {
```

Figura 83 – Código de verificação das variáveis de credenciais

2. **Inicialização da Sessão:**

Se a sessão não estiver ativa, a função session_start() é chamada para iniciar a sessão. A sessão armazena informações críticas do utilizador, como o tipo de utilizador e outros dados necessários para o funcionamento correto da plataforma.

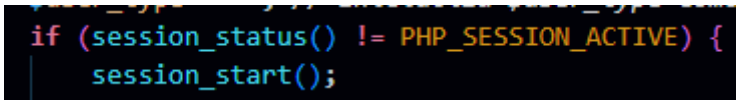


Figura 84 – Código de início de sessão

3. **Verificação dos Dados do Utilizador:**

O ficheiro verifica se a variável $_SESSION['user_dados'] está definida. Caso esteja, as informações do utilizador são utilizadas diretamente. Se não estiver definida, o sistema tenta recuperar os dados através do cookie user_data (onde as informações do utilizador estão armazenadas). Após isso, as informações do utilizador são atualizadas por meio do ficheiro post_atualizar_informacoes.php.

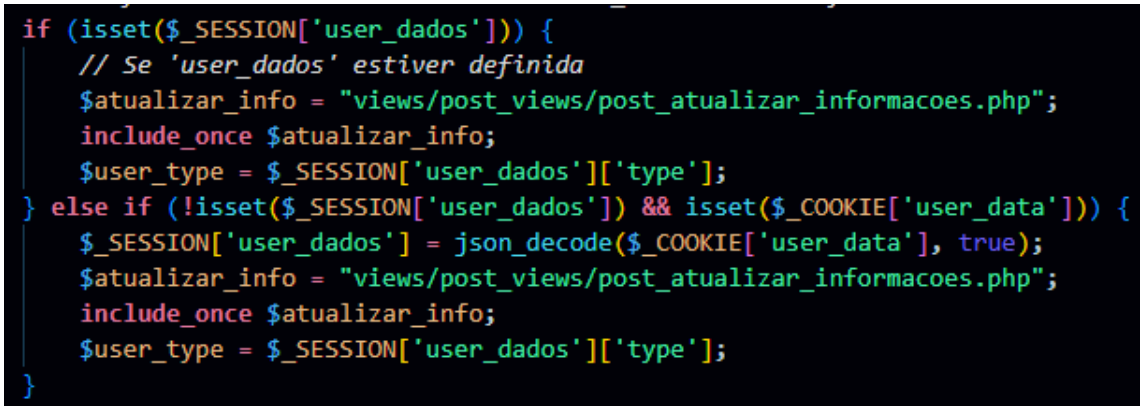


Figura 85 – Código de verificação dos dados de utilizador

4. **Redireccionamento para o Logout:**

Se o sistema não encontrar as informações da sessão ou do cookie, o utilizador é redirecionado para o logout.

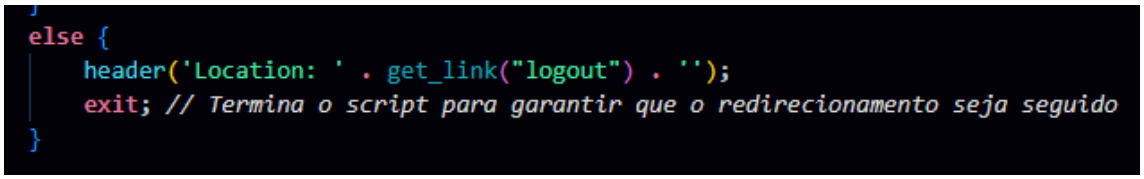


Figura 86 – Código de redireccionamento para o logout

5. **Sessão Inativa:**

Se nenhum dos cookies (manter_sessao ou sessao_ativa) estiver ativo, o utilizador é redirecionado para a página de logout para garantir a segurança e a integridade da sessão.

```
} else {
    header('Location: ' . get_link("logout") . '');
    exit;
}
```

Figura 87– Código de redireccionamento para o logout

Em resumo, o verifica-sessao.php garante que, em todas as páginas da plataforma, o utilizador está devidamente autenticado. Caso a sessão expire ou os cookies não estejam ativos, o sistema redireciona automaticamente o utilizador para a página de login ou logout, mantendo o processo de autenticação seguro.

6. Base de dados

6.1. Views

Na base de dados foram criadas duas views essenciais para melhorar a eficiência das consultas:

View: top_popular_books

Esta view lista os três livros mais populares, baseando-se no número de requisições feitas. Ela faz uma junção entre as tabelas books e requests para contar o número de pedidos de cada livro, excluindo os que foram removidos (deleted = 0) e os que ainda estão pendentes de devolução (status = 0). O resultado é ordenado por quantidade de pedidos em ordem decrescente, retornando apenas os três livros com mais solicitações.

View: top_popular_categories

Esta view apresenta as quatro categorias de livros mais populares, com base no número de requisições realizadas. Ela faz uma junção entre as tabelas genres, book_genres, requests, e duas tabelas de tradução (translation_table) para exibir os nomes das categorias em português e inglês. A consulta conta o número de requisições para cada género e organiza os resultados por popularidade, retornando as quatro categorias com mais pedidos.

Estas views permitem otimizar consultas e exibir dados frequentemente solicitados sem a necessidade de recalcular as estatísticas a cada vez que são requisitadas.

6.2.Eventos

As interações entre a base de dados e os utilizadores, embora fundamentais, não são suficientes para garantir que a plataforma e a base de dados permaneçam 100% dinâmicas e atualizadas automaticamente. Existem situações em que ações ou verificações precisam ser realizadas sem a intervenção direta dos utilizadores, como a remoção de registos antigos ou a verificação de prazos de requisição.

Para esses casos, o MySQL oferece uma poderosa ferramenta chamada eventos. Os eventos são rotinas automáticas que permitem a execução de comandos SQL em horários específicos ou intervalos de tempo definidos, diretamente no servidor de base de dados, sem que um utilizador ou aplicação precise disparar essas ações.

Por exemplo, pode-se configurar um evento para correr diariamente e verificar quais livros estão com prazo de entrega expirado, aplicando automaticamente as devidas penalidades aos utilizadores em questão. Assim, os eventos ajudam a manter a base de dados e a plataforma sempre atualizadas e em conformidade com as regras definidas, sem depender da intervenção manual.

Esses eventos são configurados no phpMyAdmin ou diretamente no servidor de base de dados, garantindo que o sistema seja mais eficiente e reduzindo a necessidade de processos manuais por parte de administradores ou utilizadores.

Para garantir que os eventos agendados na base de dados sejam executados corretamente, é fundamental que o **agendador de eventos** do MySQL esteja ativado. O agendador de eventos é um serviço interno do MySQL que controla a execução automática de eventos pré-definidos, permitindo que operações sejam realizadas periodicamente ou em horários específicos sem a necessidade de intervenção manual.

Sem o agendador de eventos ativo, mesmo que os eventos estejam corretamente configurados, eles não serão executados. Para verificar e ativar o agendador de eventos, o seguinte comando SQL pode ser utilizado:

```
SET GLOBAL event_scheduler = ON;
```

Ou na página de eventos do phpMyAdmin:

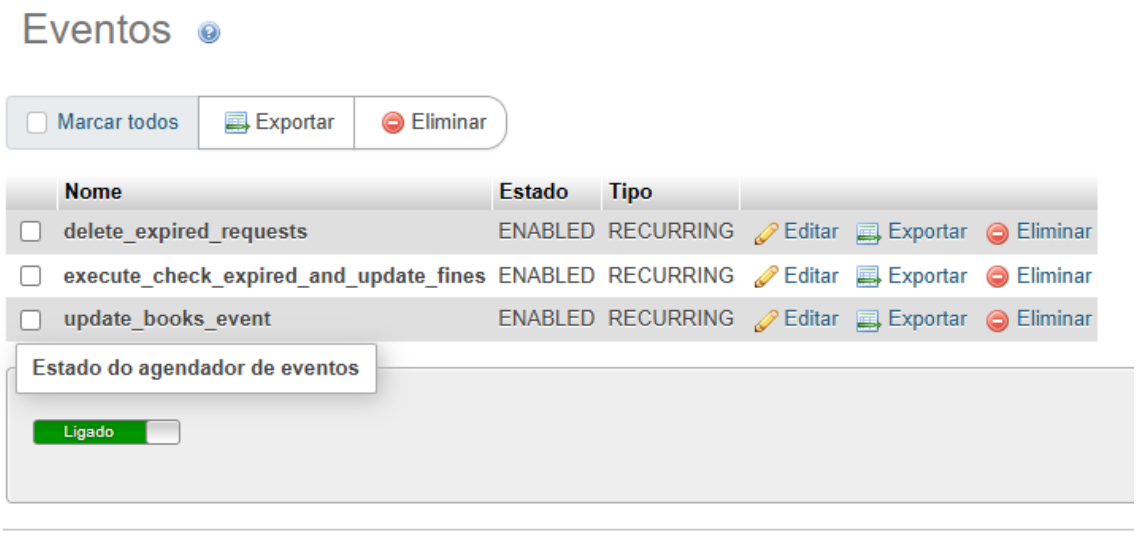


Figura 88 – Página dos eventos no phpMyAdmin

Estes eventos que ocorrem a cada minuto são fundamentais para a manutenção automatizada da plataforma, garantindo que as regras de negócio sejam aplicadas de forma contínua sem a intervenção direta dos utilizadores.

update_books_event

Este evento verifica a condição física dos livros. Se a condição física for igual a 0 (indicando que o livro está muito danificado), ele atualiza os campos available_req e available para 0, tornando o livro indisponível para requisições e reservas.

execute_check_expired_and_update_fines

Este evento executa uma série de chamadas a procedimentos armazenados, que lidam com a verificação de pedidos expirados e a gestão de multas. Isso ajuda a manter o sistema de requisições e as penalidades aplicáveis sempre atualizadas, bloqueando utilizadores se necessário.

delete_expired_requests

Este evento utiliza um cursor para percorrer todas as requisições expiradas, identificando as que já passaram da data de início (start_at < CURDATE()) e não têm uma data de fim (end_at IS NULL) e estão com o status ativo (status = 1). Para cada uma dessas

requisições, o procedimento Remover_Requisicao é chamado, para remover essas requisições.

Esses eventos ajudam a manter a integridade e o fluxo da plataforma ao longo do tempo, reduzindo a necessidade de intervenções manuais. Ajustar a frequência com que esses eventos são executados será importante à medida que a plataforma evolui e o volume de dados aumenta.

6.3. Funções

Uma função no SQL é um bloco de código reutilizável que executa uma operação específica e retorna um valor. Diferente dos procedimentos armazenados (stored procedures), as funções sempre retornam um valor, que pode ser de diversos tipos, como um número, uma string ou uma data. Elas podem receber parâmetros de entrada, realizar cálculos ou manipulações de dados, e são frequentemente usadas em consultas SQL para realizar operações mais complexas de forma simplificada.

A função **AddBusinessDays** recebe como parâmetro um número de dias úteis e retorna uma data que corresponde à data atual somada a esses dias úteis, excluindo fins de semana. O algoritmo percorre os dias, ignorando sábados e domingos, até alcançar a quantidade de dias úteis especificada.

6.4.Procedures

Um procedimento armazenado é um conjunto de instruções SQL que são guardadas na base de dados e podem ser executadas sempre que necessário. Procedimentos permitem agrupar operações complexas numa única chamada, poupando tempo e esforço ao lidar com manipulações de dados repetitivas. Ao contrário das funções, procedimentos armazenados podem ou não retornar valores e podem executar operações que afetam diretamente os dados da base, como inserir, atualizar ou eliminar registos.

Tipos de Procedimentos no Projeto

Neste projeto, existem dois tipos de procedimentos armazenados, diferenciados pela linguagem do nome:

- **Procedimentos com nomes em português**: São aqueles que interagem diretamente com os utilizadores. Quando uma ação é realizada na plataforma, como uma atualização de dados, esses procedimentos são chamados para executar a operação desejada.

- **Procedimentos com nomes em inglês**: São procedimentos internos, chamados apenas dentro da própria base de dados, geralmente como parte de eventos ou outros procedimentos. Estes são usados para automatizar processos que não requerem interação direta do utilizador, como verificações e atualizações automáticas de dados.

7. Instalação Local do Projeto

O projeto A Minha Biblioteca pode ser encontrado no GitHub no seguinte link: satias/projeto-aminhabiblioteca (github.com) . A base de dados necessária também está disponível numa pasta separada chamada base de dados dentro do repositório.

Passos para Instalação

Fazer Download do Projeto:

1. Aceda ao repositório do GitHub e faça o download dos ficheiros.

2. Após o download, extraia a pasta aminhabiblioteca para a pasta htdocs do XAMPP (ou equivalente em outros softwares):

Configuração da Base de Dados:

1. Dentro da pasta do projeto no GitHub, localize a pasta base de dados.

2. Importe o ficheiro SQL da base de dados para o MySQL via phpMyAdmin (ou linha de comando MySQL):

    - Abra o phpMyAdmin (geralmente em http://localhost/phpmyadmin).

    - Crie uma base de dados chamada aminhabiblioteca.

- Importe o ficheiro SQL aminhabiblioteca.sql localizado na pasta base de dados.

Configuração do Ficheiro de Conexão com a Base de Dados:

1. Dentro da pasta aminhabiblioteca, localize o ficheiro “connection.php” dentro da pasta “modelo”.

2. Atualize as credenciais da base de dados dentro do construtor com os valores adequados.

```php
// Construtor da classe - define as configurações de conexão ao base de dados
public function __construct()
{
    // Definir o nome do servidor (normalmente 'localhost')
    $this->host = "localhost";
    // Definir o nome de utilizador para aceder ao MySQL
    $this->user = "root";
    // Definir a palavra-passe para o utilizador do MySQL
    $this->pass =  "";
    // Definir o nome da base de dados à qual se vai conectar
    $this->db = "aminhabiblioteca";
}
```

Figura 89 – Credenciais da base de dados

Configuração de Caminhos no Projeto

Se o projeto A Minha Biblioteca não for alojado diretamente na pasta htdocs (no caso de usar XAMPP) ou em qualquer outro diretório padrão de uma ferramenta de servidor local, será necessário ajustar alguns caminhos de diretório no código. Caso o projeto tenha sido colocado diretamente na pasta htdocs, esta etapa pode ser ignorada.

Existem três ficheiros onde será necessário alterar o caminho para corresponder à localização do projeto no servidor.

1. .htaccess

No ficheiro .htaccess, a diretiva RewriteBase deve ser ajustada para refletir o caminho correto do projeto:

```
DirectoryIndex index.php
RewriteEngine on
RewriteBase /aminhabiblioteca/

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.+)$ index.php?cmd=$1  [QSA,L]
```

Figura 90 – Ficheiro .htaccess

- Linha a ser alterada: RewriteBase /aminhabiblioteca/
- Nova linha: Ajuste o caminho para o diretório onde o projeto está instalado. Exemplo:
    - Se o projeto estiver em http://localhost/meuprojeto/aminhabiblioteca/, a linha deve ser:

RewriteBase /meuprojeto/aminhabiblioteca/

2. route.php

No ficheiro route.php, o caminho base utilizado para identificar as rotas também precisa de ser atualizado:

```
<?php
$request = $_SERVER["REQUEST_URI"];

$base_path = "/aminhabiblioteca";
$rest = substr($request, strlen($base_path));

```

Figura 91- Caminho no ficheiro router.php

- Linha a ser alterada: $base_path = "/aminhabiblioteca";
- Nova linha: Ajuste para o caminho completo. Exemplo:

  $base_path = "/meuprojeto/aminhabiblioteca";

3. funcoes.php

No ficheiro funcoes.php, o caminho base para as funções do projeto também deve ser modificado:

```php
<?php
// Define a variável global que contém o diretório base
$diretorio_base = "/aminhabiblioteca/";
```

Figura 92 – Caminho no ficheiro funções.php

- Linha a ser alterada: $diretorio_base = "/aminhabiblioteca/";
- Nova linha: Ajuste o caminho conforme necessário. Exemplo:

$diretorio_base = "/meuprojeto/aminhabiblioteca/";

Resumo

Caso o projeto tenha sido colocado numa pasta diferente de htdocs, será necessário ajustar os caminhos nos ficheiros:

- .htaccess na linha RewriteBase
- route.php na linha $base_path
- funcoes.php na linha $diretorio_base

Esses ajustes garantem que o roteamento e outras funcionalidades do projeto funcionem corretamente no novo caminho do servidor local.

Iniciar o Servidor Local:

Se todos os passos anteriores foram seguidos corretamente, com a configuração das pastas e ajustes de caminhos realizados, o projeto **A Minha Biblioteca** estará pronto para ser executado. Agora, basta iniciar os serviços **Apache** e **MySQL** no XAMPP ou na ferramenta de servidor local de sua preferência. Com ambos os servidores em funcionamento, a plataforma estará acessível através do navegador e poderá ser utilizada conforme projetado.

8. Planeamento Futuro e Melhorias

No futuro, a plataforma AMinhaBiblioteca pode evoluir com diversas melhorias que irão otimizar a experiência dos utilizadores e a eficiência da gestão da biblioteca. As principais melhorias propostas são:

Responsividade em Todos os Dispositivos: Aproveitando o uso do display grid em quase todas as páginas, a adaptação para dispositivos móveis e tablets será facilitada. O menu será ocultado e acessível por um botão nesses dispositivos, tornando a navegação mais intuitiva.

Melhorias no Sistema de Login: Planeia-se aprimorar o sistema de credenciais, incluindo a implementação de um mecanismo para recuperação de senha, aumentando a segurança e conveniência para os utilizadores.

Sistema de Notificações por Email: Além das notificações internas, será possível enviar alertas e atualizações por email, permitindo que os utilizadores fiquem informados mesmo fora da plataforma.

Implementação de Sistema de Reviews: Após a devolução de um livro, os clientes poderão preencher um questionário de satisfação, permitindo que a biblioteca receba feedback para aprimorar os seus serviços.

Hub de Notificações Personalizável: Melhorias no sistema de notificações, com a adição de filtros e novos tipos de alertas, darão mais controlo aos utilizadores sobre as informações que desejam receber.

Mapa Interativo da Biblioteca: Um mapa detalhado das estantes e prateleiras codificadas ajudará os utilizadores a localizar os livros com facilidade, oferecendo uma experiência de busca eficiente.

Quadro de Avisos com Editor de Texto: Um quadro de avisos permitirá a divulgação de eventos, promoções e atualizações importantes, com suporte a edição de texto avançada via APIs como Tiny.

Melhorias na Tradução: A plataforma integrará uma API de tradução, como o Deepl, para garantir que o conteúdo seja traduzido de forma precisa, melhorando a acessibilidade para utilizadores multilíngues.

Estatísticas com Gráficos para Funcionários e Administradores: Utilizando ferramentas como GoogleCharts ou ChartJS, serão disponibilizados gráficos e visualizações de dados, auxiliando na gestão da biblioteca e na análise de desempenho.

Estas melhorias irão transformar a plataforma numa ferramenta ainda mais robusta, intuitiva e eficiente, atendendo às necessidades tanto dos utilizadores quanto dos funcionários.

9. Conclusões

O desenvolvimento desta plataforma web para gestão de uma biblioteca representou um desafio significativo e uma oportunidade de aplicar os conhecimentos adquiridos ao longo do curso. As principais dificuldades enfrentadas incluíram a criação do design e a implementação na base de dados, especialmente no que diz respeito aos eventos, procedimentos armazenados e views. Apesar dessas dificuldades, foi possível executar todas as funcionalidades planejadas, mas ainda existem funcionalidades desejadas que não foram implementadas devido a limitações de tempo e recursos.

Este projeto beneficiou-se enormemente das diversas disciplinas do curso, que forneceram a base teórica e prática necessária para enfrentar os desafios encontrados. A integração de diferentes áreas de conhecimento permitiu a criação de uma solução robusta e funcional, que atende às necessidades de gestão de uma biblioteca.

Em resumo, este projeto não só aprimorou a minhas habilidades técnicas e de design, mas também destacou a importância da interdisciplinaridade e da colaboração no desenvolvimento de soluções tecnológicas complexas. Embora haja espaço para melhorias e futuras implementações, o resultado alcançado é um testemunho do nosso esforço e dedicação.

10. Referências e Bibliografia

Coolors, Coolors - The super fast color palettes generator!

W3schools, W3Schools Tryit Editor.

Smashing Magazine, Creating A Custom Range Input That Looks Consistent Across All Browsers — Smashing Magazine.

CSS-TRICKS, Styling Cross-Browser Compatible Range Inputs With CSS | CSS-Tricks.

Wappler, Range Input display Min Max Values and show Selected Value - Wappler General / How To - Wappler Community.

Política Privacidade, Política de Privacidade e Termos de Uso - Gerador Grátis Online (politicaprivacidade.com)