Rapport d'alternance Septembre 2025

Développeur Full Stack

YILMAZ Rahman

Formation : Master 2 Bases de Données et Intelligence Artificielle
Année : 2024-2025

Maître d'apprentissage : JÉGARD Johan (johan.jegard@sweepin.fr)
Tuteur universitaire : ROXIN Ana (ana-maria.roxin@u-bourgogne.fr)

Établissement : Université de Bourgogne Europe
Entreprise : Solo Agilis Sweepin

Sommaire

Table des figures

1. Remerciements

Je tiens tout d'abord à exprimer ma gratitude envers M. Eric Leclerc qui m'a fait découvrir l'entreprise Sweepin et m'a permis de postuler au sein de celle-ci.

Un grand merci à mon établissement, l'UFR Sciences et Techniques, où j'étudie depuis ma licence 1. Cette institution m'a permis d'acquérir de nombreuses connaissances et a suscité en moi un véritable amour pour l'informatique.

Je tiens à exprimer ma sincère gratitude envers Madame Nadine Cullot, dont les explications approfondies sur les objectifs et les avantages du Master Bases de Données et Intelligence Artificielle ont renforcé mon enthousiasme à poursuivre mes études au sein de cet établissement.

Mes remerciements vont également à ma tutrice universitaire, Madame Ana Roxin, pour sa confiance en moi, ses précieux conseils et la lecture de mon rapport.

Je n'oublie pas de mentionner M. Johan Jégard, CEO de Sweepin, qui a su reconnaître mon potentiel et m'a accueilli au sein de son entreprise. En tant que nouveau maître d'apprentissage depuis le départ de William Mouline, il a su maintenir un encadrement de qualité tout en me donnant l'autonomie nécessaire pour mener à bien ce projet ambitieux.

Des remerciements spéciaux à mon lecteur et correcteur qui, grâce à ses recommandations et corrections, a veillé au bon déroulement de la rédaction de mon rapport.

2. Introduction

J'ai opté pour le parcours de master "Bases de Données et Intelligence Artificielle" à l'UFR Sciences et Techniques pour diverses raisons. La possibilité de le suivre en alternance a été un élément déterminant, et les unités d'enseignement prévues s'alignent parfaitement avec mes objectifs professionnels futurs, notamment dans le domaine de la santé numérique qui représente un secteur en pleine expansion.

Le choix d'effectuer mon alternance chez Sweepin, une startup novatrice spécialisée dans le développement d'applications mobiles dédiées aux collectivités et aux établissements de santé, découle de mon intérêt constant pour la conception d'applications ayant un impact social significatif. Cette opportunité représentait une ouverture vers de nouveaux défis techniques et humains, particulièrement dans le secteur médical où la technologie peut littéralement sauver des vies.

Cette deuxième année d'alternance m'a permis d'évoluer vers un poste de développeur Full Stack, me donnant l'opportunité de travailler sur l'ensemble de la chaîne de développement. Cette évolution s'est concrétisée par ma participation au projet Floween, une application web progressive révolutionnaire destinée au suivi des patients dans les services d'urgence.

Le projet Floween représente un défi technique et métier particulièrement stimulant. Dans un contexte où les services d'urgence français font face à une saturation croissante - avec plus de 21 millions de passages aux urgences en 2023 selon la DREES - l'amélioration de l'expérience patient devient cruciale. Les temps d'attente moyens dépassent souvent 4 heures dans les grands centres hospitaliers, créant stress et anxiété chez les patients et leurs proches. C'est dans ce contexte que s'inscrit notre solution technologique.

Mes objectifs professionnels au sein de cette alternance se concentrent sur la maîtrise des technologies web modernes, l'acquisition d'une expertise technique solide dans le domaine de la e-santé, et la participation active à la conception d'une application innovante destinée à améliorer concrètement la prise en charge des patients. Cette expérience au sein de l'équipe de développement représente pour moi une opportunité de concrétiser mes aspirations professionnelles en contribuant à un projet ayant un impact sociétal direct et mesurable.

3. Présentation entreprise

3.1. Sweepin

Solo Agilis Sweepin est une entreprise fondée en 2015 par Johan JÉGARD, Matthieu DIBAJI et Yanis KERFA. Son siège social est établi à Dijon. Elle compte environ une dizaine de collaborateurs. Nous avons plus de 100 clients principalement en France mais également dans d'autres pays comme l'Allemagne. Nos applications mobiles ont été téléchargées plus d'un million de fois dans les différents store Google play store et l'App store.
Le groupe Tessi est actionnaire de Sweepin depuis 2022. Tessi est une société française de plus de 13 000 collaborateurs opérant dans 15 pays et avec un chiffre d'affaires de 532 millions d'euros en 2023.
L'entreprise Sweepin est spécialisée dans 2 domaines. Premièrement la géolocalisation intérieure et la seconde est la création d'applications mobiles pour les villes.

3.2. Géolocalisation intérieure

L'IPS, pour Indoor Positioning System (géolocalisation intérieure), est une technologie permettant la localisation d'un objet ou d'une personne à l'intérieur d'un bâtiment. Les solutions de géolocalisation sont à destination notamment des établissements de santé, mais également de tout établissement souhaitant un système de géolocalisation intérieure dernière génération. Ce service comprend 5 solutions toutes indépendantes. La cartographie 3D et le guidage indoor web sont des solutions sans application mobile. Le guidage indoor en temps réel s'effectue à l'aide d'une application mobile téléchargée au préalable sur smartphone. Une solution de suivi des patients est assurée par un bracelet muni d'une balise bluetooth. La protection du personnel est renforcée dans les établissements de santé. Enfin, la géolocalisation du matériel dans un établissement de santé complète cette offre.

Ces innovations ont déjà conquis de nombreux établissements prestigieux comme les HCL, le CHU Amiens-Picardie, l'Hôpital Foch ou l'Hôpital Novo, qui témoignent chaque jour des bénéfices concrets apportés à leurs équipes et patients.

3.3. SmartCity

Smartcity est le service que nous proposons aux villes mais pas uniquement, elle s'adresse également aux territoires désireux d'optimiser la vie urbaine et de renforcer le lien entre citoyens, acteurs économiques et institutions. Nos applications natives pour Android et iOS offrent une expérience fluide et intuitive, en intégrant une large palette de modules. Qu'il s'agisse d'informer sur les événements et actualités, de faciliter le signalement d'incidents ou d'offrir des fonctionnalités spécifiques comme la gestion des transports et des déchets, chaque module a été pensé pour améliorer le quotidien des usagers.

Tous ces modules, au nombre de plus d'une quinzaine, sont administrables via une plateforme de back-office conviviale et performante, permettant aux clients de gérer en temps réel le contenu et d'accéder à des statistiques précises sur l'utilisation de leur application. Et parce que chaque collectivité a ses besoins propres, nous offrons la possibilité de développer des modules sur mesure afin de répondre précisément aux exigences spécifiques de nos clients.

Des villes telles que Bordeaux, Strasbourg, Dunkerque, Dole et bien d'autres nous font déjà confiance pour transformer leur territoire en un espace connecté et interactif. Un exemple illustratif est l'application **"Châteaux et légendes"**, conçue pour la Collectivité européenne d'Alsace. Cette application ludique permet aux utilisateurs de découvrir le riche patrimoine de la région – en particulier ses châteaux à travers des quiz interactifs, combinant ainsi culture et divertissement.

3.4. Back office

Le back office est intégré de manière transparente dans les deux services. Il facilite la gestion du contenu au sein des applications, permettant l'ajout et la modification du contenu, l'accès aux statistiques, ainsi que l'automatisation de certaines fonctionnalités.

3.5. Equipe

En interne, l'entreprise se divise en trois équipes distinctes. Tout d'abord, il y a l'équipe chargée de la communication et de la gestion de projet. Ensuite, il y a l'équipe de développement frontend, subdivisée en deux sous-équipes: l'une dédiée au développement pour le système d'exploitation d'Apple, iOS, et l'autre axée sur le système d'exploitation de Google, c'est-à-dire Android. Et enfin il y a la 3eme et dernière équipe de développement backend. En tant que développeur fullstack, je contribue à la fois au développement Android et aux projets backend en PHP, je suis l'exception dans l'entreprise, je switch d'équipe selon mes tâches.

3.5.1. Ma place dans l'entreprise

Je suis un développeur full-stack avec une spécialisation en développement frontend moderne. J'ai intégré l'entreprise en tant que développeur Android, mais depuis la fin de ma première année, j'interviens également sur des projets web et backend. Pour le projet Floween, j'ai pris la responsabilité du développement frontend, ce qui m'a permis d'explorer en profondeur les technologies web modernes comme SvelteKit et les Progressive Web Apps.

Mes responsabilités sur Floween incluent la conception de l'architecture frontend, l'implémentation de l'interface utilisateur responsive, l'intégration avec l'API backend, la gestion des états d'authentification, l'implémentation des fonctionnalités PWA, et l'optimisation des performances pour un usage mobile intensif en environnement hospitalier.

3.6. Méthodologie agile

3.6.1. Sprint

Chaque lundi, l'équipe se réunit pour une revue complète des tâches réalisées lors du sprint précédent, qui s'étend sur une période de deux semaines. Les éventuels retards ou blocages sont analysés, et les tâches non terminées sont réévaluées pour être réintégrées dans le sprint suivant si nécessaire. Cette réunion est également l'occasion de définir les priorités et objectifs du sprint à venir, en veillant à une répartition équilibrée des responsabilités. Cette approche permet de maintenir une progression continue tout en assurant une adaptation rapide aux imprévus.

Dans le cadre de notre méthodologie agile, nous utilisons ClickUp pour structurer et optimiser la gestion de nos sprints. Cet outil nous permet de suivre l'avancement des tâches, de gérer les tickets de correction et de prioriser les missions en fonction de leur importance et de leur urgence.

Grâce à ses fonctionnalités de planification, nous pouvons définir clairement les échéances, garantissant ainsi une gestion efficace du temps et une meilleure anticipation des livrables. L'outil permet également d'attribuer des statuts précis aux tâches, offrant une visibilité sur la progression et facilitant les ajustements nécessaires en cours de sprint.

ClickUp favorise aussi le travail collaboratif en centralisant les échanges entre les membres de l'équipe. Les fonctionnalités de commentaires, d'assignation des tâches et de partage d'informations renforcent la coordination et garantissent une meilleure organisation. Cela nous permet d'améliorer l'efficacité de nos sprints de deux semaines, de fluidifier les workflows et d'assurer une réactivité optimale face aux évolutions du projet.

3.6.2. Scrum

Chaque matin, vers 9h15, les deux équipes de développement se réunissent pour tenir un scrum. Au cours de cette réunion, chaque membre prend la parole à tour de rôle, disposant d'environ 30 secondes pour partager ses accomplissements de la veille, les défis rencontrés, ainsi que ses objectifs pour la journée. Cette démarche vise à évaluer la progression du développement, à détecter d'éventuels retards entre les équipes. Par exemple, si l'équipe de développement back-end accumule un retard sur les appels API, cela pourrait ralentir le rythme de l'équipe mobile.

3.6.3. Communication

En interne, notre mode de communication repose sur une approche transversale, sans structure hiérarchique formelle. Cette démarche favorise des échanges réguliers entre les différentes équipes. En tant que développeur FullStack mais principalement Android, je suis fréquemment en contact avec mes collègues spécialisés dans le développement iOS, abordant divers aspects tels que l'interface

utilisateur et les fonctionnalités. Assurer une cohérence entre les applications Android et iOS demeure essentiel.

Par ailleurs, il m'arrive de formuler des demandes spécifiques à l'équipe de développement back-end s'il m'est impossible de le faire moi même, comme par exemple pour la création d'appels API particuliers. Nos deux équipes de développement ont la possibilité de se réunir afin de résoudre des problématiques communes, comme cela s'est récemment produit lors d'une difficulté liée aux fuseaux horaires. Face à un client situé dans un fuseau horaire différent de la France, des ajustements ont été nécessaires. L'équipe back-end a ainsi converti toutes les dates dans le fuseau horaire français, tandis que de notre côté, nous avons dû gérer l'affichage de la date en fonction de la localisation de l'utilisateur. Par exemple, un événement à Paris à 19h00 serait à 13h00 à New York.

En parallèle, je maintiens une communication régulière avec l'équipe en charge de la communication et de la gestion de projet, assurant le lien entre les clients et les équipes de développement. Toute interaction exige une adaptation constante de mon langage pour garantir une compréhension optimale.

Pour les échanges avec mes collègues, nous utilisons Google Chat et Slack, slack est également utilisé pour communiquer avec les clients. Pour tester et inspecter les résultats des appels API, j'utilise Postman, un outil essentiel dans le processus de développement.

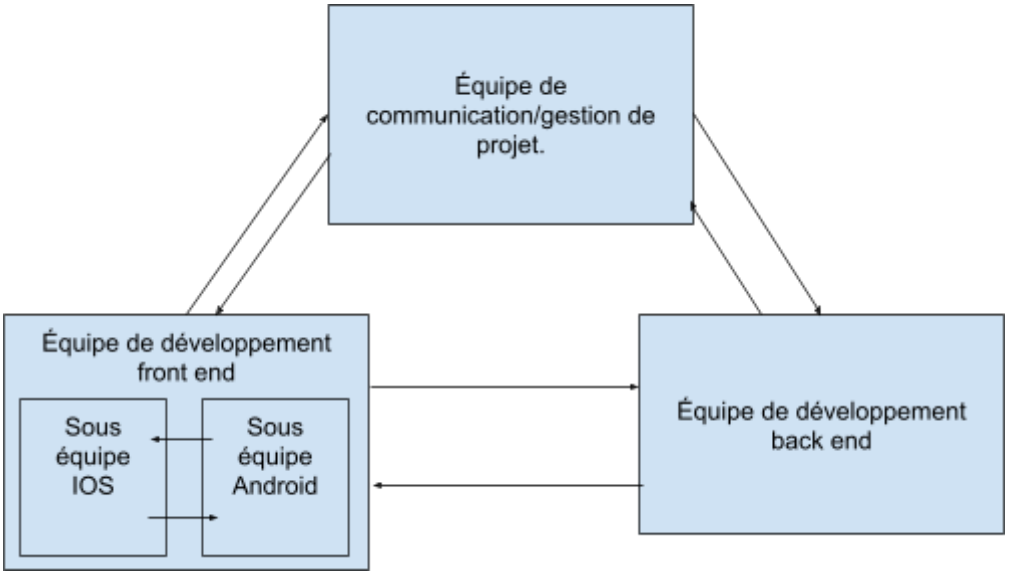


Figure 1. Les différentes équipes, les flèches représentent les communications.

4. Mission principale

4.1. Contexte et objectifs

4.1.1. Analyse du marché et positionnement concurrentiel

Le projet **Floween** (nommé comme cela en interne) naît d'une analyse approfondie du marché des solutions de suivi patient aux urgences. Le leader actuel, **FollowMe Urgences,** développé par deux médecins urgentistes, a déjà conquis plusieurs établissements prestigieux dont **l'Hôpital Foch** et le **CHU d'Amiens**. Cette solution, reconnue par **l'Agence du Numérique en Santé** (ANS) et lauréate des **Talents de la e-santé 2024**, a démontré la viabilité économique et technique du concept.

L'analyse concurrentielle révèle que **FollowMe** présente certaines limitations techniques et fonctionnelles que **Floween** ambitionne de dépasser. Notamment, **FollowMe** utilise une architecture plus traditionnelle et ne propose pas de distinction avancée entre les vues patient et accompagnant, un élément que nous considérons comme crucial pour respecter la confidentialité médicale tout en informant les proches.

Le marché représente un potentiel considérable : la France compte **630 services d'urgence** qui accueillent annuellement plus de **21 millions de patients**.

4.1.2. Objectifs stratégiques du projet

L'application Floween doit répondre à plusieurs objectifs stratégiques :

Objectifs fonctionnels :

- Permettre un suivi en temps réel du parcours patient aux urgences
- Différencier l'information selon le profil (patient vs accompagnant)
- Assurer l'interopérabilité avec les systèmes hospitaliers existants
- Garantir une expérience utilisateur optimale sur mobile
- Respecter les contraintes de sécurité et de confidentialité du secteur médical

Objectifs techniques :

- Développer une architecture moderne et scalable
- Implémenter une Progressive Web App performante
- Assurer l'internationalisation de l'application
- Optimiser les performances pour les réseaux hospitaliers souvent saturés

Objectifs métier :

- Asseoir la position de Sweepin comme acteur majeur de la e-santé : fort de notre expertise sur les solutions de tracking patient déjà en production, l'ajout

d'un module de suivi patient spécifiquement dédié aux situations d'urgence renforce considérablement notre légitimité et notre poids sur ce marché.

- Acquérir une référence prestigieuse avec l'Hôpital de Monaco

4.1.3. Répartition des rôles projet Floween

Une équipe projet dédiée de quatre personnes a été constituée afin d'assurer la réalisation du projet Floween et la couverture de toutes les compétences nécessaires :

- Le CEO (Johan Jégard) supervisait les points stratégiques : il définissait les objectifs et le périmètre du projet, recueillait les exigences fonctionnelles et validait les étapes avec les parties prenantes.
- J'étais en charge du développement front-end et de la conception ainsi que de l'implémentation de toute l'interface utilisateur web, en assurant la cohérence technique avec les autres pôles.
- Mohamed Laaraiedh prenait en main la création des maquettes visuelles et l'ergonomie des parcours utilisateurs, ce qui permettait d'intégrer les meilleures pratiques de design dès les premières phases du projet.
- William Maudit assurait le développement back-end : il concevait et maintenait l'API et la logique métier au cœur de l'application.

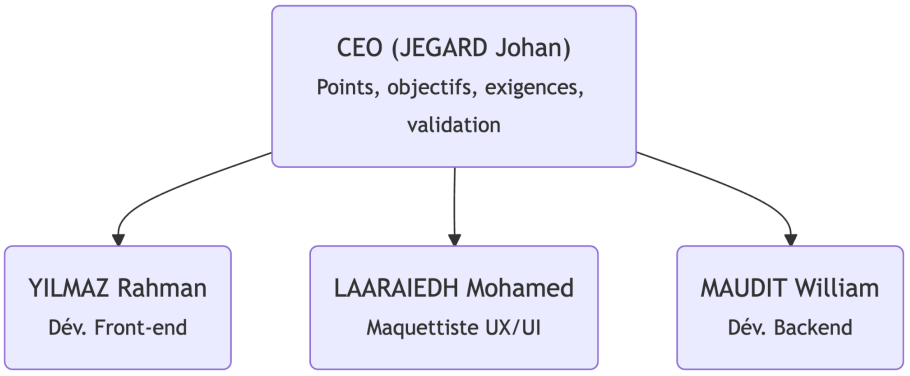


Figure 2. L'organigramme suivant illustre la répartition des rôles et la hiérarchie de l'équipe projet

4.2. Cahier des charges technique

4.2.1. Exigences fonctionnelles

L'application Floween doit être accessible depuis un navigateur web et implémentée comme une Progressive Web App (PWA) pour bénéficier des avantages du web tout en offrant une expérience proche d'une application native. L'application doit être entièrement internationalisée et mobile-only, reflétant l'usage principal sur smartphone en milieu hospitalier.

Le système doit gérer deux vues différenciées :

- Vue patient : accès complet aux informations médicales, résultats d'examens, temps d'attente estimés, et détails des prochaines étapes
- Vue accompagnant : informations limitées excluant les données confidentielles comme les résultats de prises de sang, diagnostics, ou autres données médicales sensibles

L'interface utilisateur s'inspire de FollowMe Urgences avec une liste de cards verticales représentant chaque étape du parcours aux urgences. Chaque card contient les détails de l'étape : position dans le parcours, durée estimée, nom de l'étape, description, et éventuelles sous-étapes. Les cards sont différenciées par un code couleur indiquant si l'étape est en cours (orange), terminée (vert), ou à venir (gris).

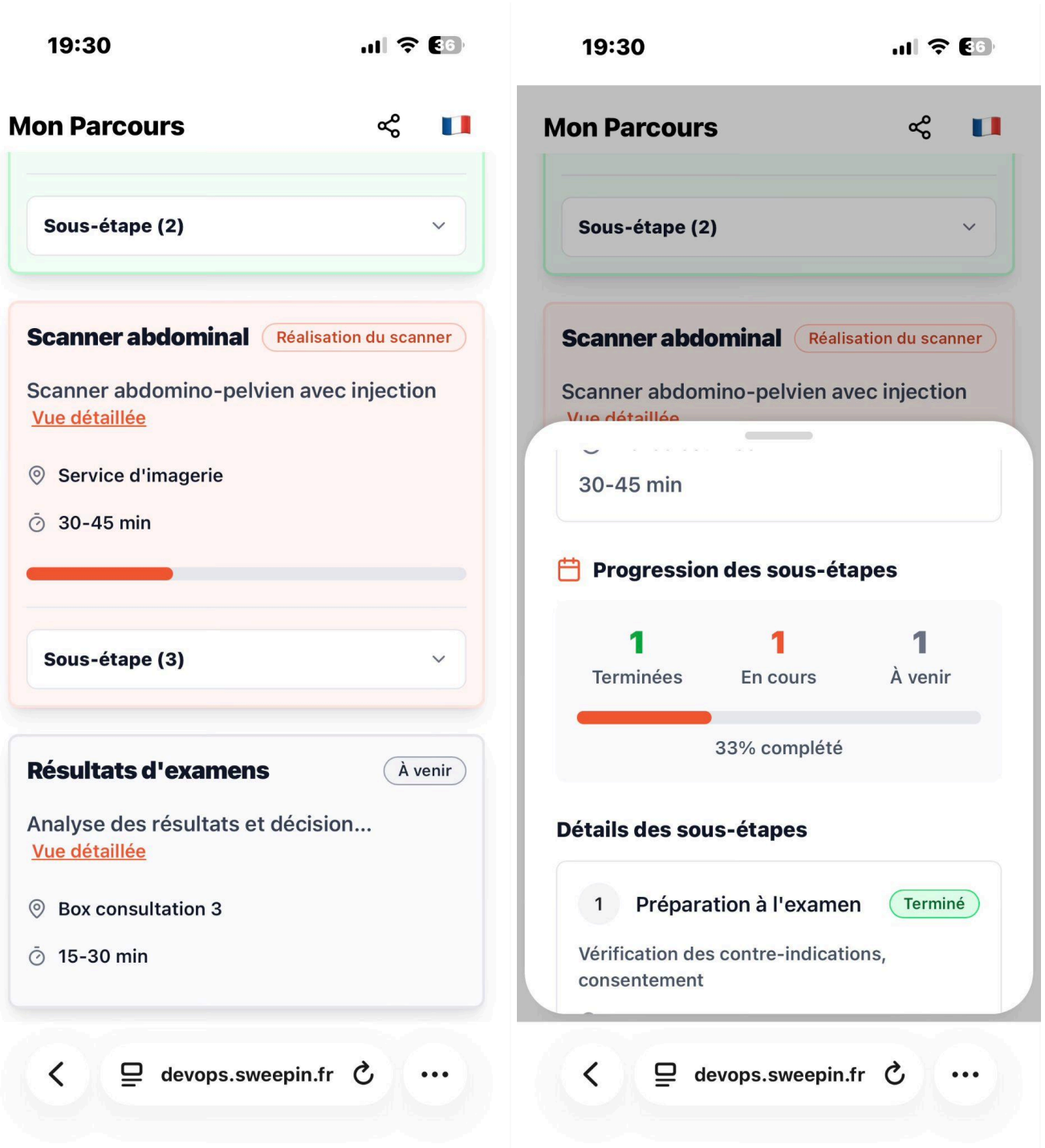


Figure 3 et 4 : Aperçu de l'application web avec des données générées par IA.

4.2.2. Exigences techniques

Architecture frontend :

- SvelteKit SSR (Server-Side Rendering) avec NodeJs pour optimiser les performances et gérer un rendu conditionnel selon le type d'utilisateur (patient ou accompagnant).
- TypeScript pour la robustesse du code et la maintenance
- TailwindCSS comme framework CSS utilitaire
- Flowbite Svelte pour les composants préfabriqués et l'accélération du développement
- PWA avec service workers pour mettre en cache les ressources statiques et dynamiques

Architecture backend :

- PHP 8.3 avec Symfony 7 pour l'API REST
- MariaDB pour la persistance des données
- JWT pour l'authentification et la sécurisation des sessions
- WildFly pour le traitement des données HL7
- Apache comme serveur web

Sécurité et authentification :

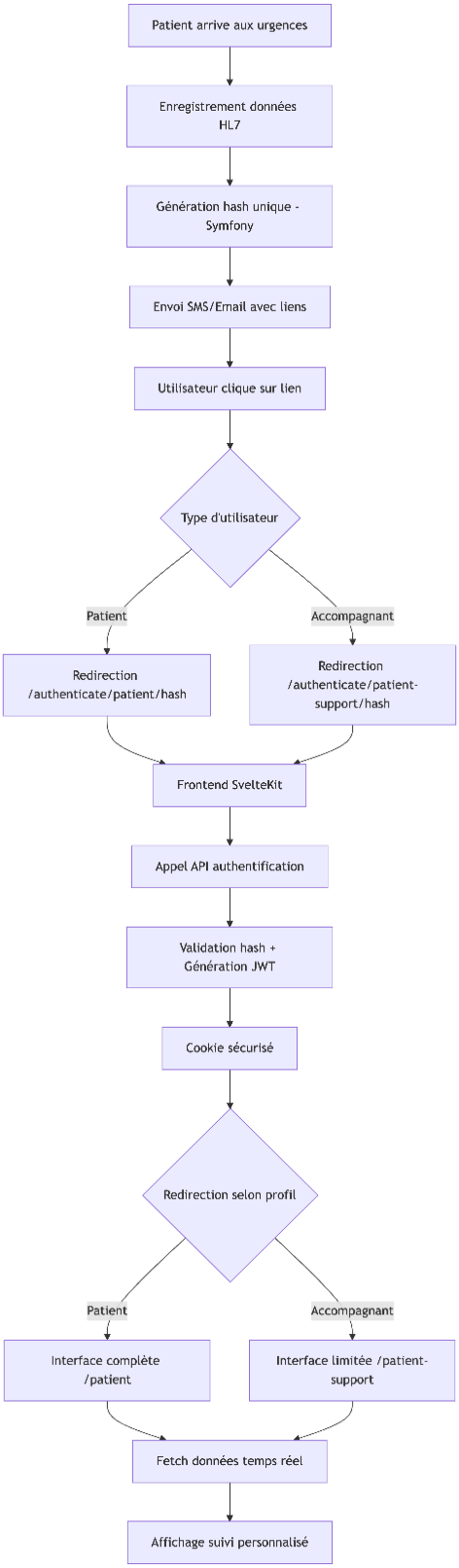
- Génération de hash unique par Symfony pour chaque patient
- Envoi d'URLs sécurisées par SMS/email : hostname/authenticate/patient/hash et hostname/authenticate/patient-support/hash
- Tokens JWT stockés en cookies, set côté serveur pour la sécurité
- HTTPS obligatoire en production
- Respect du RGPD et des normes de sécurité hospitalières

Figure 5. Stack technique de Floween

4.3. Architecture technique détaillée

4.3.1. Flux de données et authentification

Le système **Floween** fonctionne selon le processus suivant :



1. **Arrivée du patient** : Un patient arrive aux urgences et se fait enregistrer au bureau d'accueil

2. **Saisie des données** : Le personnel médical saisit les informations du patient et, optionnellement, le numéro de téléphone de son accompagnant

3. **Intégration HL7** : Les données patient sont intégrées dans la base de données via le protocole **HL7** standard, traité par le serveur **WildFly**

4. **Génération des liens** : Symfony génère un **hash unique** et crée les URLs d'authentification différenciées

5. **Notification** : Les liens sont envoyés par SMS ou email aux utilisateurs concernés

6. **Authentification** : Lorsque l'utilisateur clique sur le lien, il arrive sur le frontend **SvelteKit**

7. **Validation** : Le serveur frontend effectue un appel API avec le hash pour authentifier l'utilisateur

8. **Token JWT** : L'API retourne un token **JWT** qui est stocké en cookie côté serveur

9. **Redirection** : L'utilisateur est redirigé vers /patient ou /patient-support selon son profil

10. **Suivi temps réel** : Le serveur effectue des appels fetch à l'API avec le token pour récupérer les données de suivi

Figure 6. Schéma du parcours d’authentification patient/accompagnant et affichage du suivi personnalisé dans Floween.

4.3.2. Gestion des Progressive Web Apps

Pour la PWA, une page racine / a été créée et affiche les suivis en fonction des tokens présents en cookies. **L'avantage majeur de cette architecture réside dans le partage automatique des cookies entre la version navigateur et la PWA installée**. En effet, puisque les deux versions proviennent du même domaine, elles accèdent naturellement aux mêmes cookies stockés par le navigateur.

Cette approche technique offre plusieurs bénéfices concrets :

- **Continuité de l'expérience utilisateur** : un utilisateur peut commencer à consulter son suivi dans un navigateur web, puis passer à la PWA installée sans perdre sa session d'authentification

- **Simplicité de gestion** : aucun mécanisme de synchronisation complexe n'est nécessaire, les tokens JWT stockés en cookies sont automatiquement accessibles dans les deux contextes

- **Sécurité préservée** : chaque utilisateur ne voit que les suivis associés à ses propres tokens, garantissant confidentialité et sécurité, que l'accès se fasse via le navigateur ou l'application installée

Les fonctionnalités PWA incluent :

- **Installation sur l'écran d'accueil** pour un accès rapide, simulant une véritable application native et offrant un lancement direct sans passer par le navigateur.

- **Performance optimisée avec mise en cache intelligente** grâce à l'utilisation de service workers : les ressources statiques (HTML, CSS, JS) ainsi que certaines données dynamiques sont automatiquement stockées côté client, permettant un affichage instantané même sur des réseaux mobiles dégradés ou en cas de perte temporaire de connexion.

- **Responsive design adapté à tous les écrans mobiles** : l'interface utilisateur s'adapte automatiquement à la taille et à l'orientation de l'écran, assurant une expérience fluide aussi bien sur smartphones que sur tablettes.

4.3.3. Gestion de l'accès multi-périphériques et sécurisation des vues

Dans le cadre du projet Floween, l'accès aux interfaces sensibles (vues patient et accompagnant) a été strictement restreint aux appareils mobiles uniquement. Cette décision s'explique par la conception de certains composants d'interface, spécifiquement conçus pour une interaction tactile (gestes swipe, tap, etc.), ainsi que par la volonté de garantir une expérience optimale et sécurisée à l'utilisateur en situation de mobilité.

Implémentation technique :

- À chaque requête d'utilisateur, le serveur Node.js inspecte l'en-tête User-Agent envoyé par le navigateur. Ce champ permet d'identifier de façon fiable la nature du périphérique (mobile, tablette, desktop).
- Toute tentative d'accès aux pages sensibles depuis un navigateur desktop (ordinateur de bureau, laptop, tablette en mode bureau) déclenche une redirection automatique vers la page "/desktop-not-supported".
- Cette page informe explicitement l'utilisateur que l'application est réservée aux terminaux mobiles et propose, via un QR code, de poursuivre la navigation sur smartphone pour bénéficier de la meilleure expérience possible.

Bénéfices :

- Prévention de toute utilisation inadaptée des fonctionnalités réservées au tactile.
- Sécurité renforcée par la limitation de l'accès aux données confidentielles sur des appareils dont l'environnement pourrait être partagé ou moins sécurisé (PC public, poste partagé…).

Figure 7. Page pour les appareils non mobile

5. Analyse technique approfondie

5.1. Choix technologiques et justifications

5.1.1. SvelteKit : Le framework frontend de nouvelle génération

Le choix de **SvelteKit** pour le frontend représente une décision technique stratégique basée sur plusieurs critères d'évaluation rigoureux. Contrairement aux frameworks traditionnels comme **React** ou **Vue.js**, **SvelteKit** compile les composants au moment du build, éliminant le besoin d'une runtime library volumineuse côté client. De plus, nous avions déjà un autre projet réalisé avec ce framework : afin d'éviter de multiplier les stacks technologiques et de favoriser la réutilisation des compétences au sein de l'équipe, il a été logique de s'appuyer sur SvelteKit pour garantir cohérence, mutualisation et efficacité dans la maintenance des applications.

Avantages techniques de SvelteKit :

Performance exceptionnelle : Les applications **SvelteKit** génèrent un code JavaScript optimisé sans virtual DOM, résultant en des temps de chargement **30 à 50% plus rapides** que les solutions concurrentes selon les benchmarks de performance 2024. Cette performance est cruciale dans un environnement hospitalier où la connectivité peut être limitée.

Server-Side Rendering natif : SvelteKit intègre nativement le **SSR**, permettant un premier rendu côté serveur essentiel pour les **PWA** et l'optimisation **SEO**. Cette fonctionnalité améliore significativement l'expérience utilisateur en réduisant le **Time to First Contentful Paint** de **40% en moyenne**.

Routing dynamique avancé : Le système de routing de SvelteKit permet une gestion élégante des routes dynamiques nécessaires pour l'authentification par hash. La structure de fichiers intuitive facilite la maintenance et l'évolution du code.

Écosystème en croissance : Avec **plus de 60 000 étoiles sur GitHub** et une adoption croissante par des entreprises comme **The New York Times** et **Apple**, SvelteKit démontre sa maturité et sa viabilité à long terme.

5.1.2. Authentification JWT et sécurité

Mécanisme d'authentification par hash

Le système d'authentification repose sur un mécanisme sécurisé utilisant des liens comportant un hash unique. Cette approche élimine le besoin de mots de passe traditionnels tout en maintenant un niveau de sécurité élevé.

Processus d'authentification :

1. Génération du lien sécurisé : Un hash unique est généré pour chaque utilisateur et associé à une durée de validité limitée
2. Validation côté serveur : L'API Symfony vérifie la validité du hash en base de données
3. Génération du JWT : Une fois le hash validé, un token JWT est créé avec les informations utilisateur et les permissions associées
4. Stockage sécurisé : Le token est stocké dans un cookie sécurisé côté client avec les flags HttpOnly et Secure

Sécurisation des échanges

Protection des données sensibles :

- Chiffrement des communications via HTTPS obligatoire
- Validation des données d'entrée avec filtrage strict des paramètres
- Contrôle d'accès basé sur les rôles utilisateur définis dans le JWT

Gestion des sessions :

- Expiration automatique des tokens JWT (durée configurable)
- Révocation possible des tokens en cas de compromission
- Renouvellement automatique des tokens avant expiration

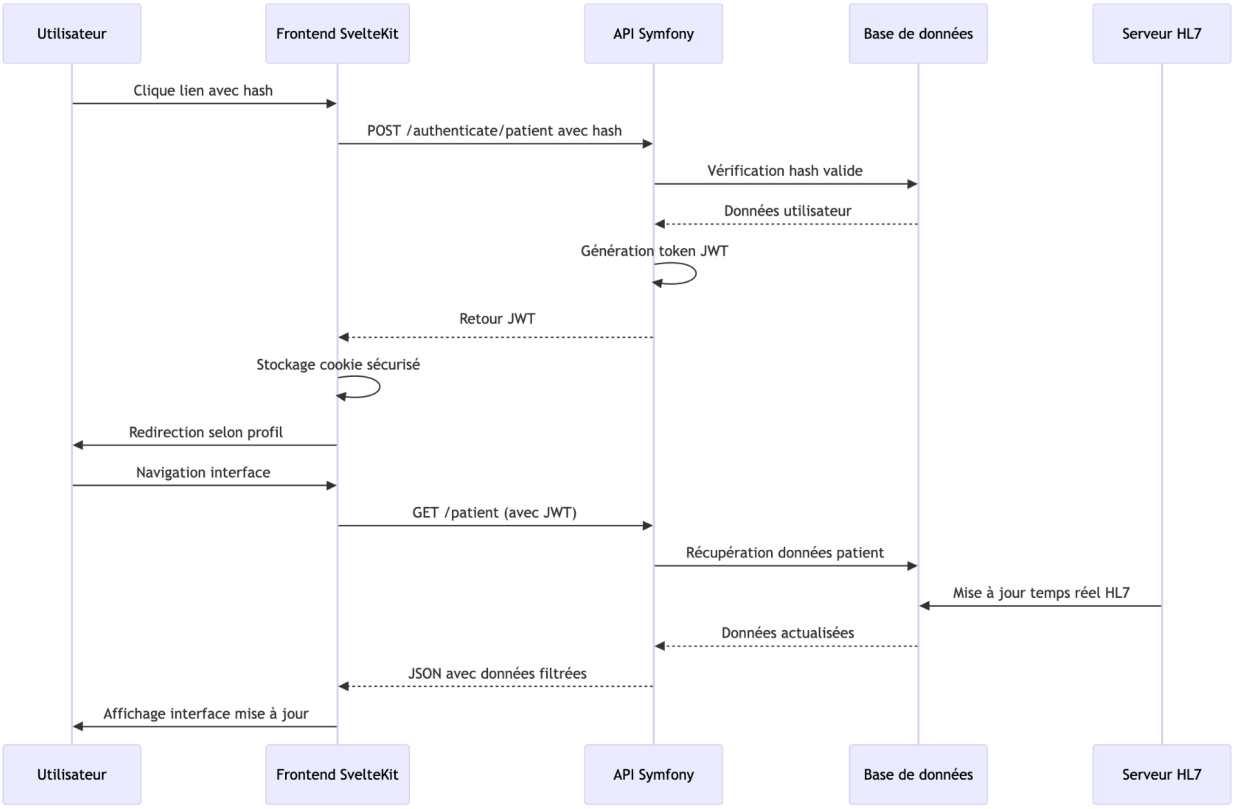


Figure 8. Flux d'authentification JWT

Analyse du flux :

1. Phase d'authentification (étapes 1-7) : L'utilisateur accède via un lien sécurisé, le système valide le hash et génère un JWT
2. Phase de navigation (étapes 8-12) : Les requêtes ultérieures utilisent le JWT pour l'autorisation, avec mise à jour temps réel des données patient via HL7

Mesures de sécurité avancées

Prévention des attaques :

- Protection contre les attaques CSRF via tokens anti-CSRF
- Limitation du taux de requêtes (rate limiting)
- Logs de sécurité pour audit et détection d'anomalies
- Validation stricte des données HL7 avant intégration

Conformité RGPD :

- Pseudonymisation des données patient
- Traçabilité des accès aux données sensibles
- Possibilité de révocation d'accès immédiate

Cette architecture garantit une sécurité robuste tout en maintenant une expérience utilisateur fluide et des performances optimales pour les mises à jour temps réel des données patient.

6. Devops

Pour le déploiement de l'application Floween, j'ai adopté une démarche structurée, privilégiant la fiabilité et l'automatisation pour éviter les erreurs humaines et garantir une homogénéité parfaite entre les environnements de développement et de production.

La chaîne CI/CD repose sur la construction d'images Docker, selon un modèle en deux étapes : une première étape dédiée à la construction de l'application, suivie d'une étape de production où seules les ressources nécessaires sont embarquées. Ce schéma garantit à la fois un déploiement rapide et des images légères, optimisées pour la production.

Le fichier Dockerfile mis en place est le suivant :

```
# Étape de build
FROM node:24-alpine AS build
RUN npm install -g pnpm
WORKDIR /app
COPY package.json pnpm-lock.yaml ./
RUN pnpm install --frozen-lockfile
COPY . .
RUN pnpm build

# Étape de production
FROM node:24-alpine AS production
WORKDIR /app
COPY --from=build /app/package.json /app/pnpm-lock.yaml ./
RUN npm install -g pnpm
RUN pnpm install --prod --frozen-lockfile
COPY --from=build /app/build ./build
ENV NODE_ENV=production
ENV PORT=3000
EXPOSE 3000
CMD ["node", "build"]
```

Figure 9. Dockerfile multi couche

- **Découpage en deux étapes** : La première étape effectue toutes les opérations de compilation. Cela permet de bénéficier du cache Docker lors de l'installation des dépendances, réduisant ainsi la durée des builds lors des modifications incrémentales.
  La seconde étape (« production ») ne conserve que les fichiers strictement nécessaires à l'exécution de l'application compilée.

- **Utilisation de pnpm** : Ce gestionnaire est privilégié pour sa rapidité et son mode de gestion optimisé des dépendances.

- **Séparation des dépendances** : On distingue clairement les dépendances de build (étape 1) et celles nécessaires en production (étape 2). Seules ces dernières sont installées dans l'image finale, ce qui limite la surface d'attaque, réduit la taille du conteneur et accélère le démarrage.

- **Variables d'environnement et port** : L'environnement est explicitement renseigné en mode « production », avec exposition du port 3000.

Déploiement

Une fois l'image Docker générée et poussée sur le registre, le déploiement sur le serveur de production est réalisé via une interface de gestion de conteneurs (Portainer), ce qui permet :

- un redéploiement rapide sans manipulation directe du serveur,

- la possibilité de rollback immédiat en cas d'incident,

- un suivi et monitoring centralisés.

Ce fonctionnement s'inscrit pleinement dans le cadre des pratiques DevOps actuelles et contribue à sécuriser la mise en production de Floween tout en garantissant la reproductibilité des environnements.

7. Tests

La phase de tests de l'application Floween a été menée exclusivement en interne durant cette période d'alternance, en préparation de la démonstration prévue à l'Hôpital de Monaco vers mi-septembre. Cette approche méthodique nous permet de valider l'ensemble des fonctionnalités développées avant la présentation officielle auprès d'un établissement de santé partenaire.

7.1. Environnement de tests et déploiement

L'infrastructure DevOps développée précédemment s'est révélée particulièrement efficace pour faciliter la phase de tests. L'utilisation de Docker et de l'interface Portainer a permis de déployer rapidement les nouvelles versions de l'application sur l'environnement de test, accélérant considérablement les cycles de validation.

Cette approche technique présente des avantages significatifs par rapport aux tests en environnement local. Chaque correction ou amélioration fonctionnelle peut être déployée immédiatement sur le serveur de test, permettant à l'équipe de valider les modifications dans des conditions proches de la production. Cette méthodologie réduit les risques de dysfonctionnements lors de la mise en production finale.

La gestion des données de test a constitué un défi technique majeur. Au moment du développement du frontend, l'API backend n'était pas encore finalisée, nécessitant la création de jeux de données fictives pour permettre les tests d'intégration. J'ai utilisé des outils d'intelligence artificielle pour générer des données patient cohérentes et représentatives des différents parcours aux urgences : consultations directes, examens complémentaires, temps d'attente variables, et scenarios de sortie diversifiés.

Cette approche présentait également l'avantage de respecter scrupuleusement les contraintes RGPD. L'utilisation de véritables données patient pour les tests aurait constitué une violation des principes de confidentialité médicale, même dans un contexte de développement interne. Les données synthétiques permettent de tester l'ensemble des cas d'usage sans compromettre la protection des données personnelles, conformément aux exigences réglementaires du secteur de la santé.

8. Évolution

Il est prévu de développer des versions natives mobiles de Floween afin de pallier les limitations inhérentes aux applications web progressives (PWA) concernant la gestion des notifications push. En effet, les navigateurs mobiles imposent des restrictions strictes sur l'affichage de notifications en arrière-plan depuis une application web, particulièrement lorsque l'utilisateur n'interagit pas activement avec l'interface. Cette contrainte technique nécessite le développement d'une application native installée qui exploite directement les SDK des systèmes d'exploitation iOS et Android pour accéder aux services de notifications push (APNs pour Apple et FCM pour Google) et garantir un fonctionnement optimal même lorsque l'application n'est pas au premier plan.

Tauri se présente comme un excellent candidat pour cette évolution technologique. Il s'agit d'un framework moderne de développement d'applications mobiles natives qui permet de créer des applications iOS et Android en utilisant des technologies web (HTML, CSS, JavaScript/TypeScript) pour l'interface utilisateur, tout en bénéficiant d'un cœur Rust performant pour la logique métier et l'accès aux APIs système. Cette architecture hybride offre le meilleur des deux mondes : la rapidité de développement des technologies web et les performances natives des langages système.

L'avantage majeur de Tauri dans notre contexte réside dans le fait que l'interface utilisateur de Floween est déjà entièrement développée et optimisée. Cette UI existante pourra être directement réutilisée sans modifications majeures, ce qui accélérera considérablement le processus de développement des versions mobiles natives. Concrètement, Tauri encapsule notre frontend web dans un conteneur natif léger, permettant l'accès aux APIs spécifiques des systèmes d'exploitation mobiles comme les notifications push, le stockage local sécurisé, l'accès à la caméra, ou encore l'intégration avec les services système.

Tauri présente des avantages significatifs par rapport à ses alternatives mobiles. Contrairement à des solutions hybrides comme Cordova/PhoneGap qui utilisent une WebView système souvent lente et limitée, Tauri exploite un runtime natif optimisé offrant des performances supérieures. Face à React Native qui nécessiterait une réécriture complète de l'interface utilisateur avec des composants spécifiques (FlatList au lieu de div, View au lieu de section) et une logique de navigation entièrement différente (React Navigation), Tauri permet de conserver intégralement le code frontend existant. De plus, React Native souffre de limitations dans l'écosystème des bibliothèques tierces, nécessite souvent des bridges natifs complexes pour accéder aux fonctionnalités système, et génère des applications plus lourdes. Tauri offre un accès direct et sécurisé aux APIs mobiles via son architecture Rust, une taille d'application réduite, et un système de permissions

granulaires qui contrôle précisément l'accès aux ressources du smartphone, tout en garantissant une expérience utilisateur native fluide.

9. Performance et optimisation

L'application Floween a été conçue pour offrir une expérience fluide, même dans des conditions de réseau limitées, ce qui est essentiel dans un contexte hospitalier. Différents scénarios de connectivité ont été testés, notamment avec une connexion fibre optique en wifi, 4G lente et 3G. Les résultats confirment que l'architecture retenue (Svelte Kit côté front et API Symfony) permet d'obtenir des temps de chargement courts, une interface réactive et un usage efficace du cache.

Métrique de Performance	Fibre Optique wifi (Sans Cache 300 mbit/s)	4G Lente (Sans Cache 1 638,4 kbit/s)	4G Lente (Cache SW 1 638,4 kbit/s)	3G (Sans Cache 750 kbit/s)
LCP (ms)	280	1 474	635	4921
FCP (ms)	280	1 474	635	4921
TTFB (ms)	110	580	580	1900
TBT (ms)	0	0	0	0
Taille HTML (Ko)	39,97	39,97	39,97	39,97
Poids Total transférés (Ko)	187,96	187,96	8	187,96
Poids Total (Ko)	655	655	655	655
Nb. Requêtes	33	33	6	33

Figure 10. Tableau de performance des différentes connexions

Décryptage des principaux indicateurs :

LCP (Largest Contentful Paint)

Le LCP est un indicateur clé car il correspond au temps nécessaire pour afficher l'élément principal à l'écran (typiquement un bloc d'informations ou un visuel marquant). À 280ms avec une connexion fibre optique en wifi, c'est quasiment instantané. Même en 4G lente, passer à 1,5s reste bon (Google recommande <2,5s). En 3G lente par contre, à presque 5s, on voit bien la limite imposée par la bande passante et la latence.

Commentaire : Ce qui est intéressant, c'est le gain observé avec le cache Service Worker : sur 4G lente, on divise le LCP par plus de deux (de 1,5s à 635ms). Cela montre que l'architecture PWA de Floween n'est pas juste pour la déco, elle a un impact immédiat en conditions réelles (mobilité, couloirs bétonnés, etc.).

FCP (First Contentful Paint)

Les valeurs sont identiques au LCP, ce qui indique que tout le contenu utile s'affiche d'un coup, sans attendre des scripts tiers ou des composants lents. Pour l'utilisateur, il n'y a pas cet effet frustrant de "squelette" ou de zone vide en attendant le chargement complet.

TTFB (Time To First Byte)

Avec 110ms avec une connexion fibre optique en wifi et 580ms sur mobile, ça montre que le backend Symfony, l'infrastucture Dockerisée et la base de données sont bien maîtrisés. Même en 3G lente, avec 1,9s, on reste dans le raisonnable pour ce type de contexte médical (où il est parfois impossible de faire mieux côté infrastructure réseau nationale).

TBT (Total Blocking Time)

Affiché à 0ms partout. Donc, aucune fonction JavaScript ou rendu ne bloque jamais le fil principal : l'utilisateur peut toujours interagir immédiatement, sans freeze ni lag lors du chargement. Ce n'est vraiment pas gagné d'avance sur des frameworks modernes, donc là-dessus, l'intégration Svelte Kit est clairement réussie.

Taille et poids des ressources, effet du cache

- Les ressources HTML et médias restent modestes (<40Ko d'HTML ; 655Ko total), ce qui garantit une accessibilité rapide.

- Le poids transféré chute à 8Ko en 4G lente avec le cache actif ! Autrement dit, si l'utilisateur a déjà ouvert l'application une fois, il n'a quasiment plus à charger de ressources, même en cas de déconnexion temporaire ou de rechargement hors couverture.

- La réduction radicale du nombre de requêtes, de 33 à 6, avec le Service Worker explique aussi les gains de rapidité constatés. Cela réduit la congestion sur les réseaux mobiles et abaisse la latence ressentie.

Pour résumer, Floween garde des bons temps de chargement même sur des réseaux dégradés, et l'emploi du cache améliore nettement l'expérience utilisateur.

Globalement, les résultats montrent que l'application est robuste et optimisée face à la variabilité des réseaux, et que les efforts d’optimisation (architecture moderne,

cache, réduction du poids des ressources) portent vraiment leurs fruits dans la pratique.

10. Conclusion

Cette deuxième année d'alternance a marqué un véritable tournant dans mon parcours professionnel. J'ai progressivement quitté mon statut de développeur Android spécialisé pour adopter un profil Full Stack, élargissant ainsi mon champ de compétences au backend (PHP/Symfony, API REST), mais aussi aux frameworks frontend modernes comme SvelteKit. Ce changement de périmètre s'est accompagné de l'apprentissage et de la maîtrise de nouvelles technologies, tant sur la partie mobile que web.

Par ailleurs, ma participation active aux réunions techniques m'a permis d'apporter mes connaissances et de conseiller l'équipe sur le choix de frameworks modernes et de bonnes pratiques d'architecture logicielle. On m'a confié des tâches stratégiques nécessitant analyse technologique et choix structurants, ce qui m'a permis de gagner en légitimité au sein de l'équipe.

Cette montée en compétences s'est également traduite par une augmentation de mon autonomie : j'ai pu prendre davantage d'initiatives et mener à bien, en toute confiance, des missions à plus forte responsabilité, comme le pilotage du développement du frontend d'un nouveau produit ou la conception de modules backend inédits.

Récapitulatif de ma première année

Durant ma première année d'alternance, j'ai principalement travaillé en tant que développeur Android sur le SDK SmartCity. Mes principales réalisations comprenaient :

Développement de modules SmartCity :

- Travail sur diverses applications pour les villes : Vitaboucle, Vercors, Dole, Ampi, Chor, Ma Ville Facile, Corsaire
- Développement d'un système d'authentification OpenID pour permettre aux utilisateurs de se connecter via les sites des collectivités
- Création de modules personnalisés selon les besoins spécifiques des clients

ToolBox V2 :

- Développement complet d'une application interne de A à Z en utilisant Jetpack Compose
- Migration d'une architecture MVP vers MVVM moderne

- Implémentation d'un mode hors ligne avec synchronisation des données
- Gestion des capteurs du téléphone pour la collecte de données de pression atmosphérique

Intégration de nouvelles technologies :

- Introduction de Jetpack Compose dans l'architecture SmartCity existante
- Création d'un système de callbacks permettant d'intégrer des composants personnalisés sans impacter les autres applications
- Développement d'une solution technique pour éviter les problèmes de surcharge de layouts XML

Récapitulatif sur les 24 mois

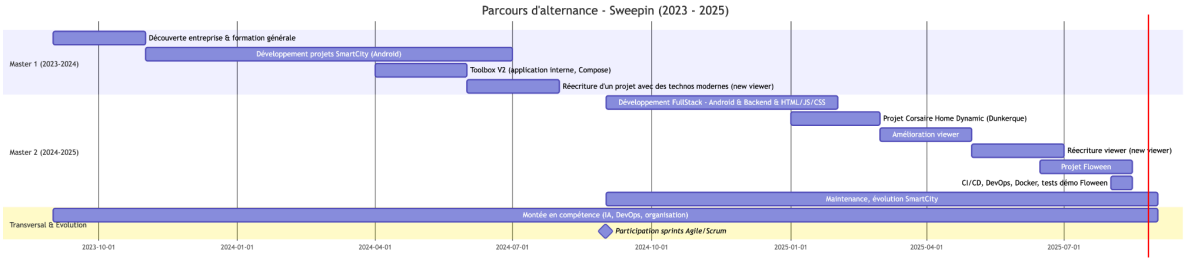


Figure 11. Diagramme de gantt générale sur les 2 années au sein de Sweepin

Évolution et montée en compétences

Évolution du périmètre de responsabilités :

Ma progression durant ces deux années d'alternance illustre une montée en compétences progressive et constante. En première année, mes responsabilités étaient exclusivement centrées sur le développement Android avec une supervision étroite de mon tuteur. Cette période m'a permis d'acquérir une maîtrise solide du développement mobile moderne, notamment Kotlin et les architectures recommandées par Google.

Suite au départ de mon tuteur initial, j'ai progressivement gagné en autonomie et en responsabilités. Cette transition s'est accompagnée d'un élargissement significatif de mon périmètre technique : de développeur Android spécialisé, je suis devenu développeur full-stack avec des compétences backend en PHP/Symfony.

Compétences techniques développées :

Côté frontend/mobile :

- Maîtrise approfondie de Kotlin et des architectures Android modernes (MVVM, Jetpack Compose)
- Expertise en Jetpack Compose pour des interfaces utilisateur réactives
- Développement web avec Svelte Kit
- Intégration d'APIs REST et WebSockets

Côté backend :

- Développement PHP avec les frameworks Symfony et Zend
- Conception et implémentation d'APIs REST
- Maîtrise de Doctrine ORM pour la gestion des données
- Architecture MVC et bonnes pratiques de développement serveur

DevOps et déploiement :

- Containerisation avec Docker et Docker Compose
- Pipelines CI/CD automatisés
- Gestion des environnements de développement, staging et production

Compétences organisationnelles :

Au-delà des aspects purement techniques, cette alternance m'a permis de développer des compétences organisationnelles essentielles :

- **Gestion de projet :** participation aux sprints Agile, estimation des charges de travail, respect des délais
- **Communication technique :** interaction avec différentes équipes (iOS, backend, communication)
- **Auto-formation :** capacité à apprendre rapidement de nouvelles technologies et frameworks
- **Résolution de problèmes :** autonomie dans l'identification et la résolution de difficultés techniques

Impact de l'intelligence artificielle :

L'intégration des outils d'IA dans mon processus de développement a considérablement amélioré ma productivité. GitHub Copilot est devenu un outil indispensable pour l'autocomplétion intelligente, tandis que ChatGPT et Claude m'aident pour le debugging et l'apprentissage de nouvelles technologies. Ces outils m'ont particulièrement aidé dans ma montée en compétences sur PHP/Symfony.

Bilan personnel :

Cette alternance de deux ans chez Sweepin m'a transformé d'étudiant en informatique en développeur full-stack autonome et polyvalent. J'ai acquis une vision complète du cycle de développement logiciel, de la conception à la mise en production, tout en développant une expertise technique solide sur plusieurs technologies.

La diversité des missions m'a permis d'explorer différents aspects du développement : applications mobiles natives, applications web modernes, APIs backend, DevOps. Cette polyvalence constitue un atout majeur pour mon avenir professionnel.

Enfin, l'environnement startup de Sweepin m'a offert des responsabilités importantes dès la première année, me préparant efficacement aux défis du monde professionnel. Cette expérience m'a donné confiance en mes capacités et m'a préparé à évoluer dans des environnements techniques exigeants.

Cette alternance constitue indéniablement une base solide pour ma carrière de développeur, m'ayant doté des compétences techniques, organisationnelles et humaines nécessaires pour réussir dans le domaine du développement logiciel moderne.

Résumé

Ce rapport retrace mon expérience d'alternance au sein de l'entreprise Sweepin au cours de ma deuxième année de master. En tant qu'étudiant en Master Informatique, spécialisé en Bases de Données et Intelligence Artificielle, j'ai pu mettre en pratique mes connaissances théoriques tout en développant de nouvelles compétences techniques dans un environnement professionnel stimulant.

Durant cette période, j'ai occupé le poste de développeur Full Stack, avec une double spécialisation en développement frontend et backend. Cette polyvalence m'a permis d'intervenir sur l'ensemble de la chaîne de développement d'un nouveau produit innovant : Floween, une application web progressive de suivi des patients aux urgences hospitalières.

Le rapport détaille particulièrement ma mission principale sur le projet Floween, une application de suivi patient en temps réel qui vise à concurrencer la solution existante FollowMe Urgences. Cette réalisation a nécessité la conception complète d'une architecture moderne incluant un frontend en SvelteKit SSR avec TypeScript, un backend API en PHP 8.3 avec Symfony 7, une base de données MariaDB, et l'intégration de données HL7 via un serveur WildFly. L'application a été conçue comme une PWA (Progressive Web App) internationalisée et exclusivement mobile.

Cette alternance m'a permis de développer des compétences techniques approfondies dans des technologies modernes, mais également des aptitudes organisationnelles et méthodologiques grâce au travail en équipe et à la méthodologie Agile. De plus, j'ai pu explorer les défis spécifiques du développement d'applications dans le domaine de la santé, incluant la sécurité des données, l'interopérabilité des systèmes, et l'expérience utilisateur en situation d'urgence médicale.

Cette expérience a constitué une étape déterminante dans mon parcours professionnel, me préparant efficacement aux défis du développement logiciel moderne dans un secteur critique, tout en renforçant mon profil de développeur Full Stack spécialisé dans les technologies web avancées.

11. Bibliographie

PHP – Présentation générale
https://en.wikipedia.org/wiki/PHP

Symfony – Documentation officielle
https://symfony.com/doc/current/index.html

Doctrine ORM
https://www.doctrine-project.org/projects/orm.html

REST API – Principes de base
https://blog.dreamfactory.com/rest-apis-an-overview-of-basic-principles

Méthodologie Agile
https://agilemanifesto.org/

ClickUp – Outil de productivité agile
https://clickup.com/

Tauri – Framework natif moderne
https://github.com/tauri-apps/tauri

Licence Apache 2
http://www.apache.org/licenses/

JWT (JSON Web Token) – Introduction et spécifications
https://jwt.io/introduction
https://datatracker.ietf.org/doc/html/rfc7519

Node.js – Guide du serveur JavaScript
https://nodejs.org/en/docs/guides

Service Workers et PWA – web.dev
https://web.dev/learn/pwa/

User-Agent — Guide de détection côté serveur
https://developer.mozilla.org/fr/docs/Web/HTTP/Headers/User-Agent

Progressive Web Apps – Documentation Google
https://developer.chrome.com/blog/getting-started-pwa?hl=fr

Flowbite – Librairie UI pour Svelte
https://flowbite.com/docs/getting-started/svelte/

SvelteKit – Documentation
https://kit.svelte.dev/docs

TailwindCSS – Documentation
https://tailwindcss.com/docs

HL7 – Norme d'échange hospitalière
https://www.hl7.org/

RGPD – Règlement Général sur la Protection des Données
https://www.cnil.fr/fr/rgpd-guide

JWT & Cookies – Sécurité et bonnes pratiques
https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html