instituto politécnico de gestão e tecnologia

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

SAA-IMP-048.01

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 aprendizagem 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).

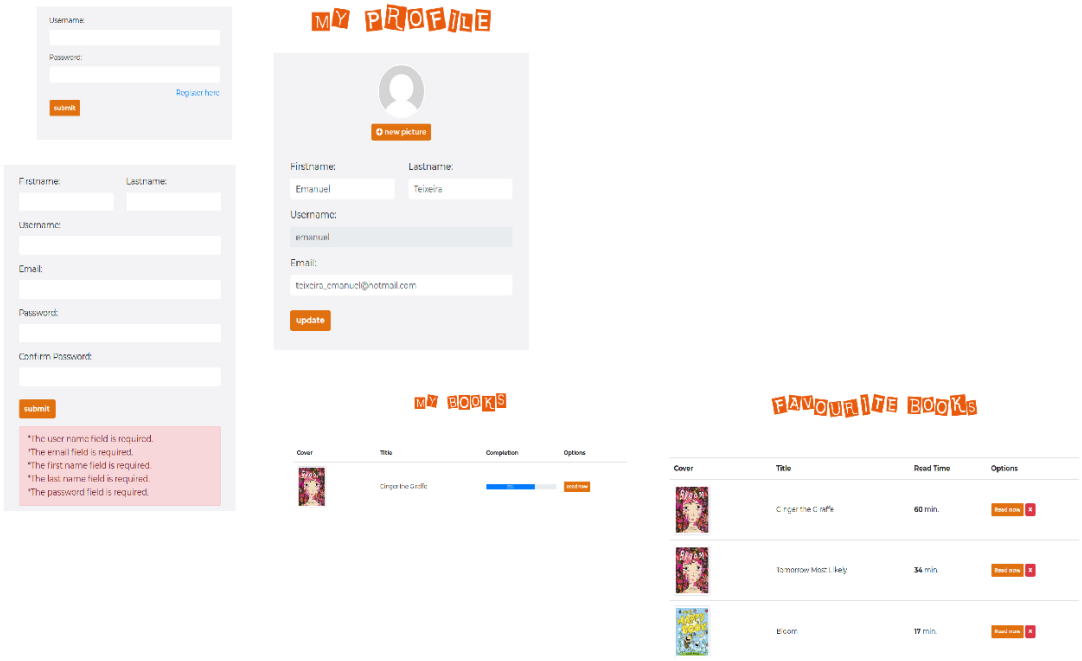

Figura 1 Protótipo da plataforma web com os vários ecrãs chave

2. Desenvolvimento

2.1 Arquitetura de Software

Arquitetura de Software Desenvolvida no Sprint C

A arquitetura do projeto foi estruturada utilizando o framework Laravel, que segue o padrão de design MVC (Model-View-Controller). Essa abordagem divide a aplicação em três camadas principais:

- **Model:** Responsável pela lógica de negócios e pela interação com a base de dados. No Laravel, utilizamos o Eloquent ORM para manipulação de dados de forma eficiente, além de consultas SQL otimizadas com stored procedures e views criadas previamente.
- **View:** Implementada utilizando o **Blade**, o motor de templates nativo do Laravel, que permitiu criar interfaces dinâmicas e reutilizáveis.
- **Controller:** Intermediário entre as views e os models, processando requisições HTTP e executando ações de acordo com a lógica de negócios.

Adicionalmente, o Laravel foi escolhido por suas funcionalidades nativas que garantem segurança e escalabilidade. A modularidade deste framework também possibilita a rápida adição de novas funcionalidades sem comprometer a estrutura existente.

2.2 Front-End

Desenvolvimentos do Sprint C

Durante o Sprint C, a integração do Laravel com o front-end foi realizada utilizando Blade e bibliotecas modernas de JavaScript. As principais entregas incluem:

- **Página Inicial:** Renderização dinâmica de livros recomendados com dados fornecidos pelos controllers do Laravel.
- **Catálogo de Livros:** Implementação de filtros, permitindo atualizações rápidas sem recarregar a página.
- **Perfil do Utilizador:** Desenvolvimento de uma interface intuitiva para exibir e editar dados do utilizador.
- **Responsividade:** Ajustes para garantir que todas as páginas sejam responsivas, utilizando CSS frameworks como Bootstrap, integrado com Blade.

2.3 Back-end

No back-end, o uso do Laravel possibilitou a criação eficiente de APIs e funcionalidades específicas:

- **API para Catálogo de Livros:** Construída utilizando os recursos de rotas e controladores do Laravel, permitindo filtragem avançada e suporte a diferentes parâmetros de busca.
- **Autenticação:** Implementada utilizando o sistema nativo do Laravel, com suporte a autenticação via **JWT (JSON Web Tokens)** para segurança.
- **Integração de Armazenamento:** Uso do Laravel Storage para upload e gestão de arquivos, como imagens e áudios associados aos livros e atividades.

ANEXO A: Implementação do view da página principal

```
@extends('layouts.app')

@section('title', 'Home - Storytails')

@section('content')

<!-- Search Section with Background Image -->

<header style="background-image: url('{{ asset('assets/img/background.png') }}');">

  <div class="container mx-auto text-center py-16">

    <h1 class="text-4xl font-bold text-gray-800">Find a book</h1>

    <form action="" method="GET" class="d-flex justify-content-center">

      <input

        type="text"

        name="query"

        placeholder="e.g. title, author..."

        class="form-control w-50 py-2 rounded-start border-0"

        style="max-width: 500px;"

      >

      <button type="submit" class="btn btn-warning rounded-end">

        <i class="bi bi-search"></i> Search

      </button>
```

```
      </form>
    </div>
</header>
<!-- Navigation Tabs for the Home Page -->
<nav class="bg-white shadow-md mt-4">
    <div class="container mx-auto flex justify-around py-4">
        <a href="#" class="text-orange-500 font-semibold">New Books</a>
        <a href="#" class="text-gray-600 hover:text-orange-500">Our Picks</a>
        <a href="#" class="text-gray-600 hover:text-orange-500">Most Popular</a>
    </div>
</nav>
<!-- Main Content for the page -->
<main class="container my-5">
    <h2 class="text-warning mb-4">New Books</h2>
    <div class="row row-cols-1 row-cols-md-4 g-4">
        @foreach($books as $book)
        <div class="col mb-5">
            <div class="card h-100">
                <img class="card-img-top" src="{{ $book->cover_image }}" alt="{{ $book->title
}}" />
                <div class="card-body p-4">
```

```
                <div class="text-center">
                    <h5 class="fw-bolder">{{ $book->title }}</h5>
                    <p>{{ $book->description }}</p>
                </div>
            </div>
            <div class="card-footer p-4 pt-0 border-top-0 bg-transparent">
                <div class="text-center">
                        <a class="btn btn-outline-dark mt-auto" href="{{ route('book.details', $book->id) }}">View details</a>
                </div>
            </div>
        </div>
    </div>
    @endforeach
  </div>
</main>
@endsection
```

ANEXO B: Implementação do controlador de livros

```
<?php
namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
use App\Models\Book;
use App\Models\Author;
use App\Models\AuthorBook;
use App\Models\Plan;
use App\Models\AgeGroup;
use Illuminate\Support\Facades\Storage;
class BookController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $books = Book::all();
        return view('book.index', [
            'books' => $books,
        ]);
    }
    /**
     * Show the form for creating a new resource.
```

```
     */
    public function create()
    {
        $authors = Author::all();
        $author_books = AuthorBook::all();
        $plans = Plan::orderBy('access_level', 'desc')->get();
        $age_groups = AgeGroup::all();
        return view('book.create', [
            'authors' => $authors,
            'author_books' => $author_books,
            'plans' => $plans,
            'age_groups' => $age_groups,
        ]);
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $request->validate([
```

```
            'title' => 'required|max:255|min:3|unique:books',
            'description' => 'required|min:5',
            'cover_url' => 'nullable|max:255',
            'read_time' => 'required|integer|min:1',
            'rating_medio' => 'nullable|numeric|between:0,5',
            'age_group' => 'required|max:50',
            'is_active' => 'required|between:0,1',
            'access_level' => 'required|integer|between:0,2',
            'pdf' => 'nullable|file|mimes:pdf|max:2048',
        ]);
        $book = new Book();
        $book->title = $request->title;
        $book->description = $request->description;
        $book->cover_url = $request->cover_url;
        $book->read_time = $request->read_time;
        $book->rating_medio = $request->rating_medio;
        $book->age_group = $request->age_group;
        $book->access_level = $request->access_level;
        $book->is_active = $request->is_active;
    $book->pdf_path = $request->pdf_path;
        // Handle PDF file upload
```

```
        $pdfPath = null;
        if ($request->hasFile('pdf')) {
                    $pdfPath = $request->file('pdf')->store('public/pdf'); // Store in storage/app/public/pdf
        }
        $book->save();
        if ($request->has('authors')) {
            foreach ($request->authors as $authorId) {
                AuthorBook::create([
                    'book_id' => $book->id,
                    'author_id' => $authorId
                ]);
            }
        }

        return redirect()->route('book.index')->with('success', 'Book created successfully.');
    }
    /**
     * Display the specified resource.
     */
    public function show(string $id)
```

```
{
    $book = Book::find($id);
    $plans = Plan::orderBy('access_level', 'desc')->get();
    $authors = Author::all();
    $age_groups = AgeGroup::all();
    return view('book.show', [
        'book' => $book,
        'plans' => $plans,
        'authors' => $authors,
        'age_groups' => $age_groups,
    ]);
}
/**
 * Show the form for editing the specified resource.
 */
public function edit($id)
{
    $book = Book::find($id);
    $plans = Plan::orderBy('access_level', 'desc')->get();
    $authors = Author::all();
    $age_groups = AgeGroup::all();
```

```
    return view('book.edit', [
        'book' => $book,
        'plans' => $plans,
        'authors' => $authors,
        'age_groups' => $age_groups,
    ]);
}
/**
 * Update the specified resource in storage.
 */
public function update(Request $request, $id)
{
    if (Book::where('id', $id)->exists()) {
        $request->validate([
            'title' => 'required|max:255|min:3|unique:books,title,' . $id,
            'description' => 'required|min:5',
            'cover_url' => 'nullable|max:255',
            'read_time' => 'required|integer|min:1',
            'rating_medio' => 'nullable|numeric|between:0,5',
            'age_group' => 'required|max:50',
            'is_active' => 'required|between:0,1',
```

```
            'access_level' => 'required|integer|between:0,2',
        ]);
        $book = Book::find($id);
        $book->title = $request->title;
        $book->description = $request->description;
        $book->cover_url = $request->cover_url;
        $book->read_time = $request->read_time;
        $book->rating_medio = $request->rating_medio;
        $book->age_group = $request->age_group;
        $book->is_active = $request->is_active;
        $book->access_level = $request->access_level;
        $book->save();
                return redirect()->route('book.index')->with('success', 'Book updated successfully.');
      }
      return redirect()->route('book.index')->with('error', 'Book not found.');
  }
  /**
   * Remove the specified resource from storage.
   */
  public function destroy($id)
```

```
{
    $book = Book::findOrFail($id);
    // Delete associated PDF if exists
    if ($book->pdf_path && file_exists(public_path($book->pdf_path))) {
        unlink(public_path($book->pdf_path));
    }
    $book->delete();
    return redirect()->route('book.index')->with('success', 'Book deleted successfully!');
}
/**
 * Display the PDF file for a book.
 */
public function viewPdf($id)
{
    $book = Book::findOrFail($id);

    // Ensure the book has a valid PDF path
    if (!$book->pdf_path || !file_exists(public_path($book->pdf_path))) {
        abort(404, 'PDF not found');
    }
    return response()->file(public_path($book->pdf_path));
```

```
    }
}
```

ANEXO C: Implementação de rotas

```
<?php

use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\BookController;
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/
```

```
// Route::get('/', function () {
//     return view('welcome');
// });


Route::get('/books', [BookController::class, 'index'])->name('book.index');
Route::get('/book/{id}', [BookController::class, 'show'])->name('book.details');
Route::post('/book/store', [BookController::class, 'store'])->name('book.store');
Route::get('/book/pdf/{id}', [BookController::class, 'viewPdf'])->name('book.pdf');

// Routes principais
Route::get('/', 'App\Http\Controllers\StoreController@index')->name('store');
Route::get('/admin', 'App\Http\Controllers\AdminController@index')
->middleware(['auth', 'admin'])
->name('dashboard');

// Routes para páginas de administração
// Route::resource('admin/product', 'App\Http\Controllers\ProductController');
Route::resource('admin/book', 'App\Http\Controllers\BookController');
Route::resource('admin/author', 'App\Http\Controllers\AuthorController');
```

```
Route::get('/about', function () {
    return view('store.about');
})->name('about');

Route::get('/pricing', function () {
    return view('store.pricing');
})->name('pricing');

Route::get('/contact', function () {
    return view('store.contact');
})->name('contact');

Route::get('/terms', function () {
    return view('store.terms');
})->name('terms');

Auth::routes();
```

ANEXO D: Implementação do modelo de livros

```
<?php


namespace App\Models;


use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;


class Book extends Model

{

    use HasFactory;


    protected $guarded=['id'];

    protected $fillable = [

        'title',

        'description',

        'cover_url',

        'read_time',

        'rating_medio',

        'age_group',
```

```
        'is_active',
        'access_level',
        'pdf_path',
    ];


    public function authors()
{
    return $this->belongsToMany(Author::class, 'author_books');
}


}
```