Guide du Développeur 1

Core & Authentification

Tes responsabilités :

- Architecture MVC du projet
- Système de connexion/déconnexion
- Sécurité de l'application
- Configuration base de données

Table des matières

1 Ton rôle dans l'équipe

Tu es le **pilier technique** de l'équipe. Sans ton travail, personne ne peut avancer car :

— Dev 2 a besoin de ta classe `Database` pour accéder aux données

— Dev 3 a besoin de ton `Router` pour créer les pages

— Dev 4 a besoin de ton module `User` pour la gestion des utilisateurs

Important

Tu dois terminer l'architecture de base et l'authentification **en priorité** pour débloquer l'équipe.

2 Tâche 1 : Créer l'architecture MVC

2.1 C'est quoi le MVC ?

Concept

MVC = Model - View - Controller. C'est une façon d'organiser le code :

— **Model** : Gère les données (requêtes SQL, logique métier)

— **View** : Affiche les pages HTML à l'utilisateur

— **Controller** : Fait le lien entre les deux (reçoit les requêtes, appelle les models, renvoie les views)

2.2 Structure des dossiers à créer

Tu dois créer cette organisation de fichiers :

```
GestioSeances/
 app/
    controllers/      → Les contrôleurs (logique)
    models/           → Les modèles (base de données)
    views/            → Les vues (HTML/PHP)
    core/             → Classes utilitaires
 public/
    index.php         → Point d'entrée unique
    css/              → Fichiers CSS
    js/               → Fichiers JavaScript
 storage/
    uploads/          → Fichiers uploadés
 database/
    schema.sql        → Script création BDD
 config.php            → Configuration
```

Tâche

À faire :

1. Créer tous ces dossiers sur ton ordinateur
2. Créer un fichier `config.php` à la racine avec les paramètres de connexion BDD
3. Créer le fichier `public/index.php` qui sera le point d'entrée

2.3 Le fichier config.php

Ce fichier doit contenir un tableau PHP avec :

— Les informations de connexion à MySQL (host, nom BDD, user, password)

— Les paramètres email pour PHPMailer (host SMTP, port, identifiants)

— Les paramètres de l'application (nom, URL, mode debug)

— Les paramètres de sécurité (nombre max de tentatives login, durée de blocage)

2.4 Le point d'entrée unique (index.php)

Concept

Front Controller Pattern : Toutes les requêtes passent par un seul fichier (`index.php`). C'est lui qui décide quel contrôleur appeler selon l'URL.

Le fichier `public/index.php` doit :

1. Charger automatiquement les classes (autoloader)
2. Démarrer la session PHP
3. Créer une instance du Router
4. Définir toutes les routes de l'application
5. Appeler la méthode pour résoudre la route actuelle

2.5 Le fichier .htaccess

Tu dois créer un fichier `public/.htaccess` qui redirige toutes les URLs vers `index.php`. Cherche sur Google : "htaccess rewrite to index.php".

3 Tâche 2 : Créer les classes Core

Ces classes sont dans le dossier `app/core/` et sont utilisées partout dans l'application.

3.1 Classe Database.php

Tâche

Objectif : Créer une classe qui gère la connexion à MySQL avec PDO.
Pattern à utiliser : Singleton (une seule instance de connexion)
Méthodes à créer :

- `getInstance()` : Retourne l'instance unique de Database
- `getConnection()` : Retourne l'objet PDO pour faire des requêtes

Concepts à rechercher :

- "PHP PDO connection"
- "PHP Singleton pattern"
- "PDO options ERRMODE EXCEPTION"

3.2 Classe Router.php

Tâche

Objectif : Créer un routeur qui analyse l'URL et appelle le bon contrôleur.
Méthodes à créer :

- `get($path, $controller, $method)` : Ajouter une route GET
- `post($path, $controller, $method)` : Ajouter une route POST
- `resolve()` : Analyser l'URL actuelle et appeler le bon contrôleur

Comment ça marche :

1. Le routeur stocke toutes les routes dans un tableau
2. Quand on appelle `resolve()`, il récupère l'URL actuelle
3. Il cherche si cette URL existe dans son tableau
4. Si oui, il instancie le contrôleur et appelle la méthode
5. Si non, il affiche une erreur 404

3.3 Classe Session.php

Tâche

Objectif : Gérer les sessions PHP de manière sécurisée.
Méthodes à créer :

- `start()` : Démarrer la session avec options sécurisées
- `set($key, $value)` : Stocker une valeur en session
- `get($key)` : Récupérer une valeur
- `has($key)` : Vérifier si une clé existe
- `remove($key)` : Supprimer une valeur
- `destroy()` : Détruire complètement la session
- `regenerate()` : Régénérer l'ID de session
- `isLoggedIn()` : Vérifier si l'utilisateur est connecté
- `setFlash($type, $message)` : Message flash (affiché une seule fois)
- `getFlash($type)` : Récupérer et supprimer le message flash

Concepts à rechercher :

- "PHP session security best practices"
- "session_regenerate_id"
- "PHP flash messages"

4 Tâche 3 : Créer le module Sécurité

4.1 Classe Security.php

Cette classe contient toutes les fonctions de sécurité de l'application.

Tâche

Méthodes à créer :

1. **Protection CSRF**
   - `generateCsrfToken()` : Génère un token aléatoire et le stocke en session
   - `verifyCsrfToken($token)` : Vérifie si le token soumis correspond à celui en session
   - `csrfField()` : Génère le HTML du champ caché à mettre dans les formulaires
2. **Protection XSS**
   - `escape($string)` : Échappe les caractères HTML dangereux
3. **Mots de passe**
   - `hashPassword($password)` : Hashe un mot de passe avec bcrypt
   - `verifyPassword($password, $hash)` : Vérifie si le mot de passe correspond
4. **Tokens**
   - `generateToken()` : Génère un token aléatoire (pour reset password)
5. **Brute Force**
   - `isAccountLocked($attempts, $lockTime)` : Vérifie si le compte est bloqué

À savoir

Concepts à rechercher :

- "PHP CSRF protection tutorial"
- "password_hash bcrypt PHP"
- "htmlspecialchars PHP"
- "random_bytes PHP"

4.2 Comprendre les attaques

Attaque	Comment s'en protéger
CSRF	Token unique dans chaque formulaire, vérifié côté serveur
XSS	Échapper toutes les sorties avec `htmlspecialchars()`
SQL Injection	Utiliser les requêtes préparées PDO (jamais de concaténation)
Brute Force	Bloquer le compte après X tentatives échouées
Vol de session	Régénérer l'ID après login, cookies HttpOnly

5 Tâche 4 : Créer le modèle User

5.1 Fichier app/models/User.php

Tâche

Objectif : Gérer toutes les opérations sur la table `utilisateurs`.
Méthodes à créer :

- `findByEmail($email)` : Chercher un utilisateur par son email
- `findById($id)` : Chercher un utilisateur par son ID
- `create($data)` : Créer un nouvel utilisateur
- `updateLoginAttempts($userId, $attempts, $lock)` : Mettre à jour les tentatives de connexion
- `resetLoginAttempts($userId)` : Remettre à zéro après connexion réussie
- `saveResetToken($userId, $token)` : Sauvegarder le token de reset password
- `findByResetToken($token)` : Trouver un utilisateur par son token de reset
- `updatePassword($userId, $password)` : Mettre à jour le mot de passe

Important

Règles importantes :

- Toujours utiliser les requêtes préparées PDO
- Jamais stocker le mot de passe en clair (toujours hasher)
- Les tokens de reset doivent avoir une date d'expiration

6 Tâche 5 : Créer le contrôleur Auth

6.1 Fichier app/controllers/AuthController.php

Tâche

Méthodes à créer :

1. **showLogin()**
   - Si déjà connecté → rediriger vers dashboard
   - Sinon → afficher la vue login.php
2. **login()**
   - Vérifier le token CSRF
   - Valider que email et password ne sont pas vides
   - Chercher l'utilisateur par email
   - Vérifier si le compte est bloqué
   - Vérifier si le compte est actif
   - Vérifier le mot de passe
   - Si échec → incrémenter les tentatives
   - Si succès → régénérer la session, stocker user_id et role, rediriger
3. **logout()**
   - Détruire la session
   - Rediriger vers login
4. **showForgotPassword()**
   - Afficher le formulaire de demande de reset
5. **forgotPassword()**
   - Vérifier le token CSRF
   - Chercher l'utilisateur par email
   - Générer un token et le sauvegarder
   - Envoyer l'email avec le lien de reset
   - Toujours afficher le même message (sécurité)
6. **showResetPassword()**
   - Vérifier que le token est valide et non expiré
   - Afficher le formulaire de nouveau mot de passe
7. **resetPassword()**
   - Vérifier que les mots de passe correspondent
   - Vérifier la longueur minimum (8 caractères)
   - Mettre à jour le mot de passe
   - Supprimer le token
   - Rediriger vers login

7 Tâche 6 : Créer les vues d'authentification

7.1 Fichiers à créer

- `app/views/auth/login.php`
- `app/views/auth/forgot-password.php`
- `app/views/auth/reset-password.php`

7.2 Contenu de login.php

Tâche

La page de login doit contenir :

- Le logo ou nom de l'application
- Un formulaire avec :
  - Champ caché CSRF token
  - Champ email (type="email", required)
  - Champ mot de passe (type="password", required)
  - Checkbox "Se souvenir de moi" (optionnel)
  - Bouton "Se connecter"
- Lien "Mot de passe oublié ?"
- Zone pour afficher les messages d'erreur/succès (flash messages)

Conseil : Utilise Bootstrap pour le design, c'est plus rapide.

8 Tâche 7 : Créer le schéma de base de données

8.1 Fichier database/schema.sql

Tu dois créer le script SQL qui crée toutes les tables. Même si certaines tables seront utilisées par d'autres devs, c'est toi qui crées le schéma complet.

Tâche

Tables à créer :

1. **utilisateurs** : id, nom, prenom, email, mot_de_passe, role, telephone, bureau, actif, tentatives_connexion, date_blocage, token_reset, token_expiration, derniere_connexion, date_creation
2. **matieres** : id, code, nom, description
3. **salles** : id, nom, batiment, capacite, equipements, disponible
4. **seances** : id, professeur_id, matiere_id, salle_id, date_seance, heure_debut, heure_fin, groupe, type_seance
5. **demandes** : id, numero, professeur_id, seance_id, type_demande, nouvelle_date, nouvelle_heure_debut, nouvelle_heure_fin, nouvelle_salle_id, motif, statut, date_soumission, date_creation
6. **pieces_jointes** : id, demande_id, nom_original, nom_stockage, type_mime, taille, chemin, date_upload
7. **validations** : id, demande_id, validateur_id, action, commentaire, date_action
8. **notifications** : id, utilisateur_id, demande_id, type, titre, message, lue, date_creation
9. **parametres** : id, cle, valeur, description
10. **logs_activite** : id, utilisateur_id, action, table_concernee, enregistrement_id, details, adresse_ip, user_agent, date_action

À savoir

N'oublie pas :

- Les clés primaires AUTO_INCREMENT
- Les clés étrangères (FOREIGN KEY)
- Les index sur les colonnes souvent recherchées
- Les valeurs par défaut
- Le charset utf8mb4

9 Planning suggéré

Tâches
Structure des dossiers, config.php, Database.php
Router.php, index.php, .htaccess
Session.php, schema.sql
Security.php
User.php (modèle)
AuthController.php (login/logout)
Forgot/Reset password
Tests, debug, support à l'équipe

10 Checklist finale

Avant de dire que tu as terminé, vérifie :

- [ ] Structure MVC créée correctement
- [ ] Connexion à la BDD fonctionne
- [ ] Router redirige vers les bons contrôleurs
- [ ] Login fonctionne (email + mot de passe)
- [ ] Logout détruit la session
- [ ] Redirection selon le rôle après login
- [ ] Mot de passe oublié envoie un email
- [ ] Reset password fonctionne
- [ ] Mots de passe hashés avec bcrypt
- [ ] Token CSRF sur tous les formulaires
- [ ] Blocage après 5 tentatives de connexion
- [ ] Script schema.sql crée toutes les tables

Bon courage Jaafar !

L'équipe compte sur toi.