Documentation Technique – Application Cinéphoria

1. **Présentation générale**

Cinéphoria est une application multiplateforme (web, mobile, bureautique) de gestion de séances et de réservation de places de cinéma. Elle comprend :

- Une **interface web** pour les visiteurs, employés et administrateurs.
- Une **application mobile Flutter** pour les utilisateurs finaux.
- Une **application bureautique Tkinter** pour les employés techniques.

L'architecture globale repose sur un backend PHP/MVC connecté à une base de données MySQL (relationnelle) et MongoDB (analytique), avec une API REST pour la communication avec les autres clients.

2. Architecture générale

- **Frontend Web** : HTML5, CSS3, SCSS, Bootstrap, JavaScript, jQuery
- **Backend Web/API** : PHP 7+, Architecture MVC
- **Mobile App** : Flutter (Dart), communication via HTTP avec API
- **Desktop App** : Tkinter (Python), accès direct à MySQL
- **Base de données** :
    - **MySQL** : utilisateurs, films, séances, réservations

MongoDB : logs, statistiques, analyse

3. Structure des répertoires (Web)

```
bash
CopierModifier
/cinephoria
├── index.php
├── /controllers
│   └── ReservationController.php
├── /models
│   └── Reservation.php
├── /views
│   ├── accueil.php
│   ├── reservation.php
├── /public
│   ├── js/
│   ├── css/
│   └── images/
├── /api
│   ├── get_reservation.php
│   └── login_user.php
```

4. **Base de données MySQL**

Principales tables :

- `utilisateur (id, nom, email, mot_de_passe, role)`
- `film (id, titre, description, affiche, date_sortie)`
- `salle (id, numero, capacite)`
- `seance (id, film_id, salle_id, date_heure, qualite)`
- `reservation (id, user_id, seance_id, nb_places)`

`incident (id, salle, description, date_incident)`

5. API REST (extrait mobile)

- **Base URL** : `http://10.0.2.2/Cinephoria/Api`

Exemple d'appel :

```
dart
CopierModifier
final response = await http.get(
  Uri.parse('$baseUrl/get_reservation.php?user_id=${widget.userId}')
);
```

Réponse attendue (JSON) :

```
json
CopierModifier
{
  "reservations": [
    {
      "film": "Inception",
      "date": "2025-07-20 20:00",
      "salle": "2",
      "siege": "A5",
      "qr_code": "base64data"
    }
  ]
}
```

6. Application bureautique (Tkinter)

- Connexion directe à MySQL via `mysql.connector`
- Authentification employé
- Déclaration d'incidents par salle
- Visualisation des incidents enregistrés

`Interface graphique avec tk, ttk, Text, Listbox, etc.`

Exemple :

```
python
CopierModifier
cursor.execute(
  "INSERT INTO incidents (salle, description, date_incident) VALUES (%s, %s,
%s)",
  (salle, description, date_incident)
)
```

7. Sécurité

- Utilisation de **requêtes préparées** (PHP & Python)
- Mots de passe **hachés avec** `password_hash()`
- **Validation côté client** + **filtrage côté serveur**
- Protection contre :
    - **Injection SQL**
    - **XSS**
    - **Session Hijacking**

`Stockage sécurisé des tokens d'authentification (mobile)`

8. Tests effectués

Type de test	Outils	Objectif
Fonctionnels	Manuel, navigateur	Vérifier le bon comportement des modules
Sécurité	Scripts PHP, Postman	Tester XSS, injections SQL
API	Postman, Flutter	Vérifier les réponses et erreurs
Responsive design	Navigateur dev tools	Adaptabilité mobile/tablette
Desktop	Tkinter (live)	Saisie, affichage, enregistrement incidents

9. Difficultés techniques rencontrées

- Gestion des communications entre **API et Flutter**
- Synchronisation temps réel des réservations
- Adaptation multi-écrans (responsive + desktop + mobile)

`Accès sécurisé aux modules sensibles (employé/admin)`

10. Perspectives d'évolution

- Ajout de paiement en ligne sécurisé

`Intégration de` **notifs push** `sur mobile`
`Export PDF des billets depuis le site ou l'app`

- Mode hors-ligne pour l'app mobile