DOCUMENTATION TECHNIQUE

SOIGNE-MOI

TABLES DES MATIERES

REFLEXIONS INITIALES TECHNOLOGIQUE SUR LE SUJET

1.1. Introduction

Ce chapitre vise à présenter les réflexions technologiques initiales qui ont guidé notre projet depuis sa phase de conception. Les choix technologiques jouent un rôle crucial dans le développement d'une application, car ils affectent directement sa performance, sa maintenabilité et sa capacité à répondre aux besoins des utilisateurs.

1.2. Choix des technologies

1.2.1. Backend

Pour le développement du backend de notre application, nous avons opté pour une combinaison de technologies qui répondent aux exigences spécifiques de notre projet. Ces technologies comprennent :

- Python Flask : Nous avons choisi Python Flask comme Framework en raison de sa simplicité, de sa flexibilité. Il est idéal pour la création d'API RESTful, ce qui correspondait à nos besoins de communication entre le frontend et le backend.
- MySQL : Nous avons selectionné MySQL comme système de gestion de base de données pour sa stabilité, sa performance et sa prise en charge de requêtes complexes. Il est largement utilisé dans l'industrie et s'intègre bien avec Python Flask.

1.2.2. Frontend

Le frontend de notre application est construite en utilisant le Framework React via NodeJs, un choix qui découle de plusieurs considérations :

- React : Nous avons choisi React pour son approche basée sur les composants, qui permet une organisation modulaire du code et une réutilisation facile des éléments d'interface utilisateur.
- Material UI : Pour garantir une interface utilisateur cohérente et attrayante, nous avons intégré Matérial UI, une bibliothèque de composants React basée sur les directives de conception de Google Material Design.

1.2.3. Application de Bureau

Nous avons développé une application de bureau en utilisant JAVAFX pour répondre aux besoins de nos utilisateurs qui souhaitent accéder à l'application depuis leur ordinateur de bureau. Les raisons derrière ce choix incluent :

- JAVAFX : JAVAFX offre une interface native et une grande compatibilité multiplateforme, ce qui signifie que notre application de bureau fonctionne de manière transparente sur divers systèmes d'exploitation notamment Windows, macOS et Linux.

1.2.4. Application Mobile

En ce qui concerne l'application mobile, nous avons opté pour le Framework Flutter de Google, car il présente plusieurs avantages :

- Flutter : Flutter permet de développer des applications mobiles iOS et Android avec un seul code source, ce qui réduit considérablement le temps et les efforts nécessaires pour prendre en charge plusieurs plateformes. De plus, Flutter offre une grande flexibilité pour créer des interfaces utilisateur personnalisées et interactives.

1.3. Justification des choix technologiques

1.3.1. Backend

Le choix de Python Flask pour le backend a été guidé par sa simplicité et sa rapidité de développement. De plus, Python possède une bibliothèque riche et est bien adapté pour la manipulation de données, ce qui est essentiel pour notre application de gestion médicale. MySQL, en tant que système de gestion de base de données fiable, permet de stocker efficacement les informations médicales des utilisateurs.

1.3.2. Frontend

React et Material UI ont été choisi pour offrir une expérience utilisateur fluide et moderne. La modularité de React nous permet de développer des composants réutilisables, tandis que Material-UI garantit une interface utilisateur esthétique et conforme aux normes de conception actuelles.

1.3.3. Application de Bureau

JavaFX a été privilégié pour son aspect natif, sa facilité d'utilisation et sa compatibilité multiplateforme. Il assure une expérience utilisateur familière et performante sur les ordinateurs de bureau.

1.3.4. Application Mobile

Flutter nous permet de créer des applications mobiles rapide et réactives avec un seul code source, ce qui réduit la complexité du développement et les coûts de maintenance à long terme. De plus, Flutter offre une flexibilité inégalée pour personnaliser l'interface utilisateur grâce à SceneBuilder.

1.4. Conclusion

Les réflexions technologiques initiales ont joué un rôle fondamental dans la création de notre application de gestion médicale. Les choix technologiques judicieux ont permis de garantir que notre application réponde aux besoins de nos utilisateurs de manière efficace, esthétique et fiable. Ces choix ont jeté les bases du développement réussi de notre projet.

CONFIGURATION DE L'ENVIRONNEMENT DE TRAVAIL

2.1. Introduction

Ce chapitre décrit les étapes nécessaires pour configurer votre environnement de travail afin de commencer à travailler avec notre application. Il comprend des instructions détaillées sur la configuration des outils, des dépendances et des prérequis pour le développement et l'utilisation de l'application.

2.2. Configuration du Backend

2.2.1. Python et Flask

Pour travailler sur le backend de l'application, vous devez disposer de Python3 installé sur notre système. Si ce n'est pas déjà fait, vous pouvez télécharger Python à partir du site officiel (https://www.python.org/downloads/). Une fois Python installé, vous pouvez configurer un environnement de virtuel pour isoler les dépendances du projet.

```
# Créez un environnement virtuel

python -m venv venv


# Activez l'environnement virtuel (sous Windows)

venv\Scripts\activate


# Activez l'environnement virtuel (sous macOS/Linux)

source venv/bin/activate
```

Ensuite installez Flask 2.3.3, c'est la version du Framework utilisé pour le backend, ainsi que les autres dépendances en utilisant pip (le gestionnaire de paquets Python) :

pip install + nom du paquet ;

Installer le fichier requirements.txt :

pip install -r requirements.txt

2.2.2. Base de données MySQL

Le backend utilize MySQL comme système de gestion de base de données. Assurez vous d'avoir MySQL installé et configuré sur votre système. Vous devrez également créer une base de données et configurer les informations de connexion dans le fichier de configuration de l'application.

2.3. Configuration du Frontend

2.3.1. Node.js et npm

Le frontend de l'application est développé en utilisant React. Poour commencer, vous devez avoir Node.js et npm (Node Package Manager) installés sur votre machine. Télécharger la version 16 depuis le site officiel de Node.js (https://nodejs.org/).

Une fois Node.js et npm installés, vous pouvez installer les dépendances du projet en utilisant la commande suivante dans le répertoire du frontend :

Npm install

2.3.2. Material-UI

Material-UI est la bibliothèque de composants utilisée pour l'interface utilisateur. Vous n'avez pas besoin de configurer manuellement Material-UI, car il est déjà inclus dans les dépendances du projet.

2.4. Configuration de l'application de Bureau (JAVAFX)

2.4.1. Java Développement Kit (JDK)

Pour le développement de l'application de bureau en JAVAFX, vous devez avoir le JAVA Developpement Kit version 21 (JDK v21) installé sur votre système. Vous pouvez télécharger le JDK depuis le site officiel d'Oracle (https://www.oracle.com/java/technologies/javase-downloads.html) ou utiliser une distribution OpenJDK (ce que j'ai utilisé pour mon développement)

2.5 Configuration de l'application mobile

2.5.1. Flutter

L'application mobile est développée avec Flutter. Pour configurer votre environnement Flutter, suivez les instructions fournies sur le site officiel de Flutter (https://flutter.dev/docs/get-started/install).

2.6. Conclusion

Ce chapitre a couvert les étapes essentielles pour configurer votre environnement de travail afin de développer et d'utiliser notre application. Assurez-vous de suivre attentivement les instructions spécifiques à chaque composant (backend, frontend, application de bureau et application mobile) pour garantir une expérience de développement fluide.

MODELE CONCEPTUEL DE DONNEES

3.1. Introduction

Le modèle conceptuel de données est une représentation abstraite des entités, des relations et des attributs qui composent la structure de données de notre application. Ce modèle joue un rôle essentiel dans la définition de la logique métier et des opérations de gestion de données. Dans ce chapitre, nous allons présenter le modèle conceptuel de données de notre application.

3.2. Schéma du Modèle conceptuel

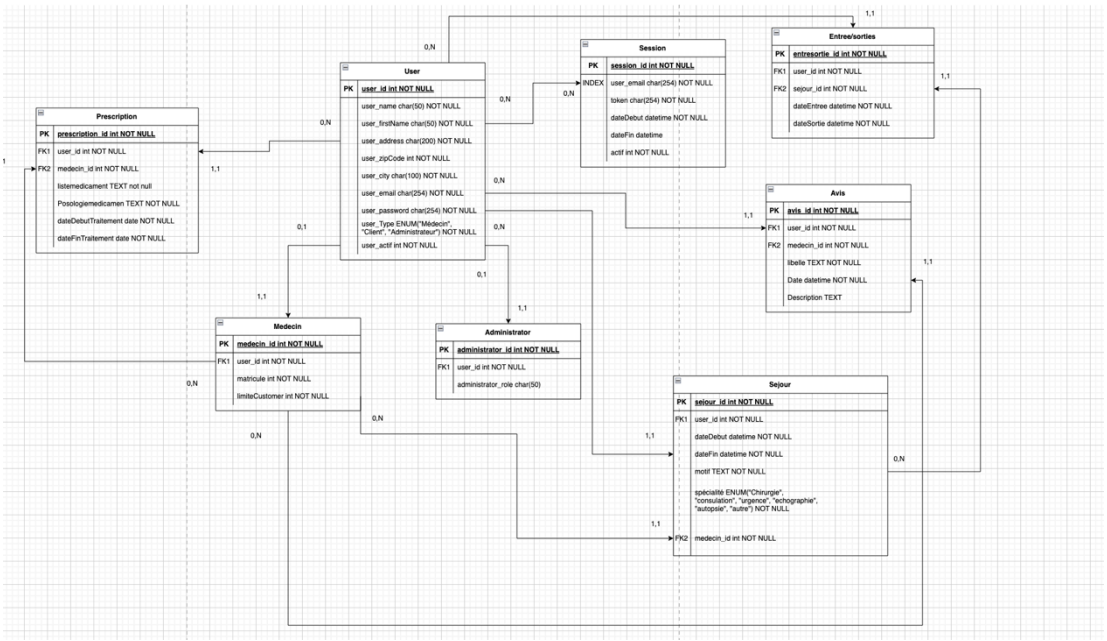

Figure 3.1 : Modèle conceptuel de données

L'image ci-dessus représente le schéma du modèle conceptuel de données de notre application. Cette représentation visuelle illustre les entités principales, leurs attributs et les relations entre elles.

3.3. Conclusion

Ce chapitre a présenté le modèle conceptuel de données de notre application, en mettant en évidence, leurs attributs et les relations qui les relient. Le schéma du modèle conceptuel de données (Figure 3.1) fournit une représentation visuelle claire de la structure de données sous-adjacente à note application.

EXPLICATION DU PLAN DE TESTS

6.1. Introduction

La phase de test est cruciale dans le processus de développement de notre application. Elle vise à garantir que notre application fonctionne correctement, répond aux exigences spécifiques et est dépourvue de bogues ou de problèmes critiques. Dans ce chapitre, nous allons expliquer notre plan de tests, y compris les différentes phases de tests que nous avons prévues, les scénarios de tests et les outils que nous allons utiliser.

6.2. Objectifs des tests

Avant d'entrer dans les détails du plan de tests, définissons les principaux objectifs que nous cherchons à atteindre grâce à notre processus de test :

- **Validation des Fonctionnalités :** Nous allons vérifier que toutes les fonctionnalités spécifiées dans le cahier des charges sont implémentées correctement et fonctionnent conformément aux attentes.
- **Stabilités et fiabilités :** Nous cherchons à identifier et à résoudre les bugs, les erreurs de programmation et les problèmes de stabilité qui pourraient entraîner des plantages ou des dysfonctionnements de l'application.
- **Performance :** Nous allons évaluer les performances de notre application, en nous assurant qu'elle réponde de manière efficace et rapide, même sous charge.
- **Sécurité :** Nous allons tester la sécurité de l'application, en nous assurant qu'elle est protégée contre les attaques potentielles telles que les injections SQL, les failles XSS, etc…

- **Compatibilité :** Nous allons vérifier que notre application est compatible avec différents navigateurs web, systèmes d'exploitation et appareils.

6.3. Phases de tests

6.3.1. Tests Unitaires

Les tests unitaires visent à tester des composants individuels de code, tels que des fonctions ou des méthodes, pour s'assurer qu'ils fonctionnent correctement. Ces tests sont automatisés et permettent d'identifier rapidement les erreurs de programmation.

6.3.2. Tests d'intégration

Les tests d'intégration évaluent la manière dont les différents composants de l'application interagissent les uns avec les autres. Ils s'assurent que l'ensemble de l'application fonctionne correctement.

6.3.3. Tests de validation

Les tests de validation vérifient que l'application répond aux exigences spécifiées dans le cahier des charges. Ils sont basés sur des scénarios d'utilisation réels et sont effectués manuellement.

6.3.4. Tests de performance

Les tests de performance évaluent la réactivité, la vitesse et la stabilité de l'application sous charge. Nous utiliserons des touls

de tests de charge pour simuler des situations d'utilisations intensive.

6.3.5. Tests de sécurité

Les tests de sécurité vérifient que l'application est protégée contre les vulnérabilités et les attaques potentielles. Nous effectuerons des tests de sécurité.

6.3.6. Tests de compatibilité

Les tests de compatibilité s'assurent que l'application fonctionne correctement sur une variété de navigateurs web, de systèmes d'exploitation et d'appareils.

6.4. Scénarios de tests

Nous élaborerons des scénarios de tests détaillés pour chaque phase de tests, en nous basant sur les spécifications fonctionnelles de notre application ? Ces scénarios incluront des étapes précises à suivre, des données de test et des critères de succès. Tels que les tests script unitaire pour les fonctionnalités de l'API.

6.5. Conclusion

La phase de tests est une étape essentielle de notre processus de développement, visant à garantir la qualité, la stabilité, et la sécurité de notre application.