

Licenciatura em Engenharia Informática

**StoryTail**English Books For Young Learners

*Luís pinto  
Miguel Pinheiro  
Ricardo Freixo  
Valério Pinheiro*

Índice

[Índice III](#_Toc181869143)

[Índice de figuras IV](#_Toc181869144)

[1. UI/UX 1](#_Toc181869145)

[2. Desenvolvimento 8](#_Toc181869146)

[2.1. Base de Dados 8](#_Toc181869147)

[ANEXO A: Implementa a estrutura da Base de Dados (US\_B001) 9](#_Toc181869148)

[ANEXO B: Inserção de 3 registos em cada tabela (US\_B002) 15](#_Toc181869149)

[ANEXO C: Cria stored procedures com queries para listar todos os livros com regras de filtros, recebidos por parâmetros. (US\_B003) 19](#_Toc181869150)

[ANEXO D: Cria stored procedures com queries para listar as atividades de um dado livro, recebido por parâmetro. (US\_B004) 21](#_Toc181869151)

[ANEXO E: Cria stored procedures com queries para listar os livros favoritos e os livros lidos de um dado utilizador, com o respetivo progresso. (US\_B005) 22](#_Toc181869152)

[ANEXO F: Stored procedures com queries para mostrar livros sugeridos a um dado utilizador (US\_B006) 24](#_Toc181869153)

[ANEXO G: Cria a view para listar os livros mais populares nos últimos 3 meses (US\_B007) 27](#_Toc181869154)

[ANEXO H: Efectua o rating médio de um livro automaticamente (EXTRA) 28](#_Toc181869155)

Índice de figuras

[Figura 1 Protótipo da plataforma web com os vários ecrãs chave 1](#_Toc181712702)

[Figura 2 Diagrama de Gant com distribuição das US para cada elemento 6](#_Toc181712703)

[Figura 3 Poster apresentação do projeto 7](#_Toc181712704)

# UI/UX

O projeto StoryTail visa desenvolver uma plataforma web interativa de histórias infantis em inglês, voltada para crianças de 3 a 9 anos e seus familiares. O desafio da interface e experiência do utilizador (UI/UX) neste contexto envolve criar um ambiente intuitivo, e lúdico, que promova o interesse pela leitura e aprendizado do idioma de forma agradável e envolvente.

Para atingir esses objetivos, o design deve alinhar-se ao público-alvo, utilizando cores leves e elementos visuais atrativos, em consonância com o manual de identidade visual e tendo a mascote (uma raposa) como figura central de referência. A usabilidade é fundamental, pois a plataforma deve permitir uma navegação intuitiva para crianças, que poderão explorar livros, realizar atividades, marcar livros favoritos e acompanhar o progresso de leitura.

Além disso, a interface precisa ser inclusiva e responsiva, adaptando-se a diferentes dispositivos e integrando funcionalidades de fácil acesso tanto para utilizadores não logados quanto logados (com conteúdo adicional para utilizadores premium). Neste sentido, o trabalho em UI/UX envolverá o desenvolvimento de protótipos que ofereçam uma experiência simplificada, eficiente e visualmente agradável, promovendo uma interação intuitiva e segura para os jovens leitores. Foi desenvolvido um protótipo da plataforma web (Figura 1) conforme o manual de identidade visual 'Brand Guidelines' fornecido (US\_B009).

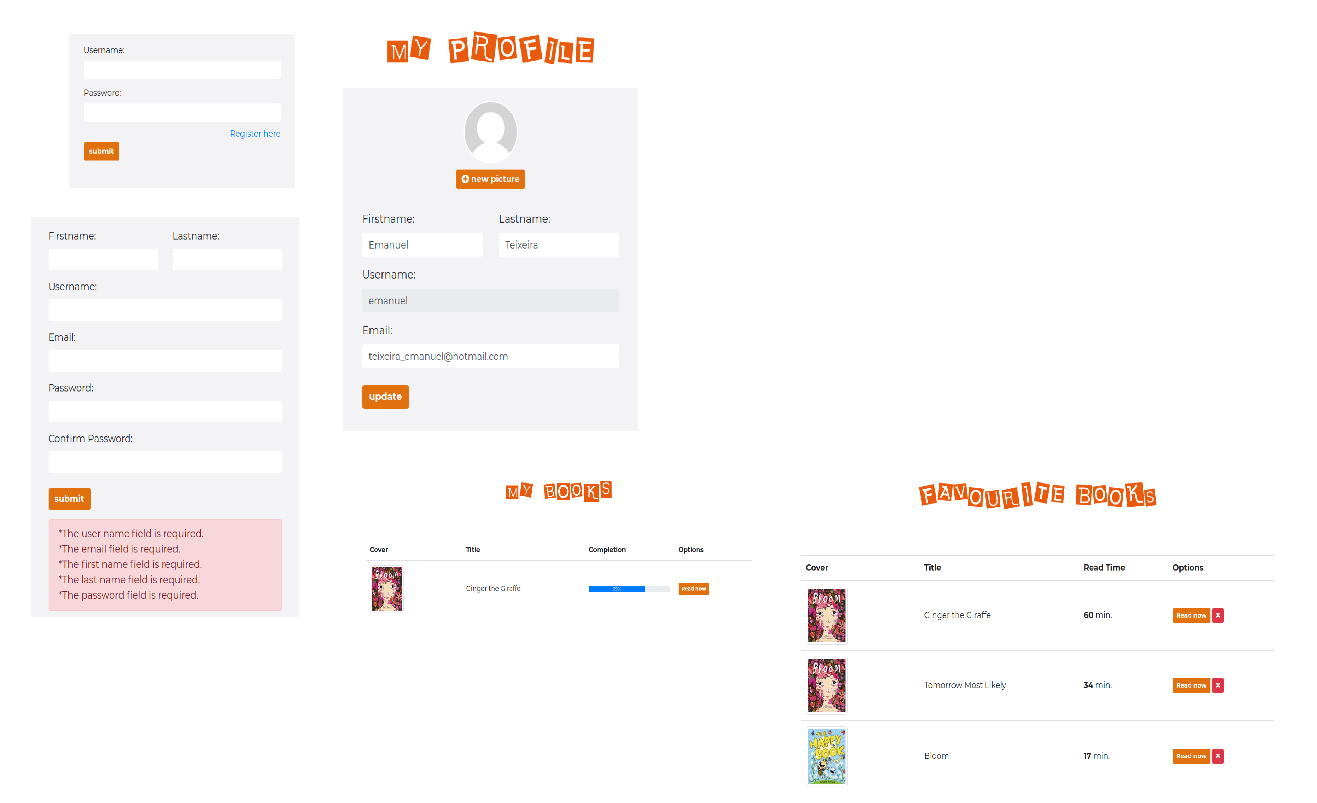


Figura 1 Protótipo da plataforma web com os vários ecrãs chave

Foram detalhadas três personas relativas aos utilizadores (US\_B010), bem como especificada uma User Journey para cada persona definida (US\_B011).

***Personas***

Criança (6 anos) - Tomás

Perfil: Tomás é um menino curioso e gosta de explorar novas histórias.

Necessidades: Acesso fácil a histórias com opções de áudio e vídeo.

Objetivos: Encontrar histórias interativas e acompanhar seu progresso.

Frustrações: Dificuldade em encontrar histórias sem suporte visual e de áudio. Design excessivamente complexo ou menus escondidos que dificultam a exploração livre.

Mãe Utilizadora (Laura, 35 anos)

Perfil: Laura é mãe do Tomás e procura ferramentas educativas para introduzir o inglês ao filho. Ela quer acompanhar o progresso dele, preferindo plataformas com conteúdos apropriados para a aprendizagem.

Necessidades: Um layout intuitivo, onde possa aceder rapidamente o progresso do filho e gerir conteúdos. A plataforma deve oferecer conteúdo adaptado para a faixa etária do filho.

Objetivos: Ajudar o filho a ler mais histórias, incentivando-o a aprender inglês.

Frustrações: Falta de controlo sobre o conteúdo acessível para o filho, dificuldades para encontrar conteúdos de acordo com o nível dele.

Administrador (Carlos, 40 anos)

Perfil: Carlos é responsável por inserir novos conteúdos e atividades na plataforma. Ele gere a base de dados e mantém a experiência interessante e relevante para o público-alvo.

Necessidades: Interface de backoffice eficiente e organizado que permita rápida inserção, edição e exclusão de conteúdos.

Objetivos: Manter a biblioteca atualizada, garantir que as atividades estejam adequadas e incentivar o uso educacional da plataforma, de acordo com o feedback.

Frustrações: Processos manuais ou repetitivos que tornem a gestão de conteúdo demorada.

**User Journey**

Persona 1: Tomás

Objetivo: Encontrar e explorar uma história em inglês com opções de narração em áudio e atividades.

Entrada na Plataforma

Motivação: Tomás está entusiasmado e quer explorar uma história nova.

Ação: Ele acede à plataforma no tablet, atraído por uma interface com ícones grandes e coloridos.

Exploração do Catálogo de Histórias

Motivação: Ele quer uma história visual e fácil de escolher.

Ação: Tomás rola pela lista de histórias usando o "infinite scroll" e vê as capas dos livros, estando algumas recomendadas para sua faixa etária.

Seleção da História

Motivação: Encontrar uma história com imagens coloridas e opções de narração.

Ação: Tomás toca na capa do livro, abrindo a página de detalhes. Ele escolhe a opção de narração em áudio.

Exploração e Interação

Motivação: Quer se divertir enquanto ouve a história.

Ação: Ele acompanha a narração em áudio, vendo as imagens e, ao fim de cada seção, interage com pequenas atividades, como quizzes e desafios de desenho, que aparecem de forma visual e com instruções simples.

Finalização

Motivação: Completar a história e ganhar algum tipo de reconhecimento.

Ação: Ao finalizar, ele vê uma barra de progresso aumentar e recebe uma “estrela” virtual por ter completado a leitura.

Persona 2: Laura

Objetivo: Acompanhar o progresso de leitura e aprendizagem do Tomás, acedendo a sugestões de histórias.

Acesso à Plataforma e Login

Motivação: Laura quer ver o que Tomás tem lido recentemente.

Ação: Ela entra na plataforma e faz login no perfil dela.

Visão Geral do Progresso do Tomás

Motivação: Verificar quais livros o filho leu e se interessou com as atividades.

Ação: Ela navega até o painel de progresso do Tomás, onde visualiza uma lista dos livros concluídos, os favoritos e o progresso nas atividades interativas.

Exploração de Novos Livros

Motivação: Procurar novas histórias que possam interessar ao filho.

Ação: Laura navega pelo catálogo e usa filtros (faixa etária, temas educativos) para encontrar novas sugestões de histórias.

Sugestão de Leitura

Motivação: Ajudar Tomás a escolher novas histórias.

Ação: Ela seleciona uma nova história e a marca como "favorita" para que ele a encontre facilmente. Também adiciona uma atividade como "quiz" para que ele explore ao terminar.

Verificação de Configurações

Motivação: Manter o perfil seguro e personalizado.

Ação: Laura reve as configurações para garantir que o conteúdo adequado à faixa etária e as preferências de Tomás estão bem definidas, ajustando se necessário.

Persona 3: Carlos

Objetivo: Inserir um novo livro na plataforma e configurar atividades no Acesso ao Backoffice

Motivação: Atualizar a biblioteca com um novo livro que acabou de ser adquirido.

Ação: Carlos faz login na área administrativa e acede o módulo de gestão de conteúdo.

Inserção do Novo Livro

Motivação: Adicionar o novo livro de forma rápida e intuitiva.

Ação: Ele insere as informações básicas (título, faixa etária, autor, tags, imagem da capa) e carrega o texto e áudio para as diferentes modalidades de leitura (texto, narração e vídeo).

Configuração de Atividades Interativas

Motivação: Criar atividades para reforçar a mensagem educativa do livro.

Ação: Carlos configura um quiz com perguntas relacionadas ao enredo e uma atividade de desenho. Ele define o nível de acesso como “restrito” (somente para usuários premium).

Revisão e Publicação

Motivação: Garantir que todas as informações estão corretas.

Ação: Antes de publicar, Carlos revê todos os dados do livro, as atividades e salva as alterações, tornando o conteúdo ativo na plataforma.

Foram distribuídas as US para cada elemento representadas no diagrama de Gantt da figura 2(US\_B013).

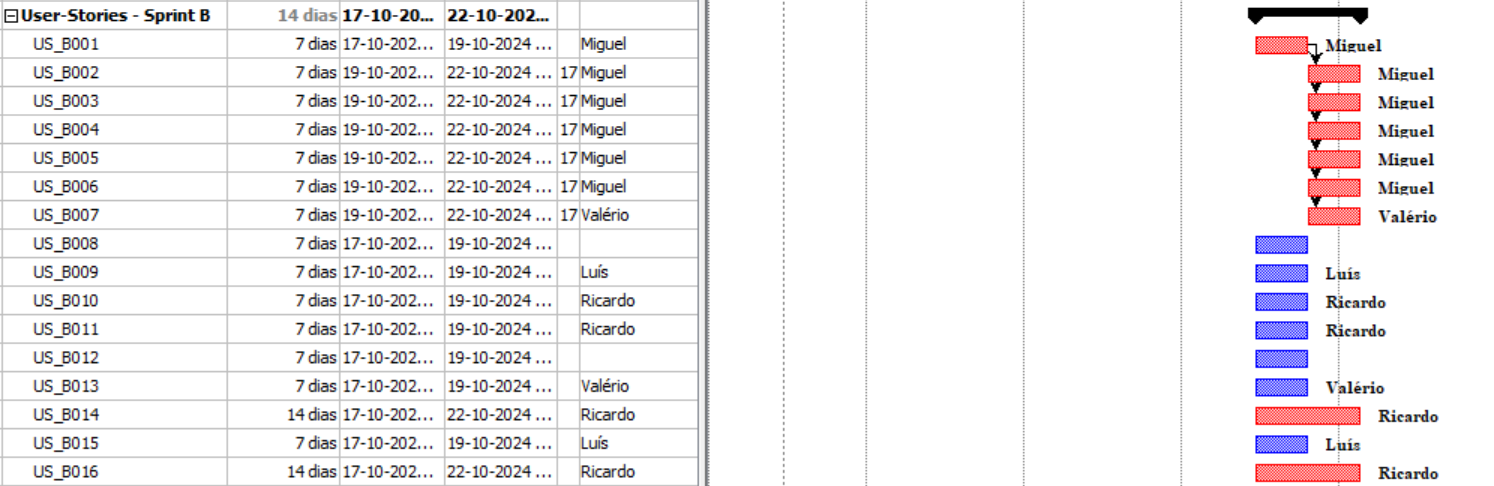


Figura 2 Diagrama de Gant com distribuição das US para cada elemento

Foi também desenvolvido um poster apresentação do projeto representado na figura 3.

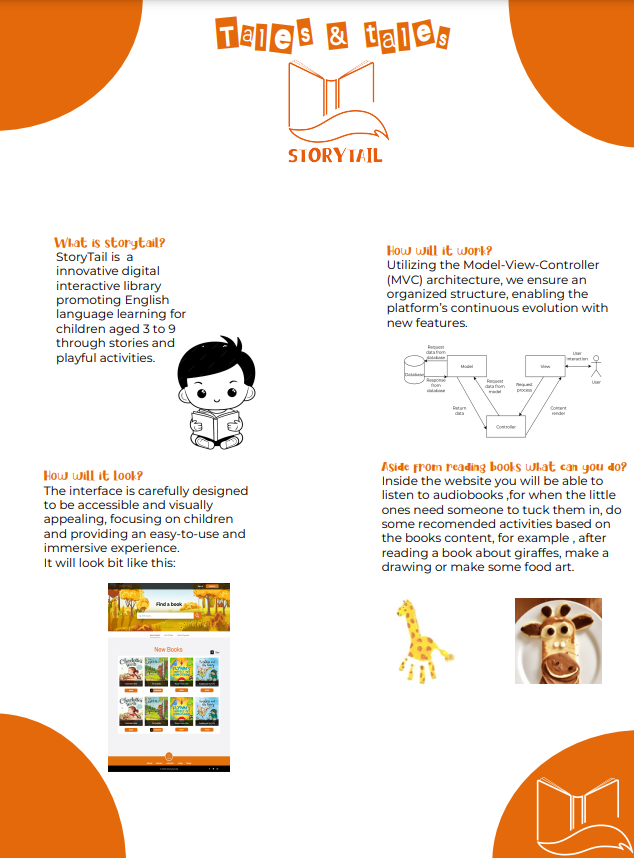
****

Figura 3 Poster apresentação do projeto

# Desenvolvimento

### Base de Dados

A estrutura da Base de Dados foi implementada com sucesso, permitindo armazenar as informações de livros, atividades e utilizadores da plataforma StoryTail. Esse passo inicial permitiu uma base robusta para o desenvolvimento das funcionalidades subsequentes (US\_B001) (Anexo A). Três registos foram inseridos em cada tabela para simular dados reais. Essa inserção inicial possibilitou a validação da estrutura e das consultas, garantindo que as interações e testes funcionassem conforme esperado com informações simuladas. (US\_B002) (Anexo B). As stored procedures foram desenvolvidas para listar todos os livros de acordo com filtros recebidos por parâmetros. Com isso, a plataforma agora pode oferecer opções de busca personalizada, essencial para atender a diversas necessidades e preferências dos utilizadores (US\_B003) (Anexo C). A criação de stored procedures para listar atividades associadas a cada livro permite que a plataforma exiba conteúdos adicionais específicos de cada história (US\_B004) (Anexo D). Outra stored procedure foi implementada para listar livros favoritos e lidos de cada utilizador, com o progresso de leitura incluído. (US\_B005) (Anexo E). A plataforma agora conta com uma stored procedure que sugere livros com base no perfil do utilizador, utilizando seu histórico e preferências. Essa funcionalidade melhora a personalização, ajudando na descoberta de novos conteúdos relevantes (US\_B006) (Anexo F). Uma view foi criada para identificar e exibir os livros mais populares nos últimos três meses. Com essa funcionalidade, a plataforma oferece informação sobre os conteúdos mais consumidos entre os utilizadores, ajudando a destacar conteúdos que têm maior apelo (US\_B007) (Anexo G). Como atividade extra foi desenvolvido o rating médio de um livro automaticamente.

ANEXO A: Implementa a estrutura da Base de Dados (US\_B001)

CREATE DATABASE bdteste;

USE bdteste;

CREATE TABLE authors (

id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

description TEXT,

author\_photo\_url VARCHAR(255),

nationality VARCHAR(100),

created\_at DATETIME,

updated\_at DATETIME

);

CREATE TABLE books (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255),

description TEXT,

cover\_url VARCHAR(255),

read\_time INT,

rating\_medio FLOAT,

age\_group VARCHAR(50),

is\_active BOOLEAN,

access\_level INT,

created\_at DATETIME,

updated\_at DATETIME

);

CREATE TABLE author\_book (

id INT AUTO\_INCREMENT PRIMARY KEY,

author\_id INT,

book\_id INT,

FOREIGN KEY (author\_id) REFERENCES authors(id),

FOREIGN KEY (book\_id) REFERENCES books(id)

);

CREATE TABLE pages (

id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

page\_image\_url VARCHAR(255),

audio\_url VARCHAR(255),

page\_index INT,

created\_at DATETIME,

updated\_at DATETIME,

FOREIGN KEY (book\_id) REFERENCES books(id)

);

CREATE TABLE videos (

id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

title VARCHAR(255),

video\_url VARCHAR(255),

created\_at DATETIME,

updated\_at DATETIME,

FOREIGN KEY (book\_id) REFERENCES books(id)

);

CREATE TABLE tags (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

created\_at DATETIME,

updated\_at DATETIME

);

CREATE TABLE tagging\_tagged (

id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

tag\_id INT,

FOREIGN KEY (book\_id) REFERENCES books(id),

FOREIGN KEY (tag\_id) REFERENCES tags(id)

);

CREATE TABLE age\_groups (

age\_group VARCHAR(50) PRIMARY KEY,

created\_at DATETIME,

updated\_at DATETIME

);

CREATE TABLE activities (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255),

description TEXT,

created\_at DATETIME,

updated\_at DATETIME

);

CREATE TABLE activity\_images (

id INT AUTO\_INCREMENT PRIMARY KEY,

activity\_id INT,

title VARCHAR(255),

image\_url VARCHAR(255),

created\_at DATETIME,

updated\_at DATETIME,

FOREIGN KEY (activity\_id) REFERENCES activities(id)

);

CREATE TABLE activity\_book (

id INT AUTO\_INCREMENT PRIMARY KEY,

activity\_id INT,

book\_id INT,

FOREIGN KEY (activity\_id) REFERENCES activities(id),

FOREIGN KEY (book\_id) REFERENCES books(id)

);

CREATE TABLE user\_types (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_type VARCHAR(100),

created\_at DATETIME,

updated\_at DATETIME

);

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_type\_id INT,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

user\_name VARCHAR(100),

email VARCHAR(255),

password VARCHAR(255),

user\_photo\_url VARCHAR(255),

created\_at DATETIME,

updated\_at DATETIME,

FOREIGN KEY (user\_type\_id) REFERENCES user\_types(id)

);

CREATE TABLE activity\_book\_user (

id INT AUTO\_INCREMENT PRIMARY KEY,

activity\_book\_id INT,

user\_id INT,

progress INT,

FOREIGN KEY (activity\_book\_id) REFERENCES activity\_book(id),

FOREIGN KEY (user\_id) REFERENCES users(id)

);

CREATE TABLE book\_user\_favourite (

id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

user\_id INT,

FOREIGN KEY (book\_id) REFERENCES books(id),

FOREIGN KEY (user\_id) REFERENCES users(id)

);

CREATE TABLE book\_user\_read (

id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

user\_id INT,

progress INT,

rating INT,

read\_date DATETIME,

FOREIGN KEY (book\_id) REFERENCES books(id),

FOREIGN KEY (user\_id) REFERENCES users(id)

);

CREATE TABLE plans (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

access\_level INT,

created\_at DATETIME,

updated\_at DATETIME

);

CREATE TABLE subscriptions (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

plan\_id INT,

start\_date DATETIME,

created\_at DATETIME,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (plan\_id) REFERENCES plans(id)

);

ANEXO B: Inserção de 3 registos em cada tabela (US\_B002)

INSERT INTO authors (first\_name, last\_name, description, nationality, created\_at, updated\_at)

VALUES

('John', 'Doe', 'Author of fictional books', 'American', NOW(), NOW()),

('Jane', 'Smith', 'Renowned for science fiction', 'British', NOW(), NOW()),

('Emily', 'Johnson', 'Specializes in historical novels', 'Canadian', NOW(), NOW());

INSERT INTO books (title, description, read\_time, age\_group, is\_active, access\_level, created\_at, updated\_at)

VALUES

('The Mystery Book', 'A thrilling mystery novel', 120, 'Adult', TRUE, 1, NOW(), NOW()),

('The Sci-Fi Adventure', 'Exploring galaxies and technology', 150, 'Young Adult', TRUE, 1, NOW(), NOW()),

('The History Chronicles', 'A dive into ancient civilizations', 180, 'Adult', TRUE, 1, NOW(), NOW());

INSERT INTO author\_book (author\_id, book\_id)

VALUES

(1, 1),

(2, 2),

(3, 3);

INSERT INTO tags (name, created\_at, updated\_at)

VALUES

('Mystery', NOW(), NOW()),

('Science Fiction', NOW(), NOW()),

('History', NOW(), NOW());

INSERT INTO tagging\_tagged (book\_id, tag\_id)

VALUES

(1, 1),

(2, 2),

(3, 3);

INSERT INTO age\_groups (age\_group, created\_at, updated\_at)

VALUES

('Infantil', NOW(), NOW()),

('Jovem', NOW(), NOW()),

('Adulto', NOW(), NOW());

INSERT INTO user\_types (user\_type, created\_at, updated\_at)

VALUES

('Admin', NOW(), NOW()),

('Editor', NOW(), NOW()),

('Viewer', NOW(), NOW());

INSERT INTO users (user\_type\_id, first\_name, last\_name, user\_name, email, password, user\_photo\_url, created\_at, updated\_at)

VALUES

(1, 'Alice', 'Smith', 'alice\_s', 'alice@example.com', 'password123', NULL, NOW(), NOW()),

(2, 'Bob', 'Brown', 'bob\_b', 'bob@example.com', 'password456', NULL, NOW(), NOW()),

(3, 'Charlie', 'Davis', 'charlie\_d', 'charlie@example.com', 'password789', NULL, NOW(), NOW());

INSERT INTO plans (name, access\_level, created\_at, updated\_at)

VALUES

('Premium Plan', 2, NOW(), NOW()),

('Standard Plan', 1, NOW(), NOW()),

('Basic Plan', 0, NOW(), NOW());

INSERT INTO subscriptions (user\_id, plan\_id, start\_date, created\_at)

VALUES

(1, 1, NOW(), NOW()),

(2, 2, NOW(), NOW()),

(3, 3, NOW(), NOW());

INSERT INTO book\_user\_favourite(book\_id, user\_id)

VALUES

(1, 1),

(2, 2),

(3, 3);

INSERT INTO activities (title, description, created\_at, updated\_at)

VALUES

('Quiz on Mystery Book', 'A quiz activity based on the mystery book', NOW(), NOW()),

('Puzzle Game', 'Solve the puzzle related to the storyline of the book', NOW(), NOW()),

('Trivia Challenge', 'General trivia based on various books', NOW(), NOW());

INSERT INTO activity\_book (activity\_id, book\_id)

VALUES

(1, 1),

(2, 1),

(3, 2);

INSERT INTO activity\_images (activity\_id, title, image\_url, created\_at, updated\_at)

VALUES

(1, 'Quiz Image', 'http://example.com/quiz\_image.png', NOW(), NOW()),

(2, 'Puzzle Image', 'http://example.com/puzzle\_image.png', NOW(), NOW()),

(3, 'Trivia Image', 'http://example.com/trivia\_image.png', NOW(), NOW());

INSERT INTO activity\_book\_user (activity\_book\_id, user\_id, progress)

VALUES

(1, 1, 50),

(2, 2, 30),

(3, 3, 70);

INSERT INTO book\_user\_read (book\_id, user\_id, progress, rating, read\_date)

VALUES

(1, 1, 100, 5, NOW()),

(2, 2, 60, 4, NOW()),

(3, 3, 20, 3, NOW());

DELIMITER $$

ANEXO C: Cria stored procedures com queries para listar todos os livros com regras de filtros, recebidos por parâmetros. (US\_B003)

CREATE PROCEDURE ListBooksByTags(

IN p\_tags VARCHAR(255), -- Parâmetro para lista de tags, separadas por vírgula

IN p\_is\_active BOOLEAN, -- Parâmetro opcional para filtrar por status do livro (ativo/inativo)

IN p\_age\_group VARCHAR(50) -- Parâmetro opcional para filtrar pelo grupo de idade

)

BEGIN

-- Variáveis temporárias para manipulação das tags

DECLARE tag\_count INT DEFAULT 0;

DECLARE tag\_condition TEXT DEFAULT '';

-- Contar a quantidade de tags recebidas no parâmetro p\_tags

SET tag\_count = (LENGTH(p\_tags) - LENGTH(REPLACE(p\_tags, ',', '')) + 1);

-- Gerar a condição de tags dinamicamente

SET tag\_condition = CONCAT(

' AND (SELECT COUNT(DISTINCT t.id) FROM tags t ',

'JOIN tagging\_tagged tt ON t.id = tt.tag\_id ',

'WHERE tt.book\_id = b.id AND FIND\_IN\_SET(t.name, ?)) = ?'

);

-- Construção da query principal para seleção dos livros

SET @query = CONCAT(

'SELECT b.id, b.title, b.description, b.cover\_url, b.read\_time, b.age\_group ',

'FROM books b ',

'JOIN tagging\_tagged tt ON b.id = tt.book\_id ',

'JOIN tags t ON t.id = tt.tag\_id ',

'WHERE FIND\_IN\_SET(t.name, ?) > 0'

);

-- Adiciona condição para status do livro, se for passado o parâmetro p\_is\_active

IF p\_is\_active IS NOT NULL THEN

SET @query = CONCAT(@query, ' AND b.is\_active = ', p\_is\_active);

END IF;

-- Adiciona condição para o grupo de idade, se for passado o parâmetro p\_age\_group

IF p\_age\_group IS NOT NULL THEN

SET @query = CONCAT(@query, ' AND b.age\_group = "', p\_age\_group, '"');

END IF;

-- Executa a query com os parâmetros dinâmicos

SET @query = CONCAT(@query, ' GROUP BY b.id');

PREPARE stmt FROM @query;

SET @p\_tags = p\_tags;

EXECUTE stmt USING @p\_tags;

DEALLOCATE PREPARE stmt;

END$$

DELIMITER ;

-- CALL ListBooksByTags('Mystery,History', NULL, NULL);

ANEXO D: Cria stored procedures com queries para listar as atividades de um dado livro, recebido por parâmetro. (US\_B004)

CREATE PROCEDURE ListBookActivities(IN bookId INT)

BEGIN

SELECT

a.id AS activity\_id,

a.title AS activity\_title,

a.description AS activity\_description,

a.created\_at AS activity\_created\_at,

a.updated\_at AS activity\_updated\_at

FROM activity\_book ab

JOIN activities a ON ab.activity\_id = a.id

WHERE ab.book\_id = bookId;

END$$

DELIMITER ;

-- CALL ListBookActivities(1);

ANEXO E: Cria stored procedures com queries para listar os livros favoritos e os livros lidos de um dado utilizador, com o respetivo progresso. (US\_B005)

CREATE PROCEDURE ListUserFavouriteBooks(IN userId INT)

BEGIN

SELECT

b.id AS book\_id,

b.title,

b.description,

b.read\_time,

b.age\_group,

b.is\_active,

b.cover\_url,

u.first\_name,

u.last\_name,

u.email

FROM book\_user\_favourite buf

JOIN books b ON buf.book\_id = b.id

JOIN users u ON buf.user\_id = u.id

WHERE buf.user\_id = userId;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE ListUserReadBooks(IN userId INT)

BEGIN

SELECT

b.id AS book\_id,

b.title,

b.description,

b.read\_time,

b.age\_group,

b.is\_active,

b.cover\_url,

bur.progress,

bur.rating,

bur.read\_date,

u.first\_name,

u.last\_name,

u.email

FROM book\_user\_read bur

JOIN books b ON bur.book\_id = b.id

JOIN users u ON bur.user\_id = u.id

WHERE bur.user\_id = userId;

END$$

DELIMITER ;

-- CALL ListUserReadBooks(1);

DELIMITER $$

ANEXO F: Stored procedures com queries para mostrar livros sugeridos a um dado utilizador (US\_B006)

CREATE PROCEDURE SuggestedBooksForUser(

IN p\_user\_id INT

)

BEGIN

-- Sugere livros que partilham tags com os livros favoritos do utilizador

SELECT DISTINCT b.id AS book\_id,

b.title,

b.description,

b.cover\_url,

b.age\_group,

AVG(bur.rating) AS avg\_rating,

COUNT(bur.id) AS total\_reads

FROM books b

JOIN tagging\_tagged tt ON b.id = tt.book\_id

JOIN tags t ON tt.tag\_id = t.id

JOIN book\_user\_favourite buf ON buf.book\_id = b.id

LEFT JOIN book\_user\_read bur ON b.id = bur.book\_id

WHERE buf.user\_id = p\_user\_id -- Compara com favoritos do utilizador

AND b.id NOT IN ( -- Exclui livros já lidos pelo utilizador

SELECT book\_id

FROM book\_user\_read

WHERE user\_id = p\_user\_id

)

GROUP BY b.id

ORDER BY avg\_rating DESC, total\_reads DESC -- Ordena pela média de avaliação e leituras

LIMIT 10; -- Limitar a 10 livros

-- Sugere livros populares que o utilizador ainda não leu

SELECT DISTINCT b.id AS book\_id,

b.title,

b.description,

b.cover\_url,

b.age\_group,

AVG(bur.rating) AS avg\_rating,

COUNT(bur.id) AS total\_reads

FROM books b

LEFT JOIN book\_user\_read bur ON b.id = bur.book\_id

WHERE b.id NOT IN ( -- Exclui livros já lidos pelo utilizador

SELECT book\_id

FROM book\_user\_read

WHERE user\_id = p\_user\_id

)

GROUP BY b.id

ORDER BY total\_reads DESC, avg\_rating DESC -- Ordena por popularidade e média de avaliação

LIMIT 10; -- Limitar a 10 livros

-- Sugere livros do mesmo grupo de idade favoritos por outros utilizadores

SELECT DISTINCT b.id AS book\_id,

b.title,

b.description,

b.cover\_url,

b.age\_group,

AVG(bur.rating) AS avg\_rating,

COUNT(bur.id) AS total\_reads

FROM books b

JOIN book\_user\_favourite buf ON buf.book\_id = b.id

LEFT JOIN book\_user\_read bur ON b.id = bur.book\_id

WHERE b.age\_group = ( -- Filtra pelo grupo de idade

SELECT age\_group

FROM books

WHERE id IN (SELECT book\_id FROM book\_user\_favourite WHERE user\_id = p\_user\_id)

LIMIT 1

)

AND b.id NOT IN ( -- Exclui livros já lidos pelo utilizador

SELECT book\_id

FROM book\_user\_read

WHERE user\_id = p\_user\_id

)

GROUP BY b.id

ORDER BY avg\_rating DESC, total\_reads DESC -- Ordena pela média de avaliação e leituras totais

LIMIT 10; -- Limitar a 10 livros

END$$

DELIMITER ;

-- CALL SuggestedBooksForUser(1);

ANEXO G: Cria a view para listar os livros mais populares nos últimos 3 meses (US\_B007)

CREATE VIEW PopularBooksLast3Months AS

SELECT

b.id AS book\_id,

b.title,

b.description,

b.cover\_url,

COUNT(bur.id) AS total\_reads, -- Total de vezes que o livro foi lido

AVG(bur.rating) AS average\_rating, -- Média das avaliações

SUM(bur.progress) / COUNT(bur.id) AS avg\_progress -- Progresso médio das leituras pelos utilizadores

FROM

books b

JOIN

book\_user\_read bur ON b.id = bur.book\_id

WHERE

bur.read\_date >= DATE\_SUB(CURDATE(), INTERVAL 3 MONTH) -- Últimos 3 meses

GROUP BY

b.id, b.title, b.description, b.cover\_url

ORDER BY

total\_reads DESC, -- Ordena por total de leituras (popularidade)

average\_rating DESC, -- Ordena por média de avaliação

avg\_progress DESC; -- Ordena por progresso médio

-- SELECT \* FROM PopularBooksLast3Months;

ANEXO H: Efetua o rating médio de um livro automaticamente (EXTRA)

CREATE TRIGGER update\_book\_rating

AFTER INSERT ON book\_user\_read

FOR EACH ROW

BEGIN

DECLARE avg\_rating DECIMAL(3, 2);

-- Calcula a média de rating para o livro específico

SELECT AVG(rating) INTO avg\_rating

FROM book\_user\_read

WHERE book\_id = NEW.book\_id;

-- Atualiza a coluna rating\_medio na tabela books

UPDATE books

SET rating\_medio = avg\_rating

WHERE id = NEW.book\_id;

END;

$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER update\_book\_rating\_on\_update

AFTER UPDATE ON book\_user\_read

FOR EACH ROW

BEGIN

DECLARE avg\_rating DECIMAL(3, 2);

-- Calcula a média de rating para o livro específico

SELECT AVG(rating) INTO avg\_rating

FROM book\_user\_read

WHERE book\_id = NEW.book\_id;

-- Atualiza a coluna rating\_medio na tabela books

UPDATE books

SET rating\_medio = avg\_rating

WHERE id = NEW.book\_id;

END;

DELIMITER ;