

RAPPORT DE STAGE

Assistant vocal intelligent pour la gestion automatisée des tâches

Réalisé par :
Zayoud Raed

Encadré par :
Mr Yossri Njeh

Année Universitaire : 2025/2026

Table des matières

1 Étude Préliminaire	3
1.1 Introduction	3
1.2 Présentation de la société	3
1.3 Spécification des besoins	4
1.3.1 Identification des acteurs	4
1.3.2 Spécification des besoins fonctionnels	4
1.3.3 Les besoins non fonctionnels	4
1.4 Analyse des besoins fonctionnels	5
1.4.1 Diagramme de cas d'utilisation	5
1.5 Conclusion	5
2 Élaboration	6
2.1 Introduction	6
2.2 Conception architecturale	6
2.2.1 Architecture de la solution	6
2.3 Diagramme de classes	6
2.4 Conclusion	7
3 Construction et Réalisation	8
3.1 Introduction	8
3.2 Construction	8
3.2.1 Environnement logiciel	8
3.2.2 Choix techniques	9
4 Construction et Réalisation	11
4.1 Introduction	11
4.2 Réalisation – Interfaces graphiques	11
4.3 Conclusion	13
Conclusion générale	14

Introduction générale

Dans un monde marqué par l'évolution rapide des technologies numériques, le temps s'impose comme une ressource précieuse à gérer avec efficacité. La planification des tâches et des rendez-vous constitue ainsi un enjeu central, aussi bien pour les professionnels que pour les particuliers. Toutefois, les outils classiques de gestion, bien qu'utiles, présentent encore certaines limites, notamment une saisie manuelle répétitive et chronophage qui peut freiner la productivité.

C'est dans ce contexte que s'inscrit mon stage, dont l'objectif principal est de concevoir et de développer une solution innovante visant à automatiser la gestion des tâches à travers une **interface vocale intelligente**. Cette dernière exploite des technologies modernes telles que la reconnaissance vocale et l'analyse sémantique, afin de transformer des commandes exprimées en langage naturel en actions concrètes : création d'événements, ajout de tâches, planification de réunions ou encore organisation par projets.

Chapitre 1

Étude Préliminaire

1.1 Introduction

Ce chapitre présente l'étude préliminaire du projet, incluant la présentation de l'entreprise d'accueil, la spécification des besoins fonctionnels et non fonctionnels, ainsi que l'analyse des besoins à travers les diagrammes UML.

1.2 Présentation de la société

WebVue est une entreprise tunisienne spécialisée dans les services informatiques, dont le siège est situé à Sfax, route de Sokra km 4, bâtiment Zohra 3052. Depuis sa création, elle s'est positionnée comme un acteur innovant dans le domaine du digital et des technologies de l'information.

Mission et positionnement : La mission principale de WebVue est d'accompagner les entreprises dans leur transformation numérique à travers la mise en place de solutions technologiques modernes, flexibles et adaptées à leurs besoins. Elle vise à propulser les activités de ses clients vers l'avenir, en s'appuyant sur la créativité, l'excellence du service et l'intégration des dernières innovations technologiques.

Domaines d'expertise : WebVue intervient dans plusieurs domaines, notamment :

- le développement d'applications web et mobiles ;
- la conception et la gestion de sites web et de plateformes e-commerce ;
- l'intégration de solutions logicielles (CRM, ERP, SaaS) ;
- le développement d'API et de services sur mesure ;
- le design UI/UX, le branding et la création de contenus multimédia.

1.3 Spécification des besoins

L'expression des besoins a pour objectif d'assurer une compréhension claire des exigences du client. Cette section s'intéresse à déterminer les besoins de notre application en commençant par identifier les acteurs du système.

1.3.1 Identification des acteurs

Un **acteur** est une personne physique ou morale, extérieure au système en cours de modélisation, qui interagit avec ce système. Dans notre application, l'acteur principal est :

- **Client** : il interagit avec le système à travers des commandes vocales exprimées en langage naturel (par exemple : « Planifie une réunion demain à 14 h »).

1.3.2 Spécification des besoins fonctionnels

Les besoins fonctionnels de l'application, liés à l'acteur identifié, sont :

- Créer un compte utilisateur ;
- Créer automatiquement des tâches via commandes vocales ;
- Planifier et gérer des événements (réunions, rendez-vous, etc.) ;
- Mettre à jour ou supprimer des tâches existantes ;
- Consulter et mettre à jour son profil.

1.3.3 Les besoins non fonctionnels

La réussite d'une application ne dépend pas uniquement de ses fonctionnalités, mais aussi d'un certain nombre d'exigences non fonctionnelles :

- **Ergonomie** : interfaces simples, compréhensibles et faciles à utiliser, respectant la charte graphique et la palette de couleurs.
- **Maintenabilité** : respect des normes de codage et de nommage, code commenté pour faciliter les évolutions.
- **Extensibilité** : conception modulaire afin de permettre l'intégration de nouvelles fonctionnalités.

1.4 Analyse des besoins fonctionnels

1.4.1 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation spécifie les besoins des utilisateurs par rapport au système.

Diagramme de cas d'utilisation global

Le diagramme de cas d'utilisation global décrit de manière générale les différents acteurs du système ainsi que les fonctionnalités principales dont ils disposent. Il offre une vue d'ensemble des besoins à satisfaire par le futur système en interaction avec les acteurs.

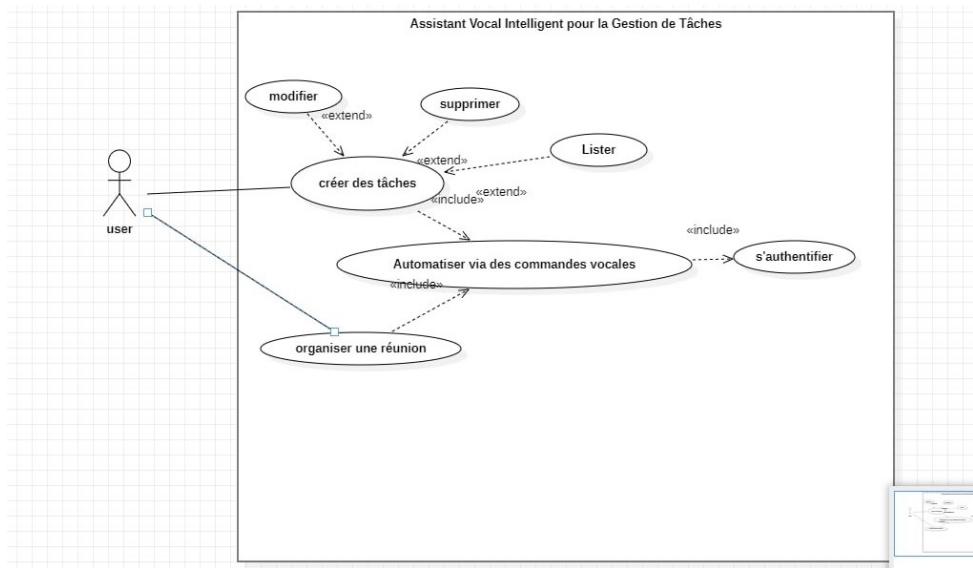


FIGURE 1.1 – Diagramme de cas d'utilisation global

1.5 Conclusion

Ce chapitre a permis de définir le cadre du projet et d'identifier les besoins fonctionnels et non fonctionnels de l'application. Dans le chapitre suivant, nous aborderons la conception architecturale et technique de la solution.

Chapitre 2

Élaboration

2.1 Introduction

Après la définition des besoins de notre projet, nous allons aborder dans ce chapitre l'étude conceptuelle de l'application. Nous présenterons d'abord l'architecture générale de la solution, puis nous détaillerons la conception technique du projet.

2.2 Conception architecturale

2.2.1 Architecture de la solution

L'architecture retenue adopte une approche en trois niveaux :

- **Tiers Front-end** : Cette couche représente la partie visible par l'utilisateur final. L'application mobile a été développée avec le framework **Flutter**, offrant une interface réactive et multiplateforme (Android et iOS).
- **Tiers Back-end** : Le serveur gère la logique métier et la communication entre le front-end et la base de données. Nous avons utilisé le framework **Laravel** (PHP) pour créer une API REST sécurisée, gérer l'authentification des utilisateurs et orchestrer les différentes fonctionnalités.
- **Tiers Base de données** : La persistance des données est assurée par **MySQL**, une base de données relationnelle robuste et performante. Elle stocke toutes les informations nécessaires au fonctionnement de l'application (profils utilisateurs, tâches, événements, etc.).

2.3 Diagramme de classes

Le diagramme de classes est un schéma utilisé en génie logiciel qui permet de présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la vue statique d'UML, car il fait abstraction des aspects

temporels et dynamiques. Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Les éléments basiques du diagramme de classes sont :

— **Classe :**

- *Attribut* : représente un type d'information contenu dans la classe.
- *Opération/Méthode* : représente un élément de comportement (un service) contenu dans une classe.
- **Association** : relation sémantique durable entre deux classes.

Après avoir identifié les différents besoins fonctionnels, nous avons pu dégager les principales classes illustrées dans la figure 2.1 pour obtenir une vue plus claire de l'application étudiée.

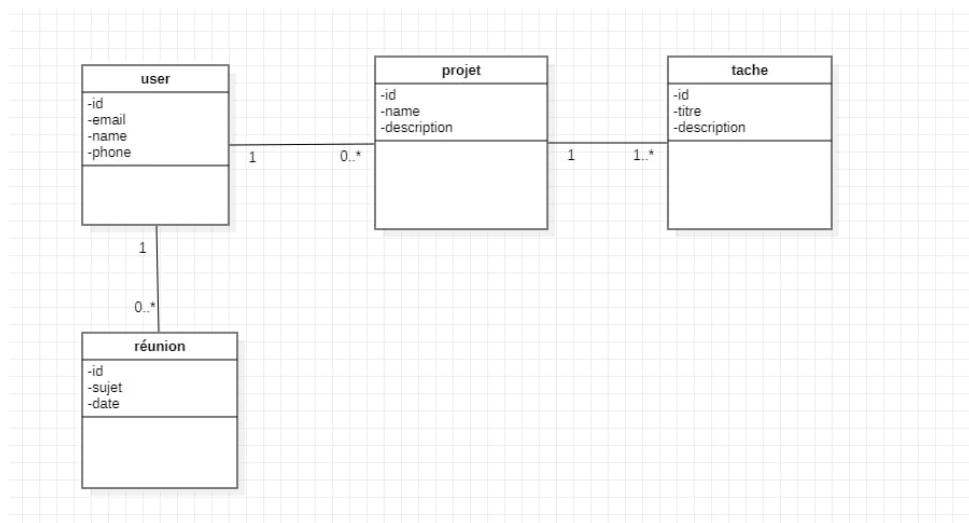


FIGURE 2.1 – Diagramme de classes principal

2.4 Conclusion

Tout au long de ce chapitre, nous avons décrit l'architecture de notre application. Dans le chapitre suivant, nous aborderons la partie **implémentation réelle** de notre application.

Chapitre 3

Construction et Réalisation

3.1 Introduction

Après avoir achevé l'étape d'élaboration, nous entamons, dans ce chapitre, la phase de **construction**, en présentant d'abord l'environnement logiciel ainsi que les différentes technologies utilisées. Ensuite, nous détaillerons la réalisation du projet à travers quelques écrans.

3.2 Construction

Cette partie présente l'environnement de développement logiciel utilisé pour la réalisation de l'application.

3.2.1 Environnement logiciel

Pour réaliser notre projet, nous avons utilisé l'outil logiciel suivant :

— **Visual Studio Code** :

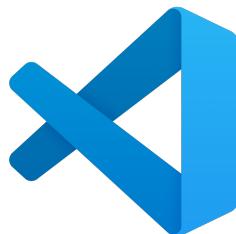


FIGURE 3.1 – Logo de Visual Studio Code

Visual Studio Code (VS Code) est un éditeur de code source gratuit et open-source développé par Microsoft. Il est conçu pour être léger mais puissant, offrant des fonctionnalités telles que la coloration syntaxique, la complétion automatique de code, le débogage intégré, la gestion de projets et l'intégration de contrôle de version (comme Git). VS

Code prend en charge de nombreux langages de programmation et est hautement personnalisable grâce à une large gamme d'extensions disponibles via son marketplace. Il est largement utilisé par les développeurs pour la programmation dans divers langages, y compris JavaScript, Python, C++, Java, et bien d'autres.

3.2.2 Choix techniques

Back end : Laravel

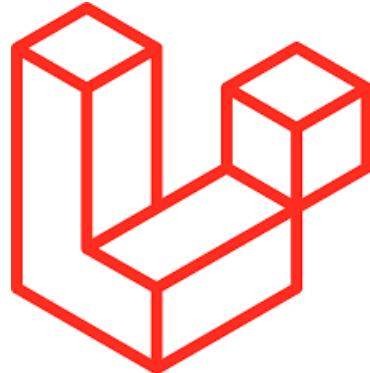


FIGURE 3.2 – Logo Laravel

Laravel est un framework **open-source** PHP destiné au développement d'applications web modernes. Il suit l'architecture **MVC** (Modèle-Vue-Contrôleur) et offre une structure claire pour la gestion du code, l'organisation des routes et la séparation des responsabilités. Parmi ses principales fonctionnalités on trouve : un puissant ORM (Eloquent), la gestion des migrations de base de données, un système d'authentification robuste, une API REST intégrée et une large communauté. Grâce à sa syntaxe expressive et sa grande extensibilité, Laravel permet de développer rapidement des applications sécurisées et évolutives.

Front end : Flutter



FIGURE 3.3 – Logo Flutter

Flutter est un framework open-source développé par Google pour la création d'applications mobiles, web et de bureau à partir d'une seule base de code. Il permet de concevoir

des interfaces utilisateur natives de haute qualité grâce à un système de widgets personnalisables. Le langage de programmation utilisé est **Dart**. Grâce à son moteur de rendu rapide et à sa riche collection de widgets pré-construits, Flutter facilite la création d'applications performantes et visuellement attrayantes sur iOS, Android, Windows, macOS, Linux et le Web.

Gestion de base de données : MySQL / phpMyAdmin



FIGURE 3.4 – Logo MySQL / phpMyAdmin

La persistance des données est assurée par **MySQL**, une base de données relationnelle robuste et largement utilisée. Pour l'administration et la gestion des données, nous utilisons **phpMyAdmin**, une interface web qui permet d'exécuter des requêtes SQL, de gérer les schémas et de superviser les tables de manière simple et intuitive. Cette combinaison offre une solution fiable, performante et facile à maintenir pour la gestion des données de l'application.

Chapitre 4

Construction et Réalisation

4.1 Introduction

4.2 Réalisation – Interfaces graphiques

Cette section présente quelques interfaces principales de l'application **Voice Task**.

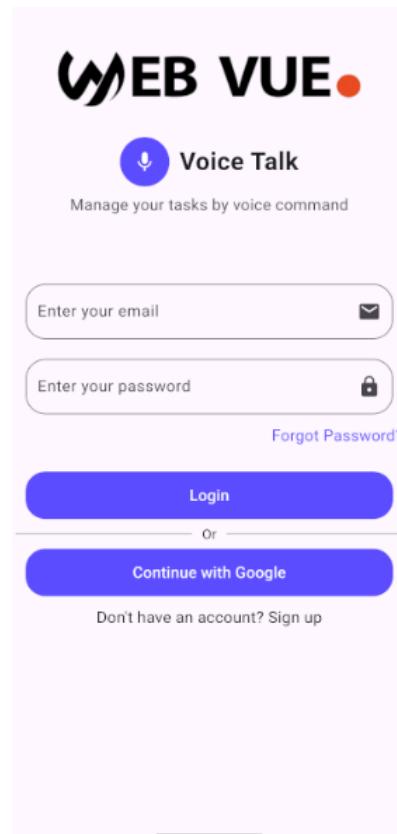


FIGURE 4.1 – Interface de connexion (Login)

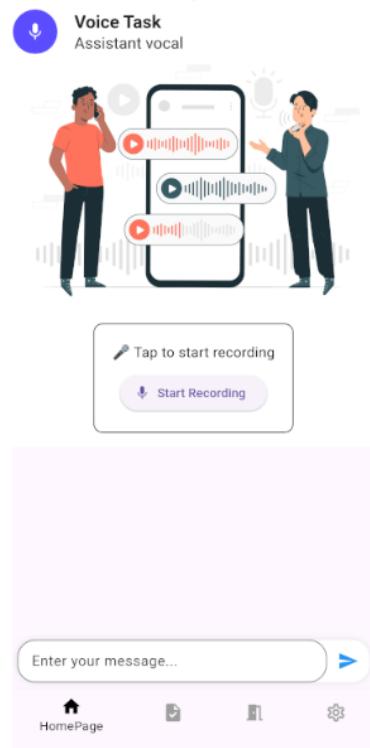


FIGURE 4.2 – Interface d'enregistrement vocal (Home Page)

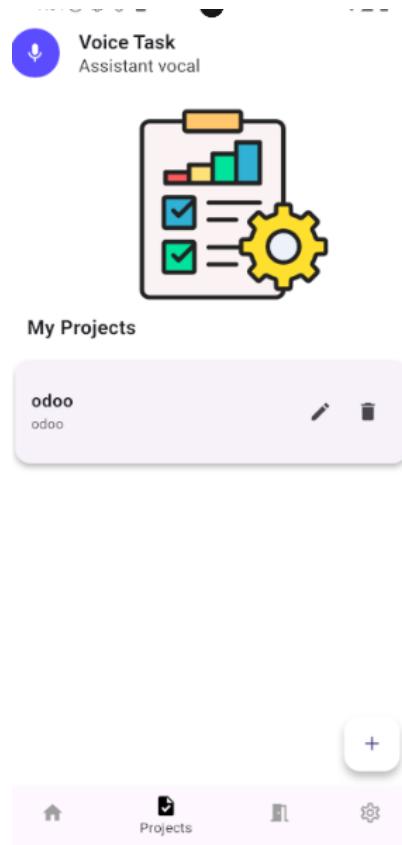


FIGURE 4.3 – Interface de gestion des projets (Projects)

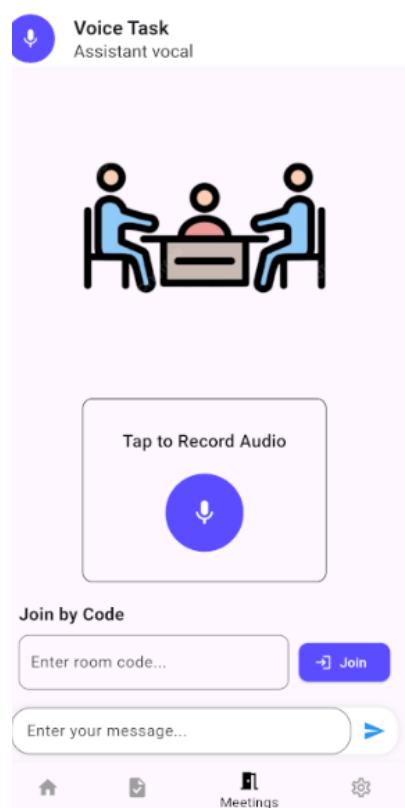


FIGURE 4.4 – Interface de réunions avec enregistrement vocal (Meetings)

4.3 Conclusion

Ce chapitre a présenté la phase de construction et de réalisation de l'application, incluant les choix techniques, l'environnement de développement et les interfaces graphiques finales. Dans la conclusion générale, nous ferons le bilan du projet et discuterons des perspectives d'évolution.

Conclusion générale

La réalisation de ce projet d'assistant vocal intelligent a permis de développer une solution pratique qui combine reconnaissance vocale et gestion de tâches. Ce travail m'a permis de mettre en application les connaissances acquises durant ma formation, d'approfondir mes compétences en développement mobile, en conception d'API et en gestion de bases de données. L'application offre une utilisation simple et rapide pour créer et organiser des tâches, ce qui améliore la productivité.

Ce projet reste une première étape et pourra évoluer vers de nouvelles fonctionnalités, comme une meilleure compréhension du langage naturel, la synchronisation avec d'autres outils ou l'ajout de rappels intelligents. Dans l'ensemble, cette expérience a été très enrichissante et m'a permis de lier théorie et pratique à travers une réalisation concrète et utile.