Université de Rouen Normandie
Faculté des Sciences et Techniques

Rapport du projet Cruciweb

Rapport technique sur le projet de langage web

Réalisé par :

- TAIBI Kamyl
- BERRAHAL Raid

Année Universitaire 2024-2025

Table des matières

1 Objet du projet

Ce projet a pour objectif de développer une application web nommée Cruciweb, dédiée à la résolution et à la création de grilles de mots croisés. Cette application vise à offrir une expérience interactive aux utilisateurs, en leur permettant non seulement de résoudre des grilles, mais aussi de proposer leurs propres créations et de les partager avec la communauté. Cruciweb se distingue par sa convivialité, sa fonctionnalité de sauvegarde de progression et son système de tri des grilles pour répondre aux différents niveaux et préférences des utilisateurs.

L'application est structurée en deux parties principales : la partie utilisateur et la partie administrateur, chacune ayant des rôles et des fonctionnalités distinctes.

Dans la partie utilisateur, l'utilisateur pourra s'inscrire sur la plateforme en créant un compte personnel. il pourra accéder aux grilles de mots croisés de la plateforme, qu'il pourra trier avec différents critères.Une fois connécté, L'application proposera un système de sauvegarde qui permettra à l'utilisateur de reprendre ses parties là où il les avait laissées. En outre, l'utilisateur aura la possibilité de soumettre ses propres grilles de mots croisés pour qu'elles soient partagées avec d'autres membres de la communauté.

La partie administrateur aura pour rôle principal la gestion des utilisateurs et du contenu de la plateforme. L'administrateur pourra ajouter ou supprimer des comptes utilisateurs. Il disposera également de droits étendus sur le contenu en permettant la suppression de grilles, sans être obligé de participer à la création ou à la résolution des grilles.

En somme, Cruciweb cherche à offrir une plateforme complète et dynamique, combinant divertissement et créativité, tout en garantissant une gestion fluide du contenu et des utilisateurs.

2 Technologies utilisées

Pour le développement de l'application *Cruciweb*, plusieurs technologies ont été utilisées afin de garantir une interface utilisateur fluide et une gestion efficace des données.

2.1 Frontend

Le frontend de l'application a été conçu à l'aide des technologies suivantes :

- **HTML** : utilisé pour structurer les pages web.
- **CSS** : pour la mise en forme et le design des interfaces.
- **JavaScript** : pour rendre les pages dynamiques et utiliser ajax en JQuery afin de faire des requetes HTTP.

2.2 Backend

La partie serveur de l'application a été développée avec **PHP**. Ce choix a été fait pour sa simplicité et son intégration facile avec des bases de données SQL. Bien que l'on ait pu opter pour **JEE**, **PHP** a été privilégié pour sa capacité à répondre efficacement aux besoins du projet notamment pour la simplicité à l'adaptation à l'architecture mvc.

2.3 Base de données

Le stockage et la gestion des données sont assurés par une base de données **SQL**, permettant une manipulation optimisée des informations via des requêtes structurées.

2.4 Résumé des technologies

L'application *Cruciweb* repose sur un ensemble de technologies robustes et éprouvées. L'utilisation combinée de **HTML**, **CSS**, **JavaScript**, **PHP**, et **SQL** garantit à la fois une interface utilisateur dynamique et une gestion efficace des données côté serveur.

3 Architecture générale du projet

L'architecture utilisée pour le développement de l'application est le modèle MVC (Model-View-Controller). L'utilisateur interagit en premier lieu avec la landing page, qui est une vue rendue par défaut par le routeur `index.php`. Ensuite, lorsque l'utilisateur effectue une interaction dans la vue, cette dernière est envoyée au routeur sous forme de requête HTTP, avec une action et des paramètres optionnels (GET ou POST) transmis par AJAX en JavaScript.

Le routeur agit en fonction du type d'action et des paramètres, et choisit le contrôleur correspondant. Ce dernier va, à son tour, interagir avec le modèle, qui interroge la base de données selon la demande du contrôleur, à l'aide d'une requête SQL. Le modèle obtient le résultat de la base de données et renvoie les données au contrôleur sous la forme appropriée, soit en JSON, soit sous forme de tableau. Enfin, le contrôleur renvoie ces données accompagnées d'une vue (rendu), qui sera affichée pour l'utilisateur sous forme de HTML.

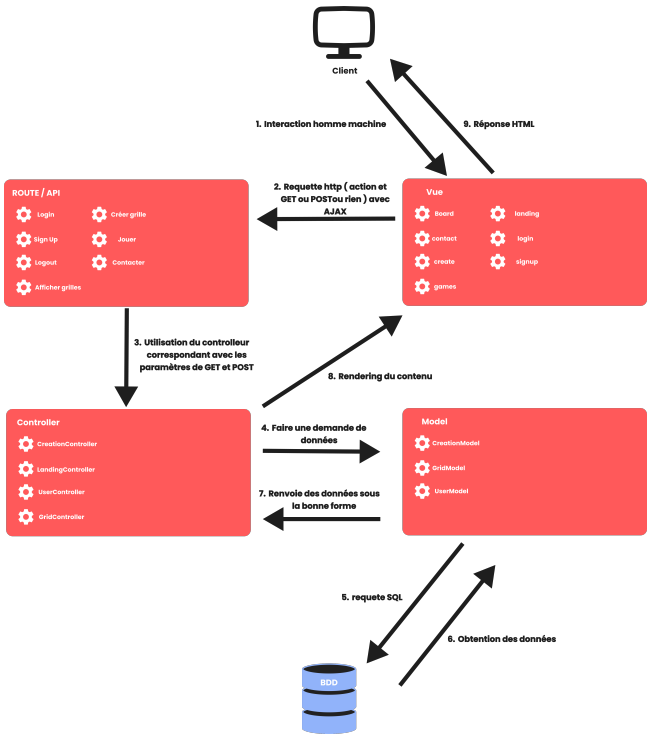

Figure 1: Architecture générale de l'application

4 Aspects Ergonomiques

Afin de garantir une interface utilisateur intuitive et ergonomique, nous avons opté pour l'utilisation de l'outil de design *Figma*. Cette approche nous a permis de planifier l'interface en amont et d'itérer facilement sur les différents prototypes avant le développement, ce qui a assuré une expérience utilisateur optimale. *Figma* a également facilité la collaboration entre les membres de l'équipe en offrant une plateforme centralisée pour les retours et ajustements.

La première étape a consisté à créer un logo intuitif et représentatif d'une application de mots croisés. Ce logo devait à la fois refléter l'aspect ludique du jeu et l'aspect cérébral des mots croisés. Ensuite, un choix de couleurs a été fait pour garantir un confort visuel. Le orange a été choisi pour les boutons et titres importants, afin de capturer l'attention de l'utilisateur sans être agressif. Le bleu clair a été sélectionné pour les boutons moins importants, permettant une différenciation subtile, tandis que le noir a été utilisé pour les éléments textuels et les fonds, créant ainsi une hiérarchie visuelle claire.

Ces couleurs ont été choisies non seulement pour leur efficacité en termes d'ergonomie mais aussi pour évoquer l'aspect à la fois ludique et stimulant intellectuellement de l'application.

Une landing page a été conçue avec un slogan captivant, pour renforcer l'attrait de l'application dès le premier contact. Cette page d'accueil comprend également des boutons intuitifs, guidant l'utilisateur vers les fonctionnalités principales avec une navigation fluide. Le design des différentes cartes, boutons et types de police a été pensé en respectant le confort visuel, en évitant une surcharge d'informations et en privilégiant une hiérarchie visuelle claire.

Enfin, pour garantir l'accessibilité, des choix typographiques ont été faits en tenant compte des tailles de police adaptées et des contrastes suffisants pour les utilisateurs ayant des difficultés visuelles. Tout cela a été réalisé dans le but de simplifier l'utilisation tout en offrant un design attractif et fonctionnel.

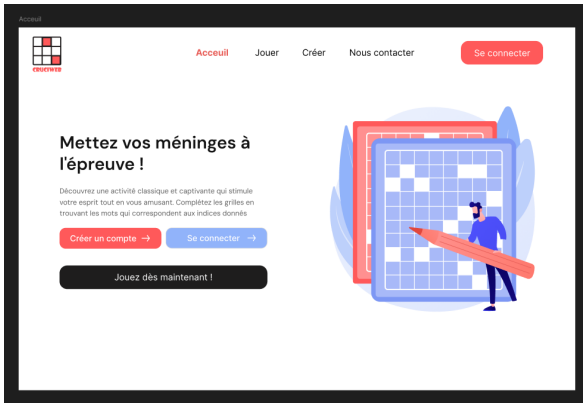

Figure 2: Landing page

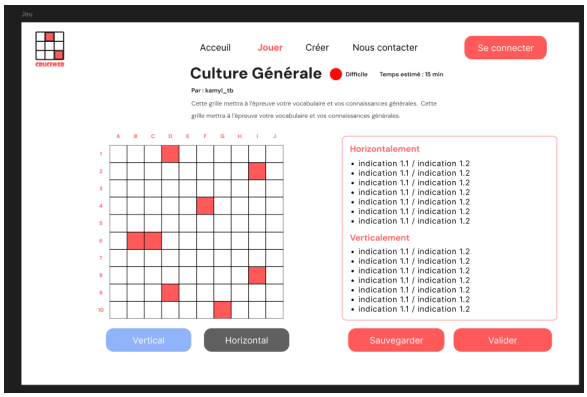

Figure 3: Jouer dans une grille

Figure 4: Design de l'application sur Figma

5 Configuration Générale du Projet

La configuration générale du projet est essentielle pour assurer un déploiement et une exécution sans heurts de l'application. Voici les étapes et les fichiers clés impliqués dans la configuration du projet.

5.1 Fichier `.env`

Le fichier `.env` contient les configurations sensibles et spécifiques à l'environnement, telles que les informations de connexion à la base de données. Voici un exemple de fichier `.env` :

```
# Base de données
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_NAME=cruciweb
DB_USER=root
DB_PASSWORD=''
DB_USER_ADMIN=root
DB_PASSWORD_ADMIN=''
```

Ce fichier doit être placé à la racine du projet et ne doit pas être inclus dans le contrôle de version pour des raisons de sécurité.

5.2 Configuration de la Base de Données

La configuration de la base de données est gérée par le fichier `Database.php` situé dans le répertoire `config`. Ce fichier utilise les variables d'environnement définies dans le fichier `.env` pour établir une connexion à la base de données.

```
<?php
namespace App\Config;
use PDO;
use PDOException;

class Database {
    private $pdo;

    public function __construct() {
        $host = $_ENV['DB_HOST'];
        $dbname = $_ENV['DB_NAME'];
        $user = $_ENV['DB_USER'];
        $password = $_ENV['DB_PASSWORD'];
        $this->connect($host, $dbname, $user, $password);
    }

    private function connect($host, $dbname, $user, $password) {
        try {
            $this->pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $user,
            $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
```

```
            die("Connection failed: " . $e->getMessage());
        }
    }

    public function getPDO() {
        return $this->pdo;
    }
}
?>
```

5.3 Autoloader

Le fichier `index.php` à la racine du projet configure l'autoloader pour charger automatiquement les classes nécessaires. Il utilise la fonction `spl_autoload_register` pour inclure les fichiers de classes situés dans les répertoires `Controllers`, `Models`, et `Views`.

```
<?php
spl_autoload_register(function ($class) {
    $path = 'app/';
    $directories = ['Controllers', 'Models', 'Views'];

    foreach ($directories as $dir) {
        $file = $path . $dir . '/' . $class . '.php';
        if (file_exists($file)) {
            require_once $file;
        }
    }
});

?>
```

5.4 Fichier `.htaccess`

Le fichier `.htaccess` est utilisé pour rediriger toutes les requêtes vers `public/index.php`, permettant ainsi un routage centralisé.

```
# Activer mod_rewrite
RewriteEngine On

# Rediriger toutes les requêtes vers public/index.php
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ public/index.php?url=$1 [QSA,L]
```

5.5 Installation des Dépendances

Les dépendances PHP nécessaires sont gérées par Composer.

Si Composer sera installé avec le script de déploiement. Une fois installé, la commande suivante sera exécuté dans le répertoire racine du projet pour installer les dépendances spécifiées dans le fichier `composer.json` :

```
composer install
```

6 Définition des APIs Utilisées dans l'Application

L'application utilise plusieurs APIs pour gérer les interactions entre le client et le serveur. Voici une description formelle des principales APIs utilisées :

6.1 API d'Authentification

6.1.1 Inscription (Sign Up)

- **URL** : `index.php?action=signup`
- **Méthode** : POST
- **Description** : Permet à un nouvel utilisateur de créer un compte.
- **Paramètres** :
  - `username` : Nom d'utilisateur
  - `fullname` : Nom complet
  - `password` : Mot de passe (haché avant d'être stocké)
  - `email` : Adresse email
- **Réponse** :
  - Succès : `{"success": true, "message": "Compte créé avec succès."}`
  - Erreur : `{"error": "Cet utilisateur existe déjà."}`

6.1.2 Connexion (Login)

- **URL** : `index.php?action=login`
- **Méthode** : POST
- **Description** : Permet à un utilisateur existant de se connecter.
- **Paramètres** :
  - `username` : Nom d'utilisateur
  - `password` : Mot de passe
- **Réponse** :
  - Succès : `{"success": true, "message": "Connexion réussie."}`
  - Erreur : `{"error": true, "message": "Login ou mot de passe incorrect."}`

6.1.3 Déconnexion (Logout)

- **URL** : `index.php?action=logout`
- **Méthode** : GET
- **Description** : Permet à un utilisateur de se déconnecter.
- **Réponse** : Redirection vers la page de connexion.

6.2 API de Gestion des Grilles

6.2.1 Création de Grille

- **URL** : `index.php?action=create`
- **Méthode** : POST
- **Description** : Permet à un utilisateur de créer une nouvelle grille de mots croisés.
- **Paramètres** :
  - `gridData` : Données de la grille (nom, difficulté, dimensions, cases noires, indices, solutions)
- **Réponse** :
  - Succès : `{"message": "Grille créée avec succès."}`

6.2.2 Sauvegarde de Progrès

- **URL** : `index.php?action=saved`
- **Méthode** : POST
- **Description** : Permet à un utilisateur de sauvegarder ses progrès sur une grille.
- **Paramètres** :
  - `gridId` : ID de la grille
  - `gridData` : Données de la grille (progrès de l'utilisateur)
- **Réponse** :
  - Succès : `{"message": "Progrès sauvegardés avec succès."}`

6.2.3 Affichage des Grilles

- **URL** : `index.php?action=grids`
- **Méthode** : GET
- **Description** : Permet d'afficher les grilles disponibles.
- **Paramètres** : Aucun
- **Réponse** : Liste des grilles disponibles au format JSON.

6.3 API de Gestion des Utilisateurs (Admin)

6.3.1 Suppression d'Utilisateur

- **URL** : `index.php?action=delete`$_{user}$&$username = \{username\}$`Méthode` : GET
- **Description** : Permet à un administrateur de supprimer un utilisateur.
- **Paramètres** :
  - `username` : Nom d'utilisateur à supprimer
- **Réponse** :
  - Succès : `{"message": "Utilisateur supprimé avec succès."}`

6.3.2 Suppression de Grille

- **URL** : `index.php?action=delete`$_{grid}$&$gridId = \{gridId\}$`Méthode` : GET
- **Description** : Permet à un administrateur de supprimer une grille.
- **Paramètres** :
  - `gridId` : ID de la grille à supprimer
- **Réponse** :
  - Succès : `{"message": "Grille supprimée avec succès."}`

7 Sécurité et anti-triche

La sécurité est un aspect crucial de notre application. Plusieurs mesures ont été mises en place pour garantir la protection des données des utilisateurs et l'intégrité des fonctionnalités. Ces pratiques couvrent le hachage des mots de passe, la gestion sécurisée des solutions de grilles de mots croisés, ainsi que les restrictions d'accès à la base de données.

7.1 Hachage des Mots de Passe

Les mots de passe des utilisateurs ne sont jamais stockés en clair dans la base de données. Au lieu de cela, chaque mot de passe est haché avec un algorithme sécurisé avant d'être sauvegardé. Nous utilisons la fonction `password_hash` de PHP, qui applique un sel unique à chaque mot de passe avant de le hacher. Cela rend les mots de passe beaucoup plus difficiles à craquer, même en cas de fuite de la base de données.

Exemple de code pour le hachage des mots de passe :

```
$passwordHash = password_hash($password, PASSWORD_BCRYPT);
```

7.2 Hachage des Solutions

Pour éviter la triche, les solutions des grilles de mots croisés sont également hachées avant d'être stockées dans la base de données. Cela garantit que, même si quelqu'un accède à la base de données, il ne pourra pas voir les solutions en clair. Nous utilisons l'algorithme SHA-256 pour hacher les solutions, ce qui garantit une forte sécurité.

Exemple de code pour le hachage des solutions :

```
$hashedSolution = hash('sha256', $solution);
```

7.3 Restrictions d'Accès à la Base de Données

Les utilisateurs normaux ont des permissions limitées sur la base de données. Ils peuvent uniquement consulter et insérer des données dans les tables, à l'exception de la table des sauvegardes de progrès, où ils peuvent également mettre à jour leurs progrès. Cela limite les risques de modifications non autorisées des données critiques.

Les administrateurs, en revanche, ont des permissions étendues qui leur permettent de gérer les utilisateurs et les grilles de mots croisés.

Exemple de configuration des permissions :

```
-- Permissions pour les utilisateurs normaux
GRANT SELECT, INSERT ON cruciweb.* TO 'user'@'localhost';
GRANT UPDATE ON cruciweb.sauvegarde TO 'user'@'localhost';

-- Permissions pour les administrateurs
GRANT ALL PRIVILEGES ON cruciweb.* TO 'admin'@'localhost';
```

8 Déploiement

Une notice de déploiement détaillant les étapes nécessaires à l'installation et à la configuration de l'application sera attachée au projet. Cette notice permettra une mise en production fluide et sans problème.

9 Conclusion

Le projet Cruciweb a été conçu et déployé tout en respectant les contraintes technologiques initiales. Les fonctionnalités principales ont été mises en œuvre avec succès, et il serait envisageable de développer une version 2.0 plus robuste. Cette future version pourrait inclure la création d'une application dédiée pour l'administrateur, afin d'améliorer à la fois la sécurité et la logique du système.

Malgré les avancées réalisées, la page de contact n'a pas pu être développée en raison du manque de temps. Cependant, cette fonctionnalité pourrait significativement enrichir l'interactivité de l'application, notamment en facilitant la communication avec les utilisateurs de Cruciweb. Par ailleurs, il serait judicieux d'ajouter une fonctionnalité permettant à l'administrateur de consulter les solutions des grilles avant leur suppression, pour une meilleure gestion du contenu.

L'aspect sécurité a été favoritisé dans l"application afin de garantir une expérience optimale et sans triches.