LICENCIATURA EM ENGENHARIA INFORMÁTICA

STORYTAIL

ENGLISH BOOKS FOR YOUNG LEARNERS

LUÍS PINTO
MIGUEL PINHEIRO
RICARDO FREIXO
VALÉRIO PINHEIRO

LABORATÓRIO DE PROGRAMAÇÃO

VILA NOVA DE GAIA
NOV DE 2024

Índice

Índice de figuras

1. 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).

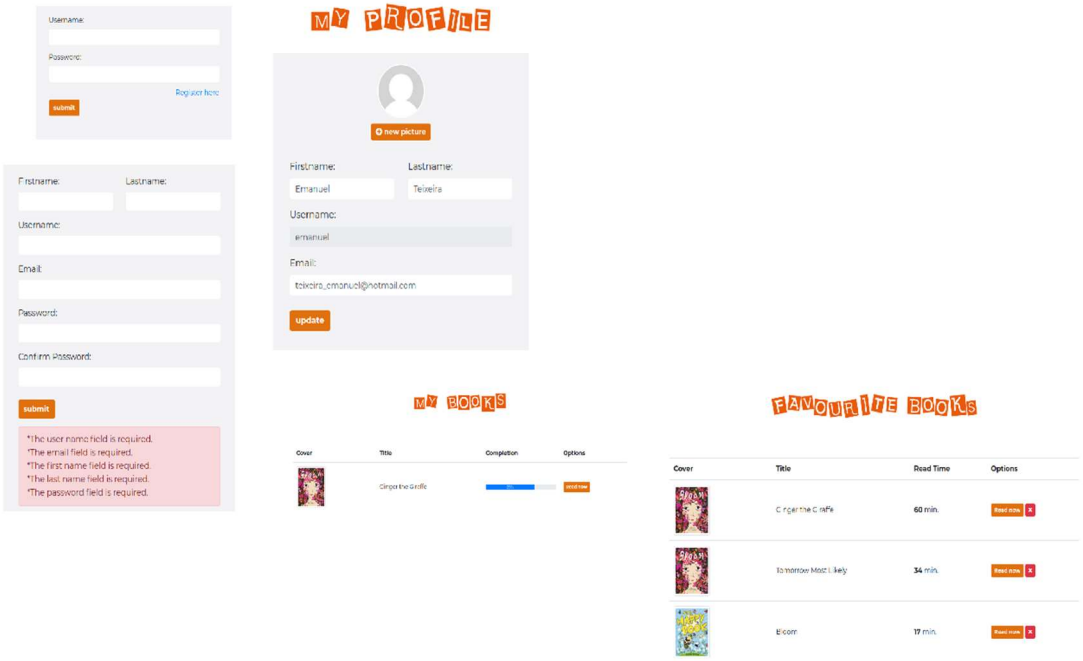

Figura 1 Protótipo da plataforma web com os vários ecrãs chave

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).

User-Stories - Sprint B	14 dias	17-10-20...	22-10-202...		
US_B001	7 dias	17-10-202...	19-10-2024 ...		Miguel
US_B002	7 dias	19-10-202...	22-10-2024 ...	17	Miguel
US_B003	7 dias	19-10-202...	22-10-2024 ...	17	Miguel
US_B004	7 dias	19-10-202...	22-10-2024 ...	17	Miguel
US_B005	7 dias	19-10-202...	22-10-2024 ...	17	Miguel
US_B006	7 dias	19-10-202...	22-10-2024 ...	17	Miguel
US_B007	7 dias	19-10-202...	22-10-2024 ...	17	Valério
US_B008	7 dias	17-10-202...	19-10-2024 ...		
US_B009	7 dias	17-10-202...	19-10-2024 ...		Luís
US_B010	7 dias	17-10-202...	19-10-2024 ...		Ricardo
US_B011	7 dias	17-10-202...	19-10-2024 ...		Ricardo
US_B012	7 dias	17-10-202...	19-10-2024 ...		
US_B013	7 dias	17-10-202...	19-10-2024 ...		Valério
US_B014	14 dias	17-10-202...	22-10-2024 ...		Ricardo
US_B015	7 dias	17-10-202...	19-10-2024 ...		Luís
US_B016	14 dias	17-10-202...	22-10-2024 ...		Ricardo

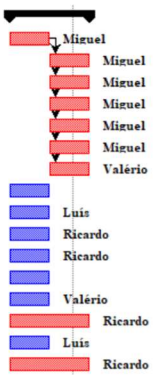

Figura 2 Diagrama de Gant com distribuição das US para cada elemento

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

Figura 3 Poster apresentação do projeto

2. Desenvolvimento

2.1. 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 ;
```