Documentation Technique - Bookmarks.bio

Sommaire

Introduction

Bookmarks.bio est une application web permettant aux utilisateurs de créer et partager des pages personnalisées pour organiser leurs découvertes, conseils et inspirations sous forme de signets (Marks). Cette plateforme transforme les signets classiques en un espace interactif et créatif.

Ce document présente la documentation technique complète du projet, incluant son architecture, sa structure, et les différentes technologies utilisées.

Architecture du projet

Bookmarks.bio utilise une architecture MVC (Modèle-Vue-Contrôleur) personnalisée, basée sur PHP. Le projet est structuré comme suit:

- **Modèles** : Gestion des données et interactions avec la base de données
- **Vues** : Templates Twig pour le rendu des pages
- **Contrôleurs** : Logique métier et coordination des interactions
- **Router** : Gestion des routes et des URLs

Technologies principales

- **Backend** : PHP 8.2
- **Frontend** : Twig, TailwindCSS, JavaScript
- **Base de données** : MariaDB 10.6
- **Environnement** : Docker
- **Paiement** : Stripe

Environnement de développement

Le projet utilise Docker pour créer un environnement de développement unifié. Trois services principaux sont définis:

1. **app** : Serveur web Apache avec PHP 8.2
2. **database** : Serveur MariaDB pour la base de données
3. **phpmyadmin** : Interface d'administration pour la base de données

Configuration Docker

Le fichier `docker-compose.yml` définit l'ensemble des services:

```
services:
  app:
    build:
      context: docker
    ports:
      - "80:80"
    volumes:
      - .:/var/www/html
    depends_on:
      - database
    networks:
      - "dev-network"

  database:
    image: mariadb:10.6.15
    ports:
      - "3307:3306"
    environment:
```

```
      MARIADB_ROOT_PASSWORD: password
      MARIADB_DATABASE: bookmarks
    volumes:
      - mysqldata:/var/lib/mysql
      - ./docker/db/:/docker-entrypoint-initdb.d
    networks:
      - "dev-network"

  phpmyadmin:
    image: phpmyadmin
    environment:
      PMA_HOSTS: database
    ports:
      - "81:80"
    depends_on:
      - database
    networks:
      - "dev-network"
```

Démarrage de l'environnement

Pour lancer l'environnement de développement, exécutez:

```
docker-compose up -d
```

Pour arrêter les services:

```
docker-compose down
```

Structure des dossiers

Le projet est organisé selon la structure suivante:

```
bookmarks.bio/
├── app/                    # Dossier principal de l'application
│   ├── cache/              # Cache pour les templates Twig
│   ├── docker/             # Configuration Docker
│   ├── node_modules/       # Dépendances JavaScript
│   ├── public/             # Fichiers accessibles publiquement
│   ├── routes/             # Configuration des routes
│   ├── sql/                # Scripts SQL
│   ├── src/                # Code source de l'application
│   │   ├── Controller/     # Contrôleurs
│   │   ├── Model/          # Modèles
│   │   └── Router/         # Système de routage
│   ├── templates/          # Templates Twig
│   └── vendor/             # Dépendances PHP (Composer)
├── docs/                   # Documentation
│   ├── design/             # Documents de conception
│   └── ...
```

```
└── ...
```

Base de données

Schéma de la base de données

La base de données utilise MariaDB et comporte les tables principales suivantes:

- **users** : Informations sur les utilisateurs
- **roles** : Rôles des utilisateurs (Admin, Moderator, Member, Visitor)
- **statuses** : États possibles des utilisateurs (Muted, Banned)
- **collections** : Collections de marques créées par les utilisateurs
- **marks** : Signets/marques créés par les utilisateurs
- **categories** : Catégories pour classer les marks
- **comments** : Commentaires laissés sur les marks
- **subscriptions** et **invoices** : Gestion des abonnements premium
- **counters** et **countertypes** : Système de compteurs pour les marks (likes, etc.)

Diagramme ER

Le schéma complet de la base de données est disponible dans le fichier `docs/diagrame_architecture_bdd.png`.

Initialisation de la base de données

Le script SQL pour créer et initialiser la base de données se trouve dans `app/sql/database_creation.sql`.

Modèles

Les modèles gèrent l'accès aux données et les opérations liées à la base de données. Ils sont situés dans le dossier `app/src/Model/`.

Modèle de base (Model.php)

Tous les modèles héritent de la classe de base `Model` qui fournit les fonctionnalités communes:

- Connexion à la base de données
- Opérations CRUD génériques
- Validation des données

Principaux modèles

- **User.php** : Gestion des utilisateurs, authentification
- **Collection.php** : Gestion des collections de marks
- **Mark.php** : Gestion des signets/marks

- **Category.php** : Gestion des catégories
- **Subscription.php** : Gestion des abonnements premium
- **Invoice.php** : Gestion des factures
- **Role.php** : Gestion des rôles utilisateurs

Contrôleurs

Les contrôleurs gèrent la logique métier de l'application. Ils sont situés dans le dossier `app/src/Controller/`.

Contrôleur de base (Controller.php)

Tous les contrôleurs héritent de la classe `Controller` qui fournit:

- Accès aux services communs
- Méthodes pour rendre les vues
- Gestion des requêtes et réponses

Principaux contrôleurs

- **HomeController.php** : Page d'accueil
- **UserController.php** : Inscription, connexion, gestion des profils
- **CollectionController.php** : Création et gestion des collections
- **MySpaceController.php** : Espace utilisateur personnalisé
- **AdminController.php** : Gestion administrative
- **BillingController.php** : Gestion des paiements et abonnements
- **FeedController.php** : Flux social et découverte

Router

Le système de routage est situé dans `app/src/Router/`. Il est responsable de la correspondance entre les URLs et les contrôleurs appropriés.

Configuration des routes

Les routes sont définies dans le fichier JSON `app/routes/routes.json`. Chaque route spécifie:

- Le contrôleur à utiliser
- L'action à exécuter
- La méthode HTTP autorisée (GET, POST, etc.)
- Les middlewares à appliquer

Middlewares

Les middlewares sont utilisés pour les fonctionnalités transversales comme:

- **AuthMiddleware** : Vérification de l'authentification
- **PermissionMiddleware** : Contrôle des permissions
- **CollectionAccessMiddleware** : Vérification des droits d'accès aux collections

Vues

Le projet utilise Twig comme moteur de templates. Les templates sont situés dans le dossier `app/templates/`.

Structure des templates

- **layout/** : Templates de base et layouts communs
- **home/** : Templates pour la page d'accueil
- **auth/** : Templates d'authentification
- **myspace/** : Templates pour l'espace utilisateur
- **admin/** : Templates pour l'administration
- **billing/** : Templates pour la gestion des paiements
- **errors/** : Templates pour les pages d'erreur

Assets et CSS

Le projet utilise TailwindCSS pour les styles, configuré via le fichier `app/tailwind.config.js`.

Système d'authentification

L'authentification est gérée principalement par le `UserController` et le modèle `User`.

Fonctionnalités d'authentification

- Inscription et création de compte
- Connexion avec email et mot de passe
- Gestion des sessions
- Protection des routes via middleware d'authentification
- Contrôle des permissions basé sur les rôles

Système de paiement

La gestion des paiements est intégrée via Stripe et gérée par le `BillingController`.

Fonctionnalités de paiement

- Abonnements premium
- Traitement des paiements sécurisés

- Gestion des webhooks Stripe
- Facturation et historique des transactions

API et services externes

Le projet intègre plusieurs services externes:

- **Stripe** : Pour le traitement des paiements
- **PHPMailer** : Pour l'envoi d'emails
- **Parsedown** : Pour la conversion de Markdown en HTML

Déploiement

Pré-requis serveur

- PHP 8.2 ou supérieur
- MariaDB 10.6 ou supérieur
- Extensions PHP: intl, pdo_mysql, gd, fileinfo
- Serveur web (Apache recommandé)

Configuration de production

Pour un déploiement en production:

1. Cloner le dépôt
2. Installer les dépendances via Composer et NPM
3. Configurer les variables d'environnement
4. Initialiser la base de données
5. Configurer le serveur web