



Ministère de l'Enseignement
Supérieur et de la Recherche
Scientifique

Année Universitaire
2025/2026



RAPPORT DE PROJET DE FIN D'ANNÉE

Spécialité: Ingénierie Logicielle

Intitulé

Développement d'une application éducative pour
enfants diabétiques <<SugarHero>>

Réalisé par

Ines CHAREF

Encadré par

Marwa NEILY

Dédicace

En termes de reconnaissance de leurs sacrifices et en témoignage de la profondeur de mes sentiments à leur égard, j'ai l'honneur de dédier ce modeste travail.

À ma chère mère

Ma raison d'être, ma raison de vivre, la lanterne qui éclaire mon chemin et m'illumine de douceur et d'amour.

À mon cher père

En signe d'amour, de reconnaissance et de gratitude pour tous les sacrifices dont il a fait preuve à mon égard.

À mes chers frères

Pour leurs encouragements. Je les souhaite un avenir plein de bonheur, de succès et de prospérité.

À toute ma Famille...

À tous ceux qui m'aiment...

Remerciements

Je commence par rendre grâce à Dieu pour la santé, la motivation et tout ce qu'Il m'a accordé durant la réalisation de ce modeste travail.

C'est avec un grand plaisir que je réserve ces quelques lignes en signe de profonde reconnaissance à tous ceux qui, de près ou de loin, m'ont apporté leurs contributions dans la réalisation de ce travail.

Tout d'abord, mes meilleurs et sincères remerciements vont au jury pour avoir accepté d'évaluer ce projet, auxquels nous présentons notre gratitude et notre profond respect.

De même des messages de respect sont à adresser aux enseignants de l'EPI et en particulier notre enseignant encadrant Mm. Neily Marwa pour l'attention qu'il a apportée à notre projet tout au long de ses divers stades allant de la conception à la réalisation et pour ces précieux conseils.

Liste des abréviations

- Glucides : **P**rincipale source d'énergie de l'organisme (simples = sucres rapides, complexes = sucres lents), régulant la glycémie
- HbA1c : **c**'est un test montre la moyenne de ta glycémie sur 3 mois
- Glycémie : **T**aux de glucose dans le sang
- Hypoglycémie : **U**ne concentration élevée de glucose dans le sang.
- Hyperglycémie : **c**oncentration de glucose dans le sang est anormalement basse.
- Insuline : **U**ne hormone produite par le pancréas qui permet au glucose d'entrer dans les cellules pour être utilisé comme source d'énergie
- UML : **UModeling *L*anguage**
- CRUD : **C**reate, **R**ead, **U**pdate, **D**elete
- 2TUP : **T**wo-Tier Upgrade Plan
- UI : **U**ser **I**nterface
- MVC : **M**odel-View-Controller
- MVP : **M**odel-View-Presenter
- MVVM : **M**odel-View-ViewModel

Table des matières

| | |
|--|-----------|
| Introduction Générale | 1 |
| 1 Étude du Contexte et Spécification des Besoins | 2 |
| 1.1 Problématique | 2 |
| 1.2 Étude de l'Existant | 2 |
| 1.2.1 Analyse de l'Existant | 3 |
| 1.2.2 Critique de l'Existant | 4 |
| 1.2.3 Solution Proposée | 4 |
| 1.3 Identification des Besoins | 4 |
| 1.3.1 Identification des Acteurs | 4 |
| 1.3.2 Besoins Fonctionnels | 5 |
| 1.3.3 Besoins Non Fonctionnels | 6 |
| 1.4 Méthodologies de Conception | 6 |
| 1.4.1 Méthodologie Classique | 6 |
| 1.4.2 Méthodologie Agile | 7 |
| 1.4.3 Processus Unifié | 7 |
| 1.4.4 Comparaison des Méthodes et Choix | 8 |
| 1.4.5 Mise en pratique | 9 |
| 2 Étude Conceptuelle | 11 |
| Conception | 11 |
| 2.1 Introduction | 11 |
| 2.2 Architecture physique | 11 |
| 2.2.1 Architecture à deux niveaux (2-tiers) | 11 |
| 2.2.2 Architecture à trois niveaux (3-tiers) | 12 |
| 2.2.3 Architecture à plusieurs niveaux (N-tiers) | 13 |
| 2.2.4 Choix de l'architecture physique | 15 |
| 2.3 Architecture logique | 15 |
| 2.3.1 Architecture MVC (Model-View-Controller) | 15 |
| 2.3.2 Architecture MVP (Model-View-Presenter) | 16 |
| 2.3.3 Architecture MVVM (Model-View-ViewModel) | 17 |
| 2.3.4 Choix de l'architecture de l'application | 18 |
| 2.4 Langage de Conception | 19 |
| 2.4.1 Langage UML | 19 |

| | | |
|----------------------------|---|-----------|
| 2.5 | Diagrammes UML | 19 |
| 2.6 | Diagramme de cas d'utilisation | 20 |
| 2.6.1 | Diagramme de cas d'utilisation globale | 21 |
| 2.6.2 | Raffinement des cas d'utilisation | 22 |
| 2.6.3 | Raffinement des cas d'utilisation «S'authentifier» | 23 |
| 2.6.4 | Raffinement des cas d'utilisation «créer un compte » | 25 |
| 2.6.5 | Raffinement du cas d'utilisation : Suivre la glycémie | 27 |
| 2.6.6 | Raffinement du cas d'utilisation : "Gérer les rappels de traitement" | 29 |
| 2.6.7 | Raffinement du cas d'utilisation : "Jouer à un quiz éducatif" | 31 |
| 2.7 | Diagrammes de séquences | 32 |
| 2.7.1 | Diagramme de séquence «Inscription» | 32 |
| 2.7.2 | Diagramme de séquence «mot de passe oublier» | 33 |
| 2.8 | Diagrammes de classe | 34 |
| 3 | Réalisation | 36 |
| 3.1 | Environnement du travail | 36 |
| 3.1.1 | Environnement matériel | 36 |
| 3.1.2 | Environnement logiciel | 36 |
| 3.2 | Présentation des interfaces de l'application | 40 |
| 3.2.1 | Interface authentification | 40 |
| 3.2.2 | Interface utilisateur | 44 |
| 3.2.3 | Interface administrateur | 50 |
| Conclusion Générale | | 55 |
| Webographie | | 56 |

Table des figures

| | | |
|----|--|----|
| 1 | MySugr | 2 |
| 2 | Glucose Buddy | 3 |
| 3 | Diabetes :M | 3 |
| 4 | Méthode en V | 7 |
| 5 | Modèle Scrum | 7 |
| 6 | Two Track Unified Process (2TUP) | 8 |
| 7 | Le processus 2TUP | 9 |
| 8 | Architecture 2-tiers | 12 |
| 9 | Architecture 3-tiers | 13 |
| 10 | Architecture à plusieurs niveaux (N-tiers) | 14 |
| 11 | Architecture MVC | 16 |
| 12 | Architecture MVP | 17 |
| 13 | Architecture MVVM | 18 |
| 14 | Diagramme de cas d'utilisation globale | 22 |
| 15 | Diagramme des cas d'utilisation créer un compte | 23 |
| 16 | Diagramme des cas d'utilisation S'authentifier | 25 |
| 17 | Diagramme des cas d'utilisation Suivre la glycémie | 27 |
| 18 | Diagramme de cas d'utilisation : "Gérer les rappels de traitement" | 29 |
| 19 | Diagramme de cas d'utilisation : "Jouer à un quiz éducatif" | 31 |
| 20 | Diagramme de séquence «Inscription» | 33 |
| 21 | Diagramme de séquence mot de passe oublier | 34 |
| 22 | Diagramme de classe | 35 |
| 23 | Logo React | 37 |
| 24 | Logo Node.js | 37 |
| 25 | Logo Express.js | 37 |
| 26 | Logo MongoDB | 38 |
| 27 | Logo VS Code | 38 |
| 28 | Logo Postman | 38 |
| 29 | Logo Lucidchart | 39 |
| 30 | Logo LaTeX | 39 |
| 31 | interface création d'un compte | 40 |
| 32 | interface saisie informations médicales | 41 |
| 33 | interface d'authentification | 41 |
| 34 | interface mot de passe oublier | 42 |
| 35 | interface mot de passe oublier | 42 |

| | | |
|----|---|----|
| 36 | interface mot de passe oublier | 43 |
| 37 | interface mot de passe oublier | 43 |
| 38 | interface de fiche educatives | 44 |
| 39 | interface suivi de glycémie | 44 |
| 40 | interface suivi de glycémie | 45 |
| 41 | interface de rappel | 45 |
| 42 | interface de rappel | 46 |
| 43 | interface de quiz | 46 |
| 44 | interface de quiz | 47 |
| 45 | interface de quiz | 48 |
| 46 | interface régime suivi repas | 49 |
| 47 | interface régime suivi activité physique | 49 |
| 48 | interface de profil | 50 |
| 49 | interface de profil | 50 |
| 50 | interface gestion des fiches éducatives | 51 |
| 51 | interface gestion des fiches éducatives | 51 |
| 52 | interface gestion des quiz | 52 |
| 53 | interface gestion des quiz ajouter un quiz | 52 |
| 54 | interface gestion des quiz modifier un quiz | 53 |
| 55 | interface gestion des utilisateurs | 53 |
| 56 | interface gestion des utilisateurs"ajouter un utilisateur | 54 |

Liste des tableaux

| | | |
|---|--|----|
| 1 | Tableau comparatif des méthodologies de conception | 8 |
| 2 | Tableau descriptif de cas d'utilisation : « S'authentifier » | 24 |
| 3 | Tableau descriptif de cas d'utilisation : « Créer un compte » | 26 |
| 4 | Tableau descriptif de cas d'utilisation : Suivre la glycémie | 28 |
| 5 | Tableau descriptif : "Gérer les rappels de traitement" | 30 |
| 6 | Tableau descriptif du cas d'utilisation : "Jouer à un quiz éducatif" | 32 |
| 7 | Environnement matériel | 36 |

Introduction Générale

Dans un monde où les technologies numériques transforment radicalement le secteur de la santé, la prise en charge des maladies chroniques chez les enfants représente un défi majeur. Le diabète de type 1, affectant particulièrement les jeunes patients, nécessite une gestion quotidienne rigoureuse impliquant surveillance glycémique, calcul des doses d'insuline et suivi nutritionnel. Pourtant, les solutions logicielles existantes peinent à répondre aux besoins spécifiques de cette population jeune, souvent confrontée à des interfaces complexes et peu engageantes.

Le diabète infantile représente un défi quotidien pour les enfants, leurs familles et les professionnels de santé. Cette maladie chronique exige une gestion rigoureuse qui repose sur un suivi régulier de la glycémie, un contrôle strict de l'alimentation et l'administration ponctuelle de médicaments ou d'insuline. Pour un enfant, ces exigences peuvent être difficiles à comprendre et à appliquer, rendant la gestion du diabète particulièrement complexe.

Face à ces défis, nous avons conçu **SugarHero**, une application web innovante spécialement dédiée aux jeunes diabétiques. Plus qu'un simple outil de suivi, SugarHero se veut un compagnon interactif et éducatif, transformant la gestion quotidienne du diabète en une expérience ludique et motivante. À travers son interface intuitive, ses fonctionnalités adaptées et son approche pédagogique, notre solution vise à :

- **Simplifier** le suivi médical pour les enfants
- **Responsabiliser** les jeunes patients par l'apprentissage
- **Soulager** les familles dans le suivi au quotidien
- **Maintenir** l'observance thérapeutique sur le long terme

Ce projet de fin d'année incarne notre volonté de mettre la technologie au service d'une cause humaine, en créant un outil qui change concrètement le vécu des enfants diabétiques. Les pages qui suivent détailleront notre démarche complète, depuis l'analyse des besoins jusqu'à la réalisation technique, en passant par les choix innovants qui font de SugarHero une solution à la fois pratique et émotionnellement engageante.

Parce que grandir avec le diabète ne devrait pas empêcher de vivre pleinement son enfance.

1 Étude du Contexte et Spécification des Besoins

1.1 Problématique

Le diabète est une maladie chronique qui exige une gestion rigoureuse et quotidienne. Pour les enfants, cette gestion est souvent compliquée par leur jeune âge et leur manque de compréhension des enjeux liés à leur santé. Les parents et les professionnels de santé sont fréquemment confrontés à des difficultés pour éduquer les enfants sur les bonnes pratiques à adopter. Les problèmes identifiés incluent :

- **Manque d'outils éducatifs adaptés** : Les applications existantes sont souvent trop complexes ou ne sont pas conçues pour les enfants.
- **Difficultés de suivi quotidien** : Les enfants ont du mal à suivre leur glycémie, leurs repas et leurs activités physiques de manière régulière.
- **Absence de motivation** : Les enfants peuvent se lasser rapidement des routines strictes liées à la gestion du diabète.

Si ces problèmes ne sont pas résolus, les enfants risquent de ne pas bien gérer leur diabète, ce qui peut entraîner des complications de santé à long terme. Il est donc essentiel de proposer une solution adaptée pour les aider à mieux comprendre et gérer leur maladie.

1.2 Étude de l'Existant

Plusieurs applications existent pour aider à la gestion du diabète, mais elles ne sont pas toujours adaptées aux enfants. Voici quelques exemples d'applications existantes :

- **MySugr** : Une application de suivi du diabète qui permet aux utilisateurs d'enregistrer leur glycémie, leurs repas et leurs activités physiques. Cependant, elle est principalement conçue pour les adultes.



Figure 1 – MySugr

- **Glucose Buddy** : Une application de suivi de la glycémie qui offre des fonctionnalités similaires, mais qui manque d'éléments éducatifs pour les enfants.



Figure 2 – Glucose Buddy

- **Diabetes :M** : Une application complète pour la gestion du diabète, mais qui est trop complexe pour les enfants.

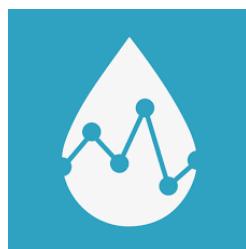


Figure 3 – Diabetes :M

1.2.1 Analyse de l’Existant

Les applications existantes offrent un suivi détaillé de la glycémie, des repas et des activités physiques. Elles sont largement disponibles sur les plateformes mobiles, mais présentent plusieurs limites :

- **Peu d’interactivité et de motivation pour les enfants** : Les applications existantes ne sont pas conçues pour engager les enfants de manière ludique.
- **Absence de gamification** : Il manque des éléments de jeu pour encourager l’apprentissage et la participation.
- **Suivi souvent orienté vers les parents ou les professionnels de santé** : Les applications ne sont pas centrées sur les besoins des enfants.
- **Difficulté d’accès et manque d’adaptabilité** : Les interfaces ne sont pas toujours intuitives pour les jeunes utilisateurs.
- **Contenu éducatif limité et peu interactif** : Les informations ne sont pas présentées de manière attrayante pour les enfants.
- **Systèmes de rappels rigides et peu personnalisables** : Les rappels ne sont pas adaptés aux besoins individuels des enfants.

1.2.2 Critique de l'Existant

Les solutions existantes présentent plusieurs limites :

- **Motivation limitée** : Les enfants peuvent se lasser rapidement des outils qui ne sont pas conçus pour eux.
- **Interfaces inadaptées** : Les interfaces ne sont pas toujours intuitives pour les jeunes utilisateurs.
- **Contenu éducatif insuffisant** : Les informations ne sont pas présentées de manière ludique et éducative.
- **Rappels peu personnalisés** : Les systèmes de rappels ne sont pas adaptés aux besoins individuels des enfants.

1.2.3 Solution Proposée

Pour surmonter les limites des solutions existantes, nous proposons de développer un site web interactif spécialement conçu pour les enfants diabétiques et leurs parents. Notre solution se distingue par les éléments suivants :

- **Spécialisation** : Le site web est entièrement dédié aux enfants diabétiques, avec une approche adaptée à leur âge et à leurs besoins spécifiques.
- **Interface éducative** : Des fiches illustrées et des quiz interactifs aident les enfants à mieux comprendre leur diabète.
- **Suivi quotidien** : Les enfants peuvent enregistrer leur glycémie, leurs repas dans un journal de bord interactif.
- **Gestion alimentaire** : Un suivi nutritionnel complet permet d'enregistrer les repas, avec calcul automatique des glucides et calories, pour une alimentation équilibrée adaptée au diabète.
- **Rappels et notifications** : Des notifications personnalisées rappellent aux enfants de prendre leurs médicaments ou de mesurer leur glycémie.
- **Section ludique** : Des quiz éducatifs et des récompenses motivent les enfants à participer activement à la gestion de leur diabète.

1.3 Identification des Besoins

1.3.1 Identification des Acteurs

Les acteurs principaux de ce projet sont :

- **Enfants diabétiques** : Les utilisateurs principaux du site web, qui doivent pouvoir suivre leur glycémie, participer à des quiz éducatifs et recevoir des rappels.
- **Parents** : Les parents doivent pouvoir superviser les données de leur enfant et accéder à des informations éducatives.
- **Administrateur** : Le gestionnaire du système qui a pour rôle de :
 - Gérer les comptes utilisateurs
 - Modérer les contenus éducatifs
 - Superviser l'ensemble des données du système
 - Configurer les paramètres globaux de l'application

1.3.2 Besoins Fonctionnels

- **Gestion des utilisateurs** :
 - Inscription : Les utilisateurs doivent pouvoir créer un compte.
 - Connexion : Les utilisateurs doivent pouvoir se connecter à leur compte avec un email et un mot de passe.
 - Récupération de compte : Les utilisateurs doivent pouvoir réinitialiser leur mot de passe via un lien envoyé par email en cas d'oubli.
 - Gestion du profil : Les utilisateurs doivent pouvoir mettre à jour leurs informations personnelles (ex : nom, âge, type de diabète, poids).
- **Suivi quotidien** :
 - Enregistrement des données : Les enfants doivent pouvoir enregistrer leur glycémie,
 - Visualisation des données : Les données doivent être générées sous forme de graphiques d'évolution et un tableau pour une compréhension facile.
 - Historique : Les utilisateurs doivent pouvoir consulter l'historique de leurs données sur une période donnée (ex : jour, semaine, mois) avec une alerte automatique.
- **Quiz éducatifs** :
 - Accès aux quiz : Les enfants doivent pouvoir accéder à des quiz interactifs sur la gestion du diabète.
 - Feedback immédiat : Les enfants doivent recevoir un feedback immédiat après avoir répondu aux questions (ex : explications, scores).
 - Récompenses : Les enfants doivent être motivés par des récompenses (ex :

badges, étoiles) pour leur participation aux quiz.

— **Rappels et notifications :**

- Rappels personnalisés : Les enfants doivent recevoir des rappels pour mesurer leur glycémie, prendre leurs médicaments ou faire de l'exercice.
- Personnalisation : Les rappels doivent être personnalisables en fonction des besoins de chaque enfant (ex : horaires, fréquence).

— **Gestion des régimes alimentaires :**

- Recommandations personnalisées : Le système propose des régimes adaptés en fonction des valeurs de glycémie (ex : menus pauvres en sucres en cas d'hyperglycémie).
- Automatisation : Un algorithme génère des conseils nutritionnels basés sur les dernières mesures et le type de diabète.

1.3.3 Besoins Non Fonctionnels

- **Sécurité** : Les données des utilisateurs (ex : informations personnelles, données de santé) doivent être protégées contre les accès non autorisés.
- **Performance** : Le système doit être capable de gérer un grand nombre d'utilisateurs simultanément sans perte de performance.
- **Disponibilité** : Le site web doit être accessible 24/7, avec un temps d'arrêt minimal pour la maintenance.
- **Convivialité** : Le site web doit être accessible et fonctionnel sur différents appareils (ordinateurs, tablettes, smartphones).
- **Compatibilité** : Le site web doit être accessible sur différents systèmes d'exploitation (Windows, macOs, Linux).

1.4 Méthodologies de Conception

1.4.1 Méthodologie Classique

Nous allons, tout d'abord, définir une méthode classique telle que la méthode en V.

Méthode en V Le modèle en V est un modèle d'organisation des activités, qui est dérivé du modèle en cascade. Il est marqué par un flux descendant d'activités détaillant un produit jusqu'à sa réalisation, tandis qu'un flux ascendant assemble le produit en prouvant

sa qualité afin de faire un test qui se déroule en parallèle avec les autres activités. En cas d'anomalie, ce modèle permet d'éviter le retour aux phases précédentes.

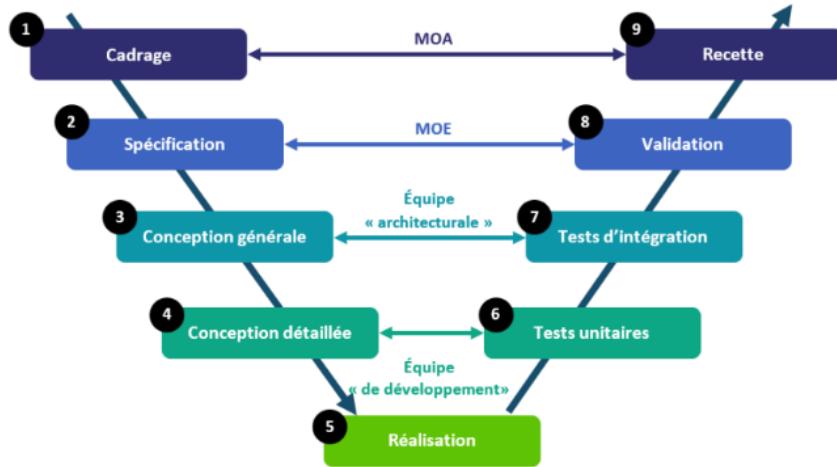


Figure 4 – Méthode en V

1.4.2 Méthodologie Agile

À ce stade, nous définissons une méthodologie de développement agile telle que le modèle Scrum.

Modèle Scrum C'est une méthode dynamique et partageable, la plus couramment utilisée. Elle repose sur des sprints et est basée sur la notion d'être toujours prêt à réorganiser le projet au fur et à mesure de sa progression.

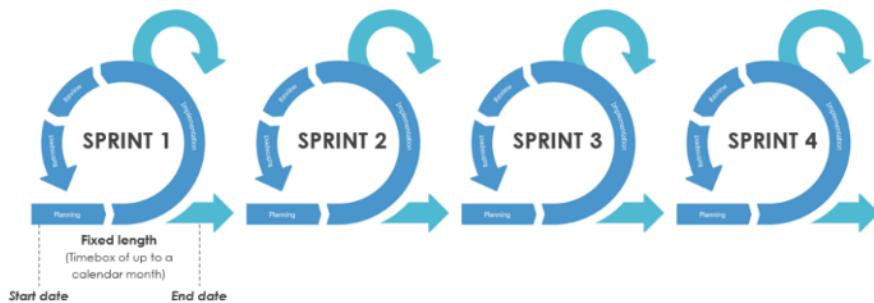


Figure 5 – Modèle Scrum

1.4.3 Processus Unifié

Nous allons définir une méthode de processus unifié telle que 2TUP.

Two Track Unified Process (2TUP) 2TUP présente un cycle de développement en Y et qui sépare et parallélise la résolution des questions fonctionnelles et techniques. Ce cycle de vie ressemble au cycle en cascade mais apporte une forme itérative interne à certaines tâches.

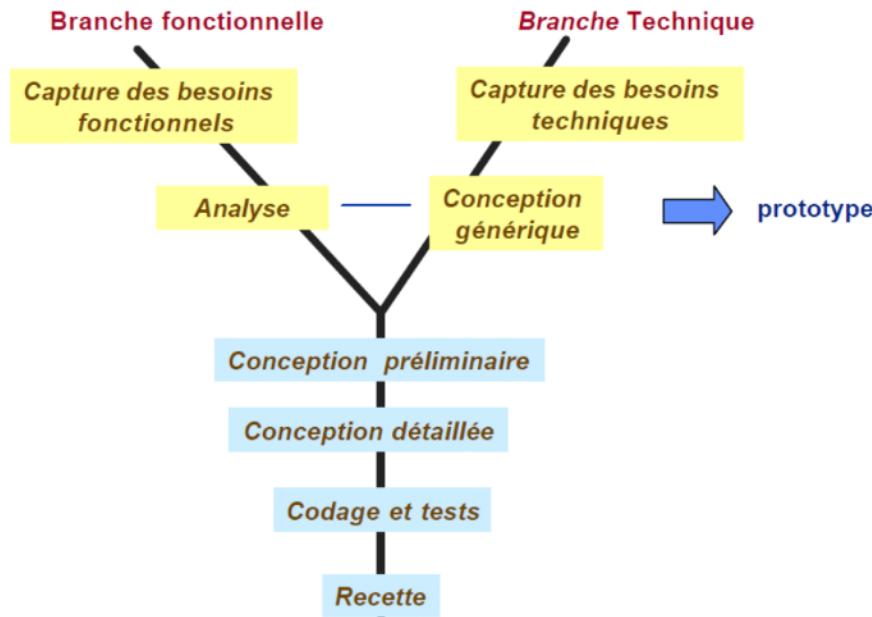


Figure 6 – Two Track Unified Process (2TUP)

1.4.4 Comparaison des Méthodes et Choix

| Méthode | Avantages | Inconvénients | Adaptabilité |
|-------------------------|---|---|--|
| Classique | Structure claire, planification détaillée | Peu flexible, difficile à adapter aux changements | Projets à exigences stables |
| Agile | Flexibilité, adaptabilité aux changements | Nécessite une communication constante | Projets à exigences évolutives |
| Processus Unifié | Gestion des risques, structure claire | Complexité accrue | Projets nécessitant une gestion rigoureuse |

Table 1 – Tableau comparatif des méthodologies de conception

Notre projet est basé sur un processus de développement bien défini qui va de la détermination des besoins fonctionnels attendus du système jusqu'à la conception et le codage final. C'est pour cela qu'on a besoin d'un cycle de développement qui dissocie les aspects techniques des aspects fonctionnels tout en commençant par une étude préliminaire. Notre choix s'est alors porté vers la méthode **2TUP** vu qu'elle est caractérisée par une approche nouvelle et originale et qu'elle respecte le cadre de notre projet.

1.4.5 Mise en pratique

La méthode 2TUP est proposée pour le développement des logiciels fournissant des services sur Internet. Le principe fondateur de 2TUP est que toute évolution imposée au logiciel peut être décomposée et traitée en parallèle selon les axes fonctionnels et technologiques. La mise en œuvre du logiciel consiste à fusionner les résultats de ces deux branches de processus. Selon le processus 2TUP que nous suivons, notre travail passe nécessairement par les étapes suivantes : Une étude préliminaire contenant les descriptions de services et les principaux cas d'utilisation constitue la première version de la spécification générale.

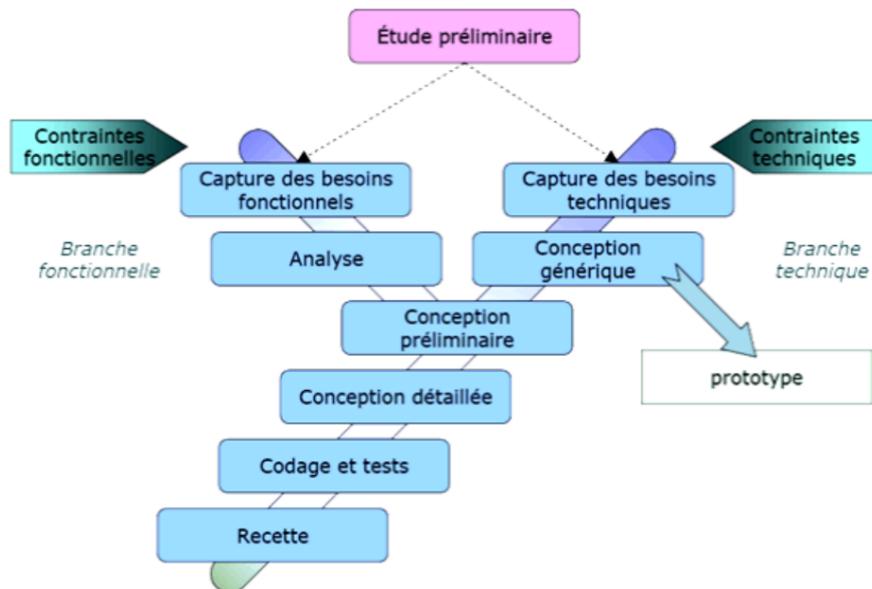


Figure 7 – Le processus 2TUP

Branche droite < technique >

- Capitalise un savoir-faire technique et/ou des contraintes techniques (Les techniques développées pour le système le sont indépendamment des fonctions à réaliser).
- Elle permet la capture des besoins non fonctionnels qui sont des contraintes que

l'application doit prendre en compte (par exemple les contraintes d'intégration, de développement et de performances).

Branche gauche « fonctionnelle »

- Elle vise la capture des besoins fonctionnels et l'analyse des spécifications fonctionnelles qui produit un modèle des besoins focalisé sur le métier des utilisateurs.
- Produit un modèle focalisé sur le métier des utilisateurs finaux et dégager toutes les fonctionnalités du système à réaliser.

Branche milieu « réalisation »

- À réunir les deux branches.
- Permettant de mener une conception applicative.
- Permettant la livraison d'une solution adaptée aux besoins des utilisateurs.

2 Étude Conceptuelle

2.1 Introduction

En faisant le choix de l'agilité et en se référant à la démarche de 2TUP, nous commençons maintenant une phase importante et indispensable dans le cycle de vie d'une application. Cette phase est la conception ayant pour but d'expliquer le déroulement de notre application et d'assurer ainsi une bonne compréhension des besoins des utilisateurs.

2.2 Architecture physique

2.2.1 Architecture à deux niveaux (2-tiers)

Le modèle d'architecture logicielle appelé architecture à deux niveaux (ou architecture 2-tiers) est principalement employé dans les applications client-serveur. Elle est constituée de deux niveaux principaux :

- **La couche client** : Le client joue le rôle de l'interface de l'utilisateur. Il a pour mission d'interagir directement avec l'utilisateur et de gérer les tâches locales, telles que la gestion des formulaires, la vérification des données ou l'envoi de requêtes au serveur.
- **La couche serveur** : La logique métier et les données sont gérées par le serveur. Il prend en compte les demandes du client, procède aux traitements requis (souvent en travaillant avec une base de données) et transmet les réponses au client. Il est possible d'intégrer des bases de données ainsi que des serveurs d'applications.

Avantages

- **Simplicité** : La conception et la mise en œuvre sont simples à comprendre et à gérer.
- **Performance** : Les traitements côté client et serveur peuvent être optimisés séparément.
- **Moins de dépendances** : Moins de couches signifie moins de points de défaillance.

Inconvénients

- **Maintenance limitée** : Toute modification dans la logique métier peut nécessiter une mise à jour simultanée des deux couches.
- **Faible évolutivité** : Le modèle ne supporte pas bien les applications à grande échelle avec un grand nombre d'utilisateurs ou des traitements complexes.

- **Sécurité** : La transmission directe entre client et serveur peut être plus vulnérable aux attaques si elle n'est pas bien sécurisée.

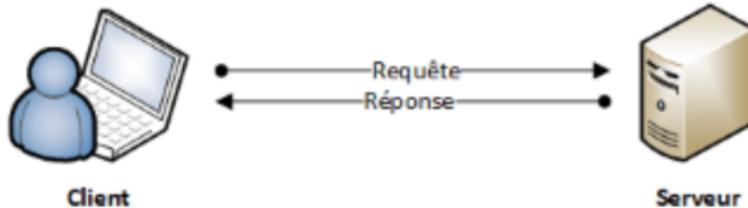


Figure 8 – Architecture 2-tiers

2.2.2 Architecture à trois niveaux (3-tiers)

La structure logicielle à trois niveaux (ou architecture 3-tiers) est couramment employée afin de concevoir des applications solides et évolutives. Elle ajoute une dimension supplémentaire à l'architecture 2-tiers, en distinguant clairement les responsabilités et en améliorant la modularité et l'évolutivité. Les trois niveaux clés comprennent :

- **Couche de présentation (Client)** : Responsable de la communication avec le client final. Offre une interface visuelle ou textuelle afin de faciliter la saisie de données et la visualisation des résultats.
- **Couche métier (Logique d'application ou middleware)** : Inclut la logique professionnelle, c'est-à-dire les règles et les traitements propres à l'application. Son rôle est de faire le lien entre la couche de présentation et la couche de données.
- **Couche de données (Base de données)** : Gère le stockage, la récupération, et la gestion des données et fournit des services pour accéder et manipuler les données nécessaires pour la logique métier.

Avantages

- **Modularité** : Les couches sont indépendantes, facilitant la maintenance et les mises à jour.
- **Évolutivité** : Améliore la répartition des charges en utilisant des serveurs spécifiques à chaque couche.
- **Réutilisabilité** : La logique métier peut être réutilisée pour différents types de clients.

Inconvénients

- **Complexité accrue** : La conception, le déploiement, et la gestion demandent plus d'efforts.
- **Latence potentielle** : Les communications entre couches peuvent introduire une surcharge.
- **Coût élevé** : Nécessite plus de ressources (serveurs, personnel qualifié).

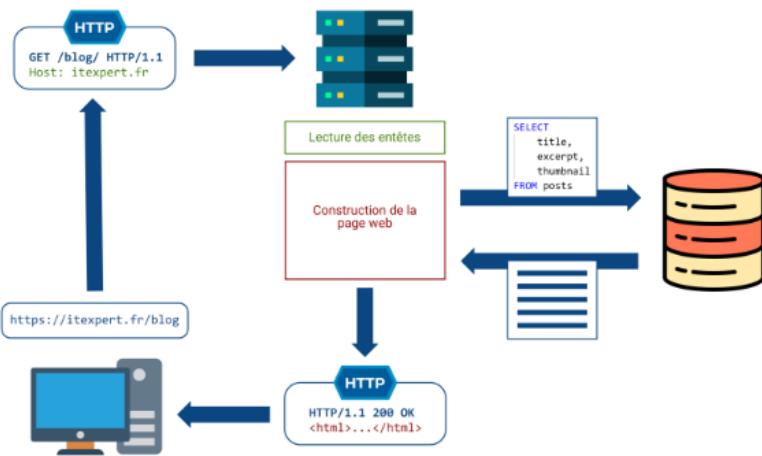


Figure 9 – Architecture 3-tiers

2.2.3 Architecture à plusieurs niveaux (N-tiers)

L'architecture N-tiers est une extension de l'architecture 3-tiers, où le nombre de couches peut augmenter pour répondre à des besoins complexes ou spécifiques. L'objectif de cette architecture est de garantir une plus grande modularité, une plus grande capacité de croissance et une plus grande souplesse, en permettant de séparer les diverses préoccupations fonctionnelles du système en couches spécifiques.

Les différentes couches

- **La couche de présentation (client)** : offre une interface utilisateur (UI) permettant aux utilisateurs finaux d'interagir avec le système.
- **La couche applicative** : regroupe la logique de l'application, traite les données et transfère les résultats entre la présentation et la couche de données.
- **La couche d'objets métier** : représente toutes les entités persistantes de l'application, telles que la Facture, le Bon de Commande, le Client, etc.

- **La couche de données (Base de données)** : gère le stockage, la récupération et la gestion des données, et fournit des services pour accéder et manipuler les données nécessaires pour la logique métier.

Avantages

- **Modularité** : Chaque étape est autonome et peut être élaborée, mise à jour ou mise en place sans interférer avec les autres.
- **Scalabilité** : Permet de faire évoluer chaque couche de manière autonome. Par exemple, il est possible d'augmenter le nombre de serveurs dans la couche de logique métier ou de données en fonction de la charge.
- **Maintenance facilitée** : Si une modification est effectuée, la maintenance devient plus facile, car chaque tier peut être mis à jour de manière autonome.
- **Sécurité renforcée** : La division des préoccupations entre les tiers permet de séparer les éléments essentiels du système (comme la couche de données).

Inconvénients

- **Complexité accrue** : La conception, le déploiement et la gestion demandent plus d'efforts.
- **Latence potentielle** : Les communications entre couches peuvent introduire une surcharge.
- **Coût élevé** : Nécessite plus de ressources (serveurs, personnel qualifié).

Les couches de l'arch. n-tiers

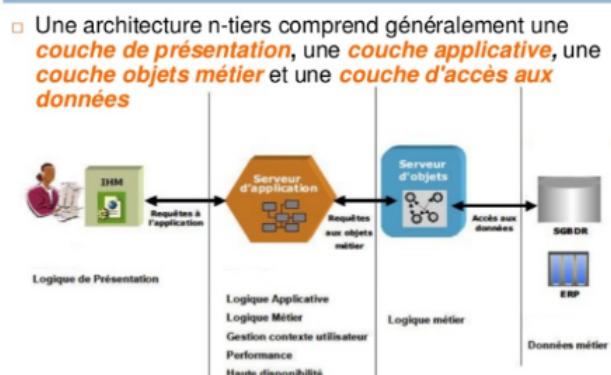


Figure 10 – Architecture à plusieurs niveaux (N-tiers)

2.2.4 Choix de l'architecture physique

Pour mon projet, l'architecture 3-tiers est le choix le plus adapté, car elle offre un bon équilibre entre simplicité et évolutivité. Comparée à une architecture 2-tiers, qui lie directement l'interface utilisateur à la base de données, l'architecture 3-tiers ajoute une couche applicative (backend), ce qui améliore la modularité, la sécurité et la gestion de la logique métier. D'un autre côté, bien que l'architecture N-tiers soit plus avancée et adaptée aux grandes applications complexes nécessitant plusieurs services distincts, elle ajoute une complexité inutile pour mon projet. Avec React pour le front-end, Node.js pour la logique métier et MongoDB pour les données, l'architecture 3-tiers me permet d'avoir une séparation claire des responsabilités, une meilleure scalabilité et une facilité de maintenance, tout en restant efficace et relativement simple à déployer.

2.3 Architecture logique

L'architecture logique représente la manière dont les composants logiciels sont organisés pour fonctionner ensemble en répondant aux besoins fonctionnels tout en assurant une structure claire et modulaire. Trois modèles principaux d'architectures logicielles sont fréquemment utilisés :

2.3.1 Architecture MVC (Model-View-Controller)

MVC signifie "Modèle, Vue, Contrôleur". Il s'agit d'un modèle de conception qui concerne l'organisation du code. La structure du code est divisée en trois parties :

- **Modèle** : Il comprend le code responsable de la logique métier et des données. Ces données proviennent généralement d'une base de données ou d'un service externe comme une API.
- **Vue** : Elle correspond à ce qui est affiché sur l'écran. La Vue affiche les données reçues du contrôleur comme résultat.
- **Contrôleur** : Il assure la communication avec l'utilisateur et orchestre l'application en déterminant quelles actions doivent être effectuées en réponse aux interactions de l'utilisateur.

Avantages

- **Clarté de la structure** : La séparation claire des responsabilités entre le modèle, la vue et le contrôleur rend l'architecture facile à comprendre et à maintenir.
- **Amélioration de la collaboration dans les équipes** : Les développeurs frontend

se concentrent sur la Vue, tandis que les développeurs backend se concentrent sur le Modèle et le Contrôleur.

- **Modularité** : Chaque composant est indépendant, ce qui facilite la maintenance et les mises à jour sans affecter les autres parties du système.

Inconvénients

- **Dépendance accrue entre les composants** : Une implémentation incorrecte peut entraîner une forte dépendance entre le Modèle, la Vue et le Contrôleur, ce qui compromet la modularité.
- **Tests plus complexes** : Bien que la division en couches simplifie les tests unitaires, tester les interactions entre les composants (Modèle, Vue et Contrôleur) peut être complexe.
- **Complexité pour de petites applications** : Pour des applications simples, l'architecture MVC peut être trop complexe par rapport à des solutions plus légères.

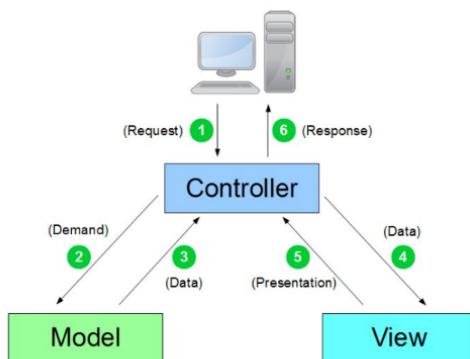


Figure 11 – Architecture MVC

2.3.2 Architecture MVP (Model-View-Presenter)

L'architecture MVP est une évolution de l'architecture MVC dans laquelle la Vue est complètement découpée de la logique métier. La logique de présentation est gérée par un Présentateur. Ce dernier interagit avec le Modèle pour obtenir les données et met à jour la Vue.

- **Modèle** : Gère les données et la logique métier.
- **Vue** : Affiche les informations à l'utilisateur et envoie les événements utilisateurs au Présentateur.

- **Présentateur** : Intermédiaire entre la Vue et le Modèle. Il coordonne les échanges et gère la logique de présentation.

Avantages

- **Réutilisabilité de la logique métier** : L'indépendance du Présentateur par rapport à l'interface utilisateur facilite son adaptation à différentes plateformes.
- **Adapté aux applications complexes** : MVP est utile pour des interfaces riches ou des applications nécessitant une logique de présentation avancée.
- **Meilleure séparation des responsabilités** : La Vue devient entièrement passive, ce qui offre une plus grande modularité.

Inconvénients

- **Code plus volumineux** : Le Présentateur peut devenir très complexe, surtout pour des applications avec de nombreuses interactions.
- **Moins adapté aux frameworks modernes** : Les frameworks comme React ou Angular utilisent des paradigmes différents qui rendent MVP moins pertinent.

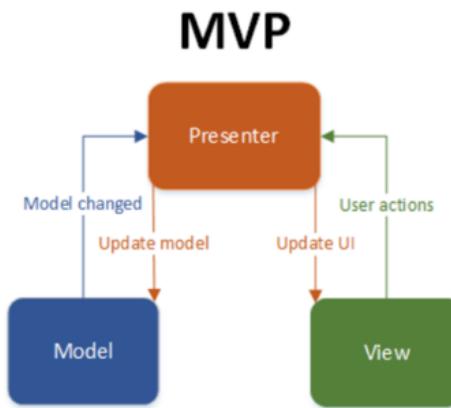


Figure 12 – Architecture MVP

2.3.3 Architecture MVVM (Model-View-ViewModel)

L'architecture MVVM est couramment utilisée pour structurer les applications, notamment pour le développement d'interfaces utilisateur. Elle sépare les préoccupations pour rendre le code plus modulaire et facile à maintenir. Voici les éléments essentiels :

- **Modèle** : Représente les données de l'application et la logique métier. Il gère les données à l'aide de bases de données ou de services externes.

- **Vue** : La représentation graphique de l’interface utilisateur. Elle affiche les données, sans contenir de logique métier.
- **ViewModel** : Un médiateur entre le Modèle et la Vue, qui fournit des données formatées pour la Vue et gère les interactions utilisateur avec le Modèle.

Avantages

- **Séparation des préoccupations** : Chaque composant (Modèle, Vue, ViewModel) a un rôle bien défini, ce qui facilite la gestion de l’application.
- **Maintenance simplifiée** : Le code devient plus modulaire et plus facile à comprendre.
- **Réutilisabilité** : Le ViewModel peut être utilisé avec différentes vues.

Inconvénients

- **Configuration de binding** : Si mal implémentée, la configuration de la liaison des données peut entraîner des erreurs.
- **Surcharge de développement** : Si les besoins du projet sont simples, MVVM peut être surdimensionné.
- **Complexité initiale** : Le modèle peut être difficile à maîtriser pour les débutants.

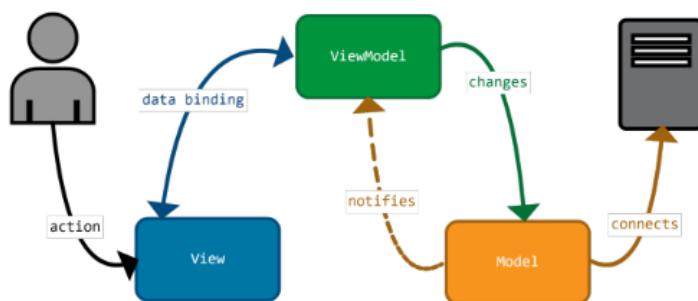


Figure 13 – Architecture MVVM

2.3.4 Choix de l’architecture de l’application

Pour un site web classique, l’architecture MVC (Model-View-Controller) est généralement la plus adaptée, car elle sépare clairement la logique métier, l’interface utilisateur et le contrôle des actions, facilitant ainsi la gestion du code. Si le site présente des interactions plus complexes, l’architecture MVP (Model-View-Presenter) offre une plus grande modularité. L’architecture MVVM (Model-View-ViewModel), quant à elle, est plus adaptée

aux frameworks modernes comme React ou Angular, mais pour un site simple, MVC reste le choix le plus pratique et facile à maintenir.

2.4 Langage de Conception

Pour élaborer cette application, nous devons établir une conception modeste pour atteindre le but de notre projet. Pour cela, nous devons choisir un langage de conception adapté à nos besoins.

2.4.1 Langage UML

« UML » (en anglais Unified Modeling Language ou langage de modélisation unifié) est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.

2.5 Diagrammes UML

Vues dans la modélisation UML

Une vue est constituée d'un ou plusieurs diagrammes. On distingue deux types de vues :

Les vues statiques

Les vues statiques permettent de représenter le système physiquement. Elles incluent :

- **Diagrammes de classes** : Représentent des collections d'éléments de modélisation statiques (classes, paquetages, etc.), qui montrent la structure d'un modèle.
- **Diagrammes d'objets** : Ces diagrammes montrent des objets (instances de classes dans un état particulier) et des liens (relations sémantiques) entre objets.
- **Diagrammes de cas d'utilisation** : Identifient les utilisateurs du système (acteurs) et leurs interactions avec le système.
- **Diagrammes de composants** : Permettent de décrire l'architecture physique statique d'une application en termes de modules : fichiers sources, librairies exécutables, etc.
- **Diagrammes de déploiement** : Montrent la disposition physique du matériel qui compose le système et la répartition des composants sur ce matériel.

Les vues dynamiques

Les vues dynamiques montrent le fonctionnement du système. Elles incluent :

- **Diagrammes de collaboration** : Montrent des interactions entre objets (instances de classes et acteurs).
- **Diagrammes de séquence** : Permettent de représenter des collaborations entre objets selon un point de vue temporel, en mettant l'accent sur la chronologie (envois de messages).
- **Diagrammes d'états-transitions** : Permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.
- **Diagrammes d'activités** : (Une variante des diagrammes d'états-transitions) Servent à représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.

2.6 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation identifie les fonctionnalités fournies par le système (cas d'utilisation), les utilisateurs qui interagissent avec le système (acteurs) et décrit les interactions entre ces derniers.

Composition du diagramme de cas d'utilisation

Le diagramme de cas d'utilisation se compose de :

- **Un cas d'utilisation** : Représente une fonctionnalité fournie par le système pour répondre aux besoins des acteurs. Il est décrit sous la forme d'un verbe à l'infinitif suivi par un complément, par exemple (gérer caisse, supprimer produit), et est représenté par une ellipse contenant son nom.
- **Un acteur** : Représente un rôle tenu par une entité (utilisateur, objet, autre application, etc.) externe au système qui interagit avec le système. Une même entité peut jouer plusieurs rôles.
- **Une association** : Elle est utilisée pour lier des acteurs avec des cas d'utilisation. L'association indique qu'un acteur participe au cas d'utilisation. Elle est représentée par une ligne reliant l'acteur et le cas d'utilisation.
- **Relation entre cas d'utilisation** : Le diagramme de cas d'utilisation permet d'établir des relations entre ces cas d'utilisation. Il existe 3 types de relations :

- **Relation d'inclusion** : Symbolisée par le mot-clé <<include>> (le cas A <<include>> B), le cas d'utilisation de base inclut obligatoirement l'autre.
- **Relation d'extension** : Symbolisée par le mot-clé <<extend>> (le cas A <<extend>> B), le cas d'utilisation de base est une extension optionnelle de l'autre à un certain point de son exécution.
- **Relation de généralisation/spécialisation** : Les cas d'utilisation descendants héritent de la description de leur parente commune. Ils peuvent également avoir des descriptions spécifiques supplémentaires.

Spécification des cas d'utilisation

Chaque cas d'utilisation doit être précisé par une description textuelle qui peut être structurée en plusieurs sections :

- **Titre** : Nom du cas d'utilisation.
- **Résumé** : Description brève du cas d'utilisation.
- **Les acteurs** : Noms des acteurs principaux qui déclenchent le cas.
- **Pré-condition** : Décrit dans quel état doit être le système avant que ce cas d'utilisation puisse être déclenché.
- **Post-condition** : Décrit l'état du système après le déroulement de la séquence nominale.
- **Scénario nominal** : C'est un chemin d'exécution le plus général du cas d'utilisation.
- **Enchaînement alternatif** : C'est une succession d'une ou de plusieurs étapes qui peuvent remplacer une ou plusieurs étapes du scénario nominal. Un scénario alternatif propose une autre possibilité de déroulement du cas d'utilisation que celle prévue dans le scénario nominal.

2.6.1 Diagramme de cas d'utilisation globale

La figure suivante illustre le diagramme de cas d'utilisation globale, qui nous permet d'obtenir une vision globale du comportement fonctionnel de l'application.

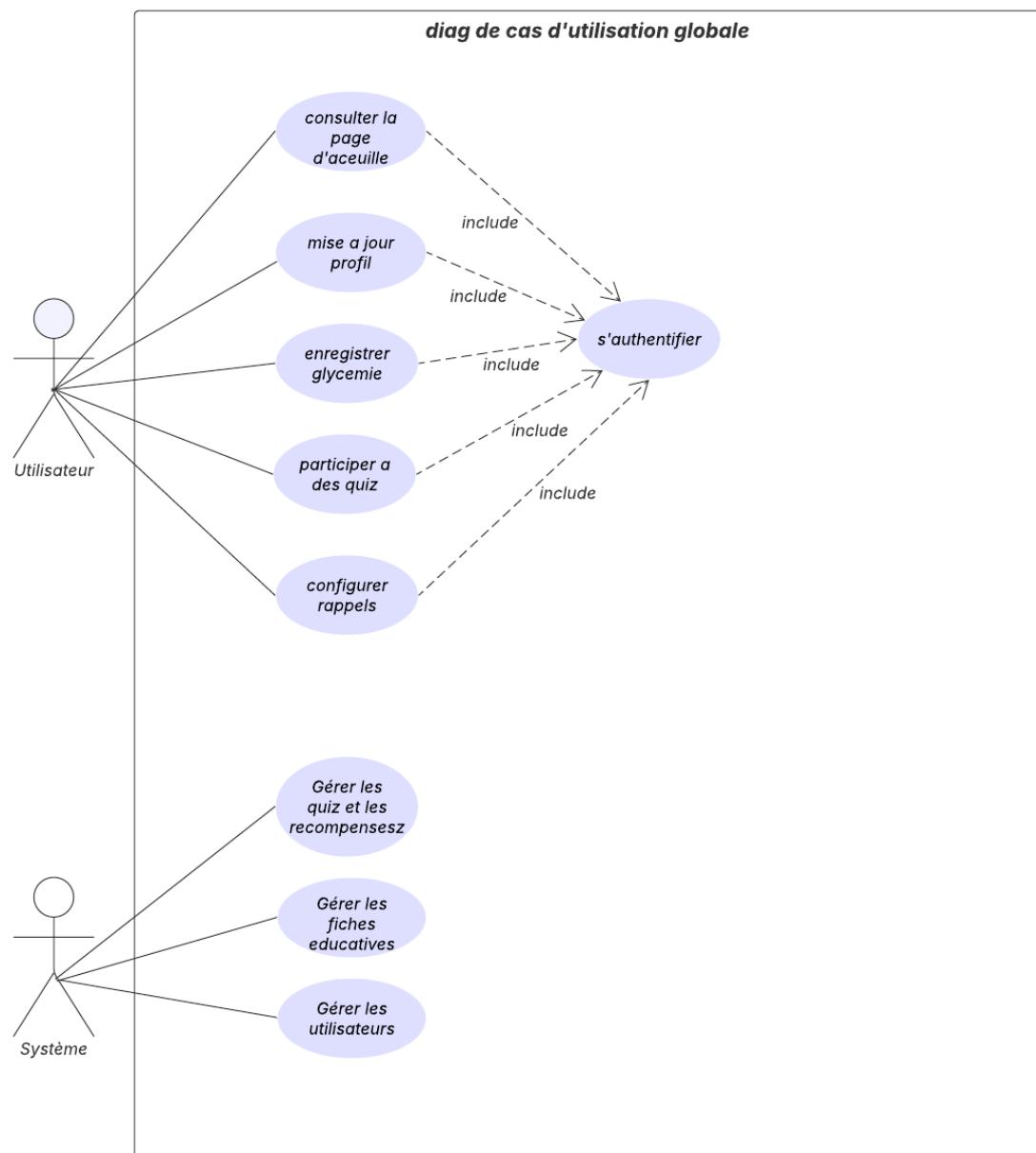


Figure 14 – Diagramme de cas d'utilisation globale

2.6.2 Raffinement des cas d'utilisation

Dans le but de mieux comprendre les cas d'utilisation présentés dans le diagramme de cas d'utilisation général, nous détaillons par le raffinement et les tableaux descriptifs pour décortiquer et retrouver les cas d'utilisation inclus.

2.6.3 Raffinement des cas d'utilisation «S'authentifier»

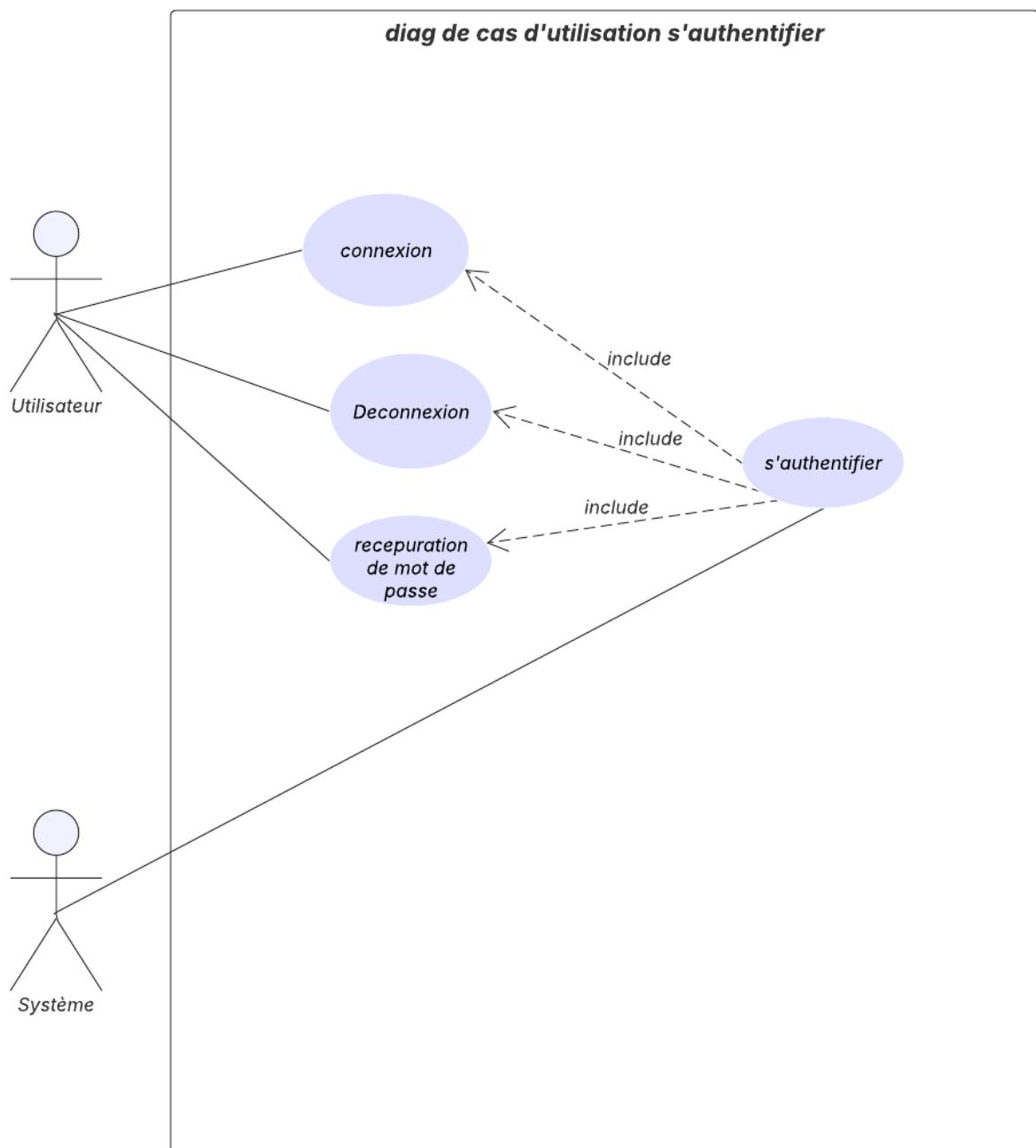


Figure 15 – Diagramme des cas d'utilisation créer un compte

| Élément | Description |
|---------------------|--|
| Acteurs | Utilisateur, Systeme |
| Résumé | L'utilisateur se connecte à son compte en fournissant ses identifiants. |
| Pré-condition | L'utilisateur possède un compte valide. |
| Post-condition | L'utilisateur est authentifié et a accès à son compte. |
| Scénario nominal | <ol style="list-style-type: none"> 1. L'utilisateur ouvre l'application et clique sur "Connexion" 2. Il saisit son Email et son mot de passe 3. Le système vérifie les identifiants 4. Si valides, l'utilisateur est redirigé vers Sugarhero 5. Un token de session est généré et stocké. |
| Scénario alternatif | <ul style="list-style-type: none"> — Les informations d'identification sont incorrectes : le système affiche un message d'erreur et invite l'utilisateur à réessayer. — Le compte n'existe pas : le système propose de créer un nouveau compte. |

Table 2 – Tableau descriptif de cas d'utilisation : « S'authentifier »

2.6.4 Raffinement des cas d'utilisation «créer un compte »

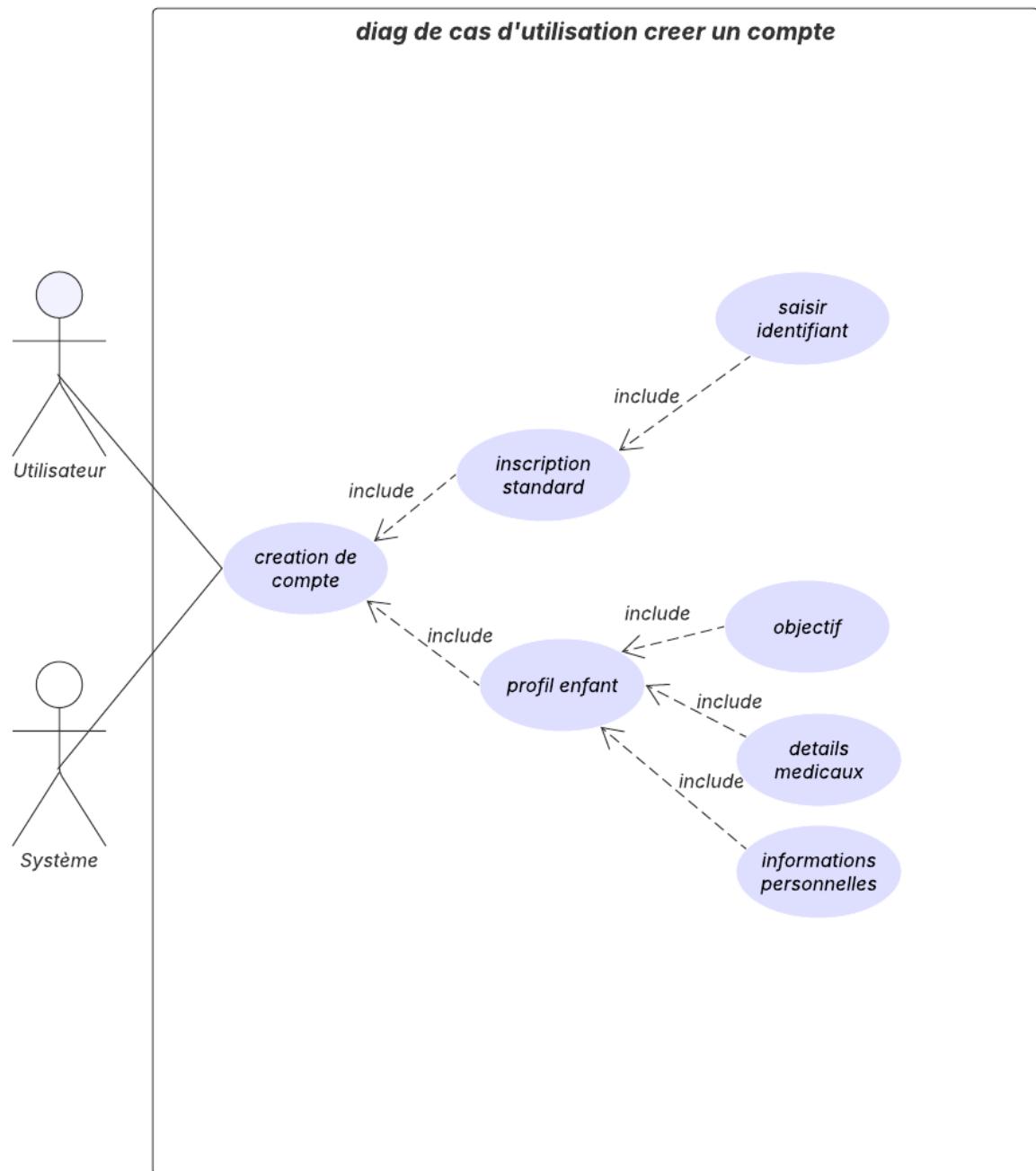


Figure 16 – Diagramme des cas d'utilisation S'authentifier

| Élément | Description |
|---------------------|---|
| Acteurs | utilisateur |
| Résumé | L'enfant crée un compte en renseignant son nom, son rôle et son mot de passe. |
| Pré-condition | <ul style="list-style-type: none"> — L'utilisateur ne doit pas déjà avoir un compte — Une connexion Internet est requise. |
| Post-condition | <ul style="list-style-type: none"> — Le compte est créé avec les informations nécessaires. |
| Scénario nominal | <ol style="list-style-type: none"> 1. Saisit email et mot de passe 2. Vérifie et enregistre les informations 3. Demande configuration initiale 4. Affiche les questions de configuration 5. Répond aux questions 6. Enregistre les réponses 7. Confirmation de la création du compte |
| Scénario alternatif | <ul style="list-style-type: none"> — L'utilisateur entre un nom déjà utilisé : — Le système affiche un message d'erreur et propose de choisir un autre nom. — Le mot de passe ne respecte pas les critères de sécurité : — Le système affiche un message indiquant les règles de sécurité à respecter. — Erreur de connexion Internet : — Le système informe l'utilisateur et propose de réessayer plus tard. |

Table 3 – Tableau descriptif de cas d'utilisation : « Créer un compte »

2.6.5 Raffinement du cas d'utilisation : Suivre la glycémie

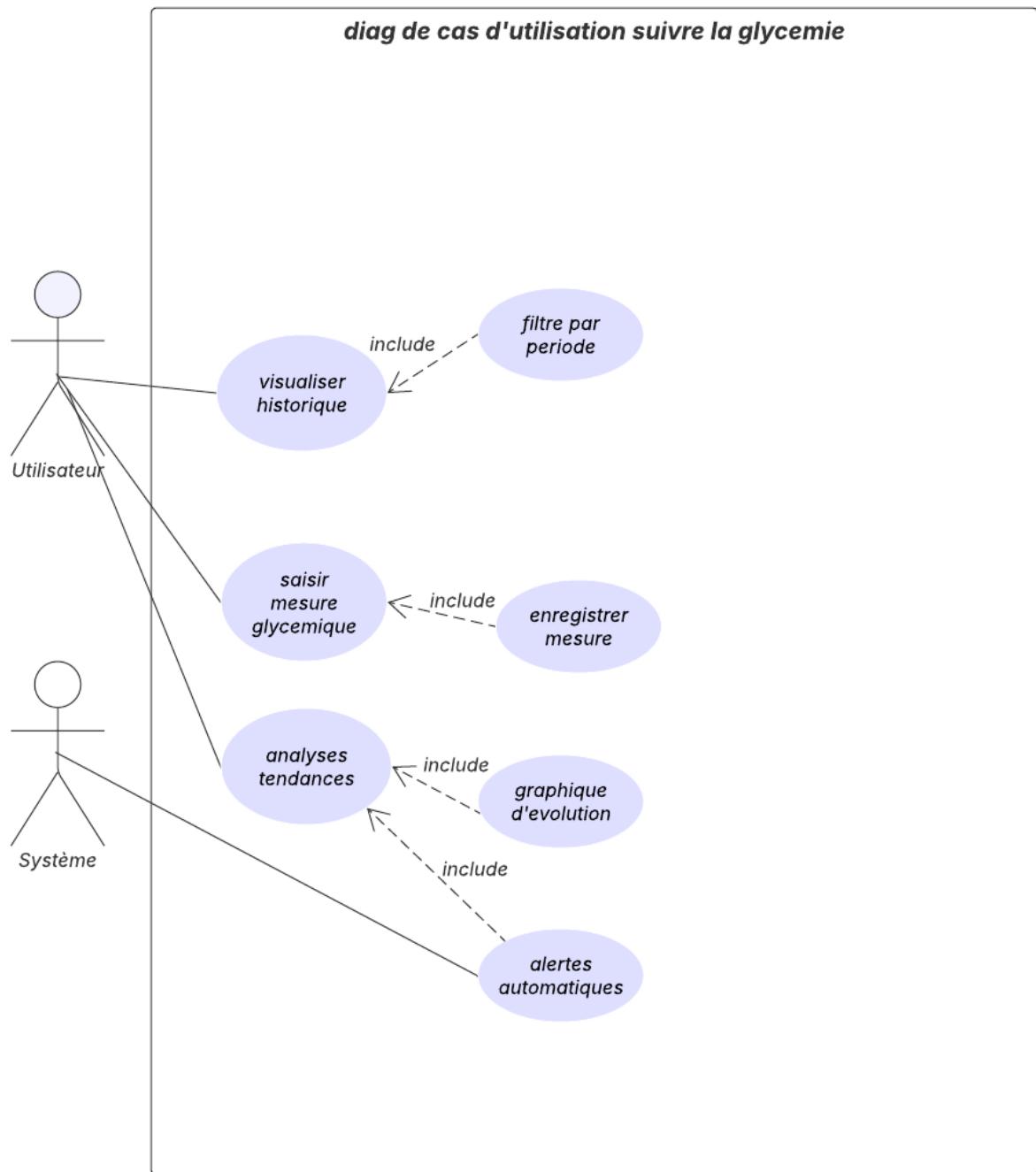


Figure 17 – Diagramme des cas d'utilisation Suivre la glycémie

| Élément | Description |
|---------------------|--|
| Acteurs | utilisateur, Système d'authentification |
| Résumé | L'utilisateur enregistre les valeurs de glycémie pour suivre l'évolution. |
| Pré-condition | <ul style="list-style-type: none"> — L'utilisateur est authentifié. |
| Post-condition | <ul style="list-style-type: none"> — Les données de glycémie sont enregistrées et accessibles dans l'historique. |
| Scénario nominal | <ol style="list-style-type: none"> 1. Saisit une nouvelle mesure (date, heure, valeur) 2. Enregistre la mesure dans la base de données 3. Génère et affiche un graphique d'évolution 4. Affiche l'historique des mesures et une alerte automatique |
| Scénario alternatif | <ul style="list-style-type: none"> — Valeur anormale détectée : <ul style="list-style-type: none"> — Le système affiche une alerte et propose des recommandations. — Problème de connexion <ul style="list-style-type: none"> — Le système enregistre temporairement les données et les synchronise plus tard. — Erreur de connexion Internet : |

Table 4 – Tableau descriptif de cas d'utilisation : Suivre la glycémie

2.6.6 Raffinement du cas d'utilisation : "Gérer les rappels de traitement"

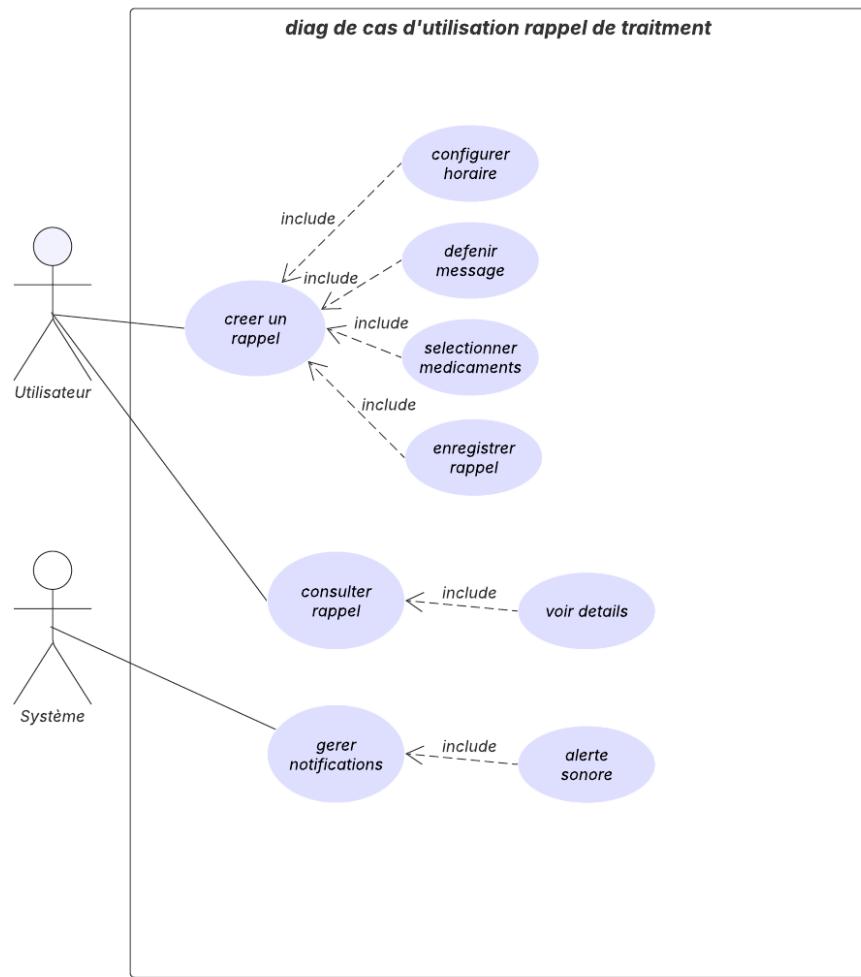


Figure 18 – Diagramme de cas d'utilisation : "Gérer les rappels de traitement"

| Élément | Description |
|---------------------|---|
| Acteurs | utilisateur |
| Résumé | L'utilisateur configure des rappels pour les prises d'insuline, les repas ou d'autres traitements médicaux. |
| Pré-condition | — L'utilisateur est authentifié. |
| Post-condition | — Les rappels sont enregistrés et activés. |
| Scénario nominal | <ol style="list-style-type: none"> 1. L'utilisateur configure un rappel (heure, type, fréquence) 2. Le système enregistre le rappel 3. Le système envoie une notification à l'heure prévue 4. L'utilisateur consulte, modifie ou supprime un rappel |
| Scénario alternatif | <ul style="list-style-type: none"> — Le rappel est manqué : — Le système envoie une deuxième notification après un certain temps. — Problème de configuration : — Le système informe l'utilisateur d'une erreur et propose une correction. |

Table 5 – Tableau descriptif : "Gérer les rappels de traitement"

2.6.7 Raffinement du cas d'utilisation : "Jouer à un quiz éducatif"

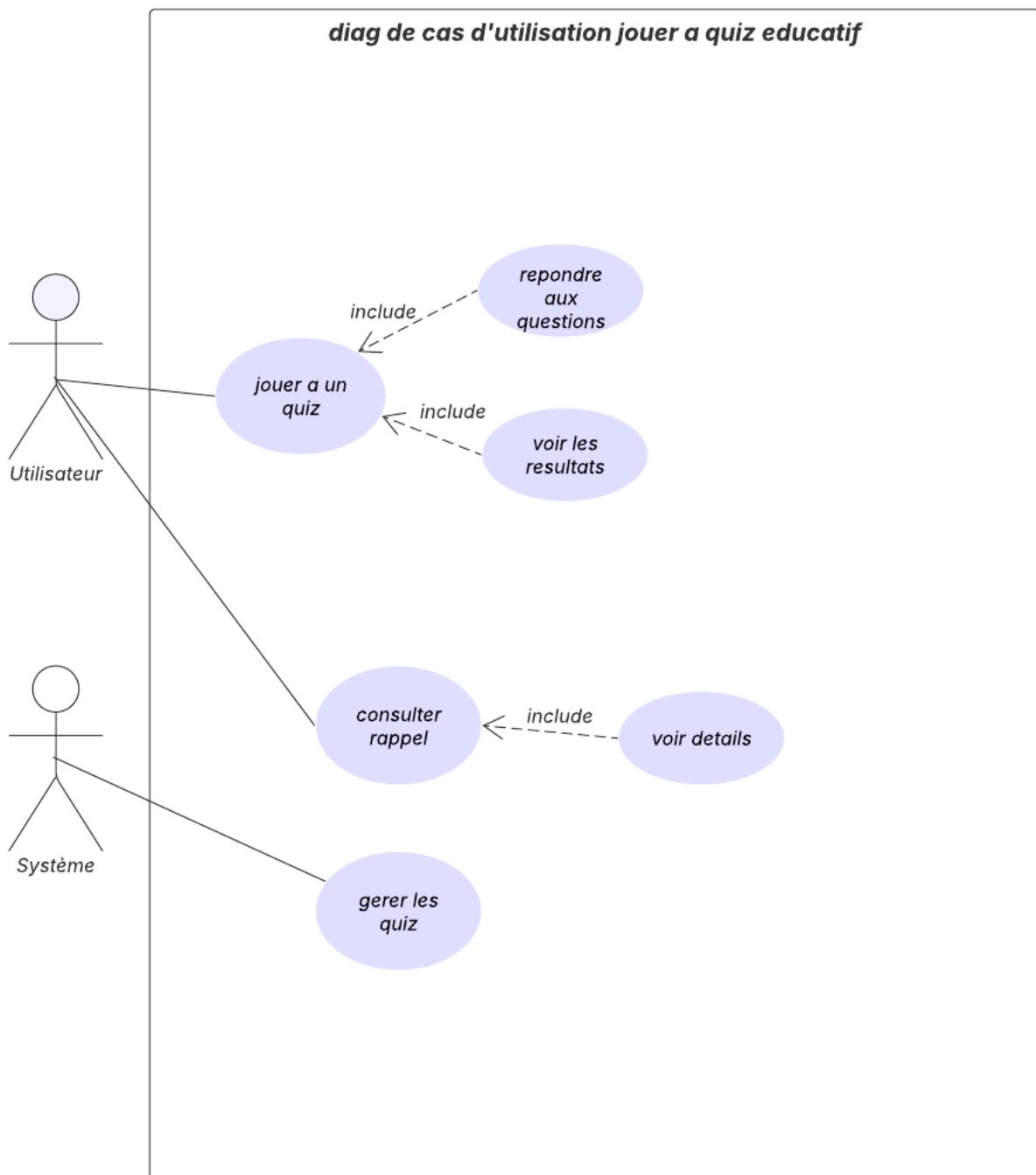


Figure 19 – Diagramme de cas d'utilisation : "Jouer à un quiz éducatif"

| Élément | Description |
|---------------------|--|
| Acteurs | Enfant, Système d'authentification |
| Résumé | L'enfant joue à un quiz pour tester ses connaissances sur la gestion du diabète. |
| Pré-condition | — L'enfant est authentifié. |
| Post-condition | — Les résultats du quiz sont enregistrés et peuvent être consultés. |
| Scénario nominal | <ol style="list-style-type: none"> 1. L'enfant sélectionne un quiz. 2. Il répond aux différentes questions. 3. Il valide ses réponses. 4. Le système affiche le score et les récompenses |
| Scénario alternatif | <ul style="list-style-type: none"> — L'enfant quitte en cours de jeu — Le système enregistre la progression et propose de reprendre plus tard. |

Table 6 – Tableau descriptif du cas d'utilisation : "Jouer à un quiz éducatif"

2.7 Diagrammes de séquences

Le diagramme de séquence permet de modéliser la dynamique de l'application et d'identifier les objets, les messages entre ces objets et leurs ordres. En fait, focalise sur l'enchaînement du message. Pour bien comprendre la dynamique de notre système nous proposons les diagrammes de séquence illustrés dans les figures suivantes.

2.7.1 Diagramme de séquence «Inscription»

Si l'utilisateur ne possède pas de compte, il peut en créer un. Pour cela, il doit remplir le formulaire d'inscription en renseignant son email, son mot de passe, son rôle ainsi que

les informations requises. Si tous les champs sont correctement remplis, un message de confirmation s'affiche et le compte est créé avec succès. Dans le cas contraire, l'application affiche un message d'erreur demandant à l'utilisateur de compléter ou corriger les informations manquantes.

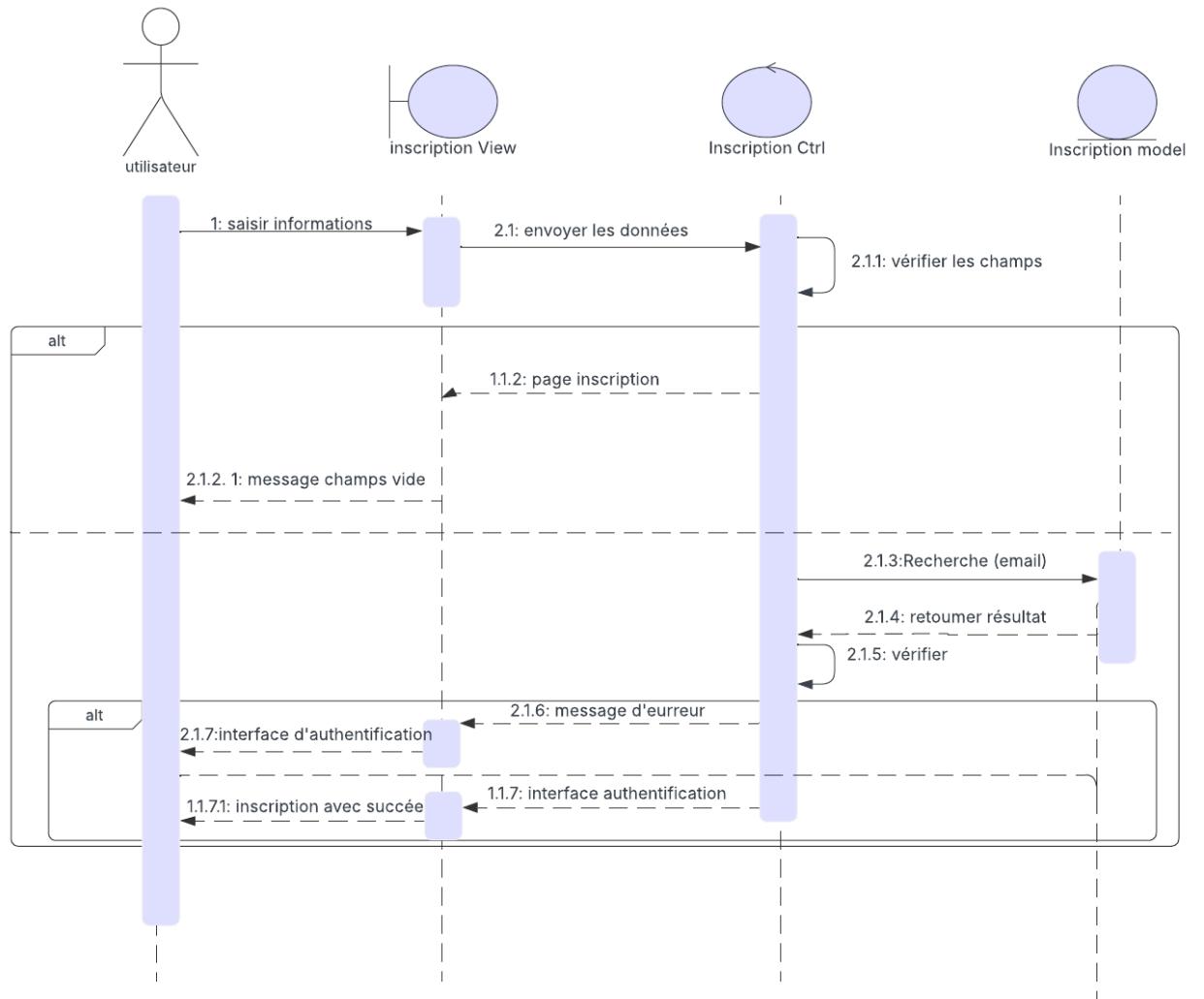


Figure 20 – Diagramme de séquence «Inscription»

2.7.2 Diagramme de séquence «mot de passe oublier»

Lorsqu'un utilisateur oublie son mot de passe, il saisit son email sur l'écran de réinitialisation. Le système envoie alors un lien sécurisé à cette adresse email. L'utilisateur clique sur ce lien pour accéder à une page lui permettant de saisir un nouveau mot de passe. Après validation, le système met à jour les informations dans la base de données et confirme la réinitialisation. Si l'email n'existe pas ou si le lien a expiré, un message d'erreur explicite

guide l'utilisateur pour recommencer la procédure.

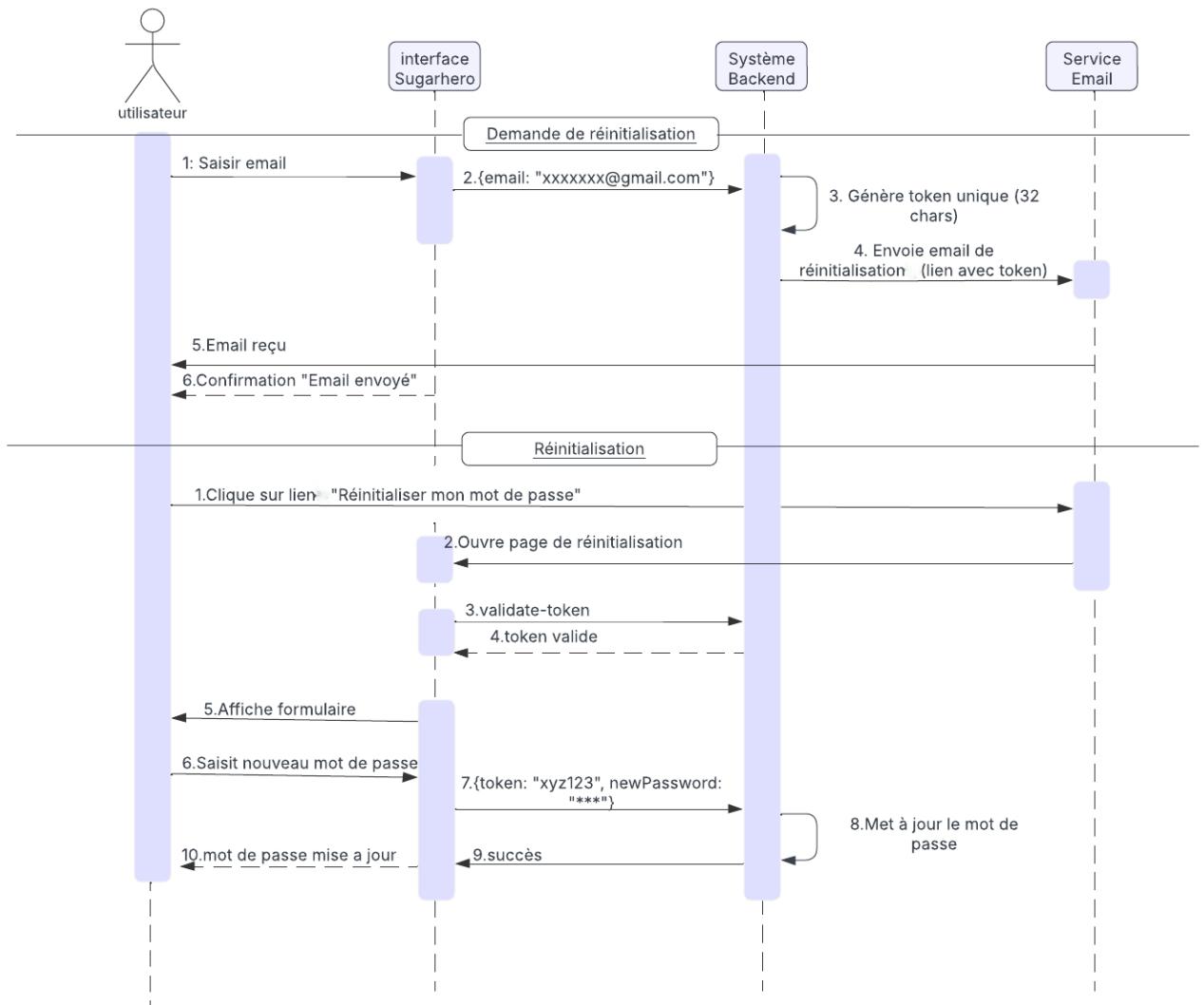


Figure 21 – Diagramme de séquence mot de passe oublier

2.8 Diagrammes de classe

Un diagramme de classes est une modélisation statique du système en termes des classes constituant le système et les associations entre elles. De même qu'une classe décrit un ensemble d'objets, une association décrit un ensemble de liens; les objets sont des instances de classes et les liens sont des instances de relations. C'est une modélisation objet qui consiste à créer une représentation abstraite des objets du système ayant une existence matérielle ou virtuelle qui vont interagir pour réaliser les cas d'utilisation.

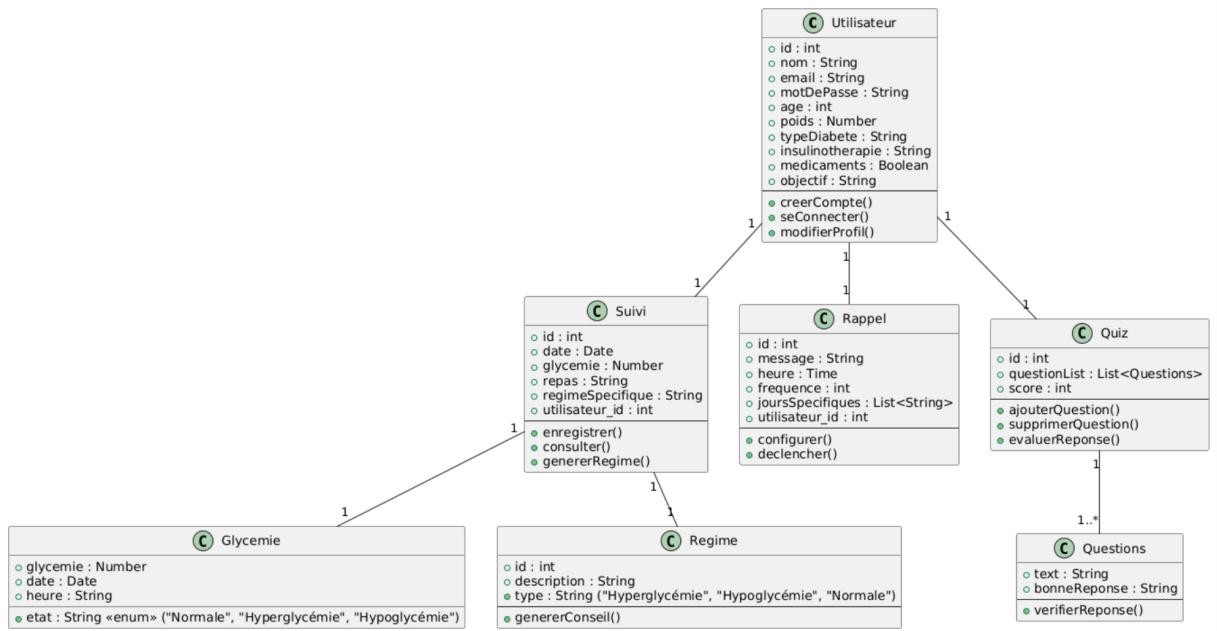


Figure 22 – Diagramme de classe

3 Réalisation

Dans ce chapitre, nous présentons la partie réalisation et mise en œuvre des différents composants décrits au niveau du chapitre précédent. Dans un premier temps, nous présenterons l'environnement matériel et logiciel. Dans un second temps nous décrirons le travail réalisé en détaillant quelques fonctionnalités réalisées.

3.1 Environnement du travail

Dans cette section nous présentons l'environnement du développement du travail aussi bien matériel que logiciel.

3.1.1 Environnement matériel

Pour développer l'application, nous avons utilisé comme environnement matériel un ordinateur portable

| | |
|-------------------------------|----------------|
| Marque | MSI |
| Processeur | Intel Core i7 |
| Mémoire (RAM) | 16 GO |
| Disque Dur | 338.96 GO NTFS |
| Système d'exploitation | Windows 11 |

Table 7 – Environnement matériel

3.1.2 Environnement logiciel

Cette partie est consacrée à la présentation des différentes technologies et outils logiciels utilisés pour le développement de notre application web.

1. Frontend

- **React** : Bibliothèque JavaScript pour les interfaces utilisateur. Choisie pour sa flexibilité, ses composants réutilisables et sa gestion d'état efficace.

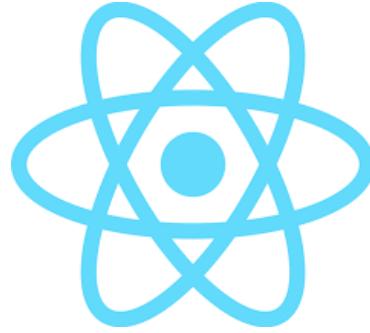


Figure 23 – Logo React

2. Backend

- **Node.js** : Environnement d'exécution JavaScript côté serveur (moteur V8). Permet un développement full-JS et gère efficacement les requêtes asynchrones.



Figure 24 – Logo Node.js

- **Express.js** : Framework minimaliste pour Node.js. Simplifie la création d'API RESTful et la gestion des routes.



Figure 25 – Logo Express.js

3. Base de données

- **MongoDB** : Base de données NoSQL orientée documents. Adaptée pour stocker les données utilisateurs et les mesures de glycémie sous forme de documents JSON.



Figure 26 – Logo MongoDB

4. Outils de développement

- **Visual Studio Code** : Éditeur avec support étendu pour JavaScript/TypeScript, intégration Git et extensions utiles.



Figure 27 – Logo VS Code

- **Postman** : Plateforme collaborative pour tester les API.



Figure 28 – Logo Postman

- **Lucidchart** : Outil de création de diagrammes UML.

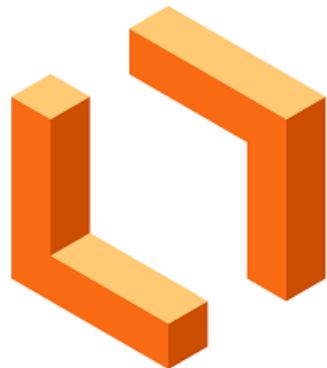


Figure 29 – Logo Lucidchart

5. Rédaction technique

- **LaTeX** : Système de composition de documents professionnels, utilisé via Overleaf pour la collaboration.



Figure 30 – Logo LaTeX

3.2 Présentation des interfaces de l'application

L'une des premières priorités dans la création d'une application ergonomie est de procurer à l'utilisateur une interface simple d'utilisation avec une navigation facile, performante et lisible. Dans cette partie nous illustrons quelques interfaces que nous avons réussi à implémenter.

3.2.1 Interface authentification

L'accès à l'application web est protégé par un système d'authentification. L'administrateur et le patient doivent saisir leurs email et mot de passe de passe à fin d'y accéder.

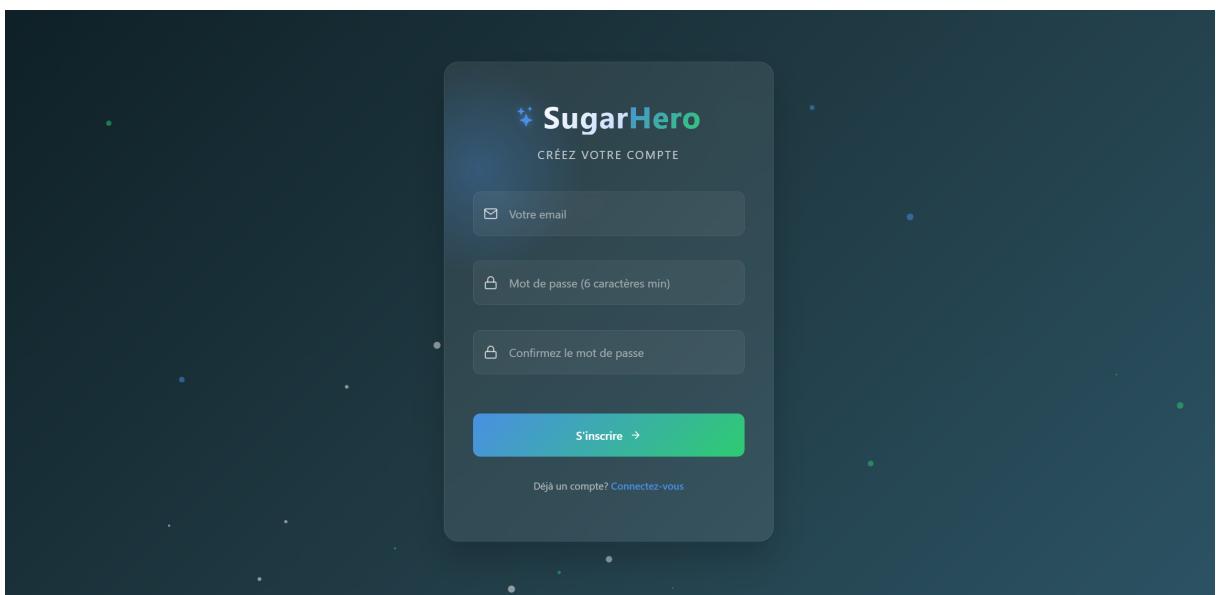


Figure 31 – interface création d'un compte

Formulaire simple avec champs essentiels (email, mot de passe). Validation en temps réel avec messages d'erreur explicites.

The screenshot shows a dark-themed web form titled "Complète tes informations pour commencer l'aventure". It contains several input fields with placeholder text and icons:

- Nom :** Ton prénom
- Âge (1 à 12 ans) :** Ton âge
- Poids (kg) :** Ton poids
- Insulinothérapie :** Choisis ton traitement
- Prends-tu des médicaments ?** (radio buttons: Oui, Non)
- Objectif :** Ton objectif (ex: mieux gérer mon diabète)

A green button at the bottom right says "Créer mon Profil →".

Figure 32 – interface saisie informations médicales

Formulaire structuré pour enregistrer les données de santé. Suggestions automatiques basées sur l'historique utilisateur.

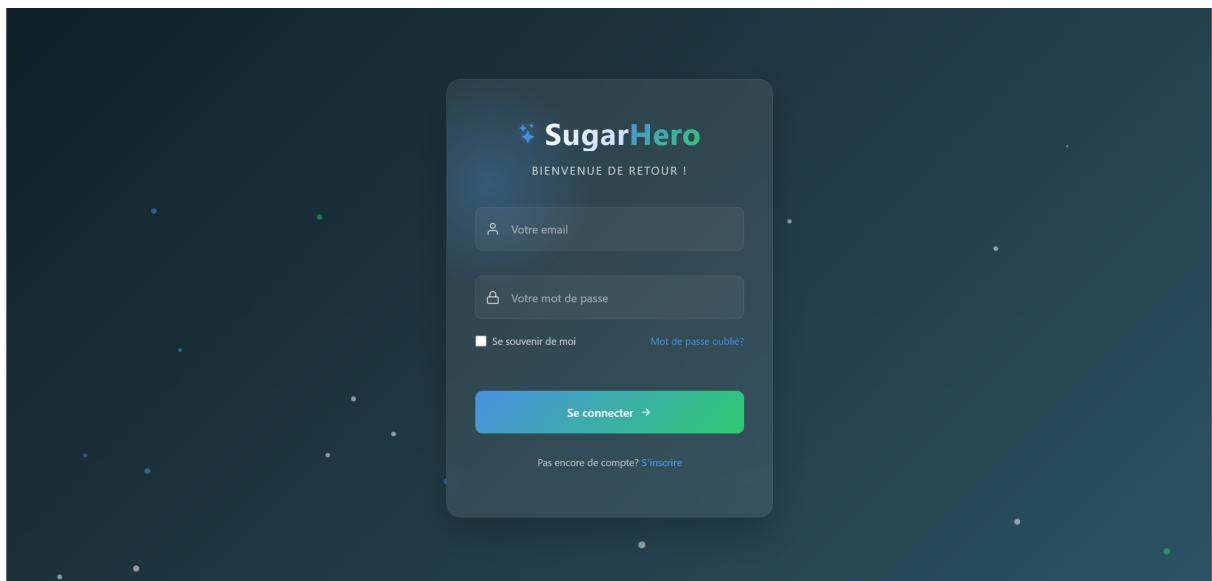


Figure 33 – interface d'authentification

Page épurée avec accès par email/mot de passe. Liens vers récupération de compte et création de profil.

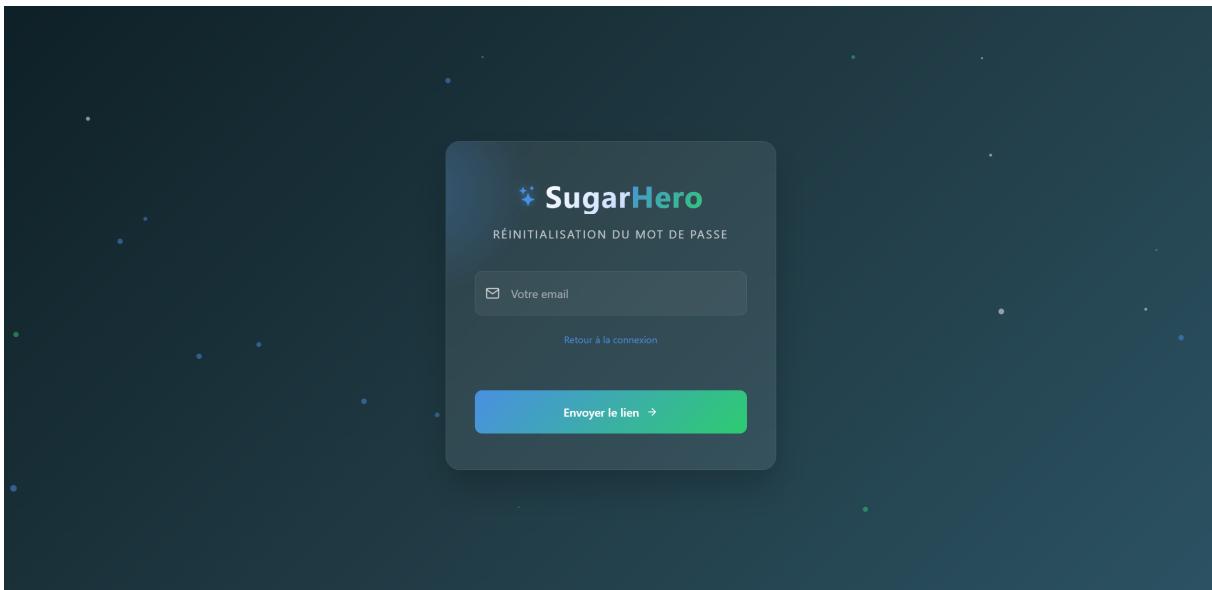


Figure 34 – interface mot de passe oublier

l'utilisateur doit entrer son email et clique sur le button envoyer le lien

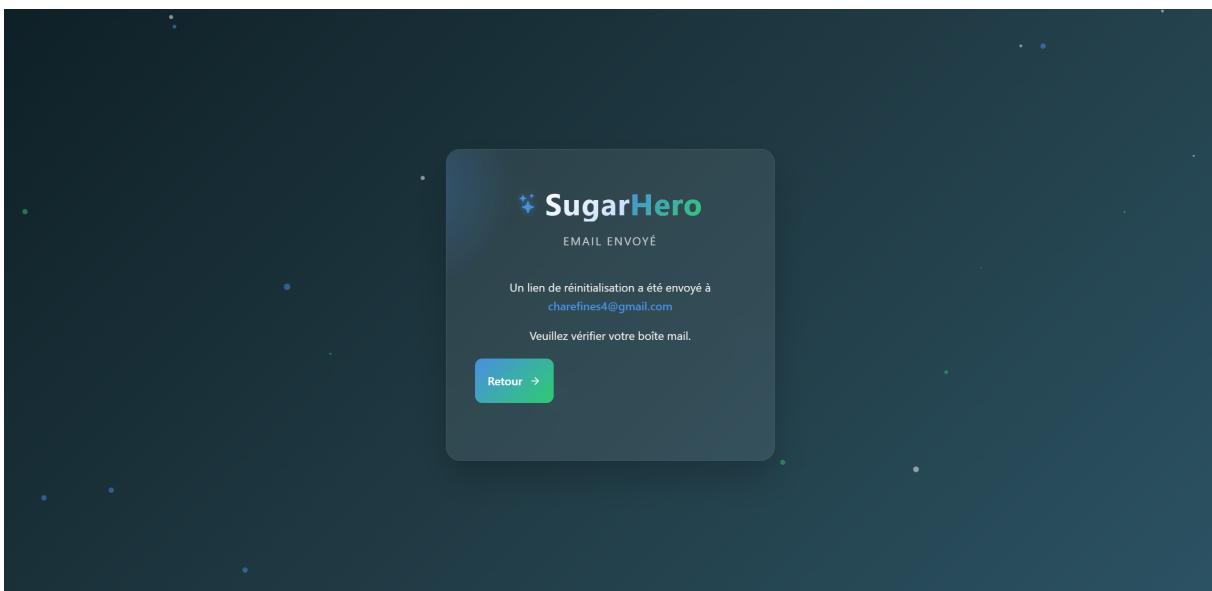


Figure 35 – interface mot de passe oublier

le lien de recuperation de compte envoyé avec sucées a l'email de l'utilisateur

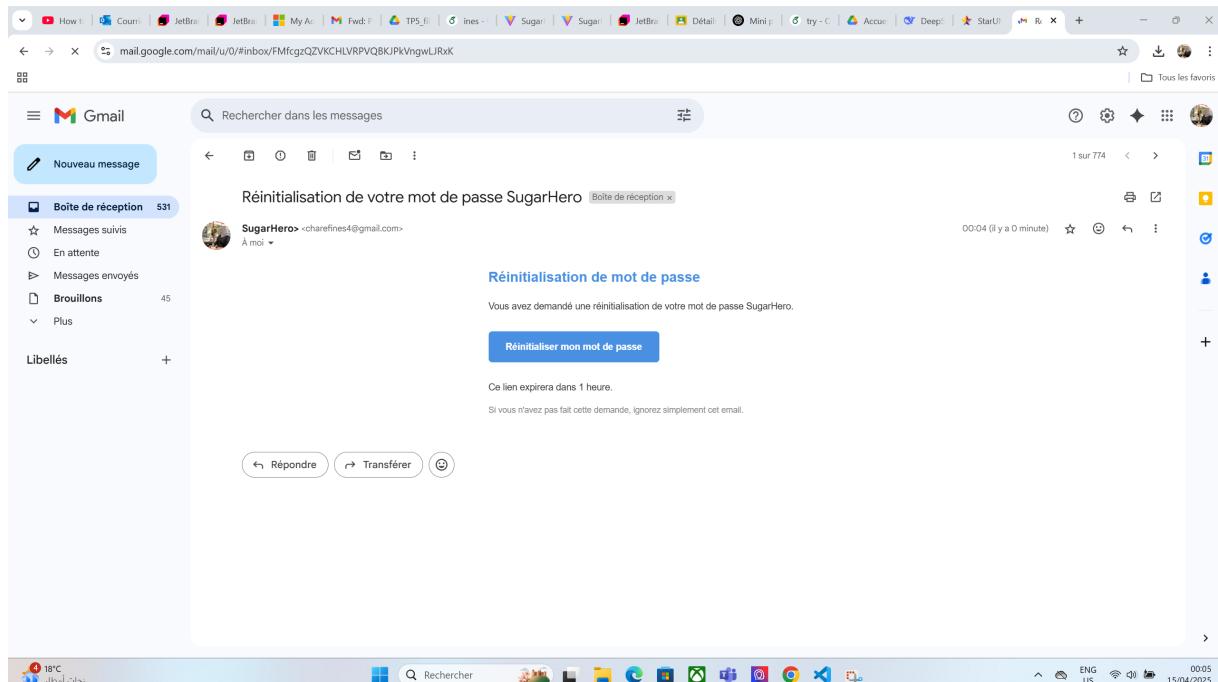


Figure 36 – interface mot de passe oublier

l'email de l'utilisateur avec le lien qui expire après 1h

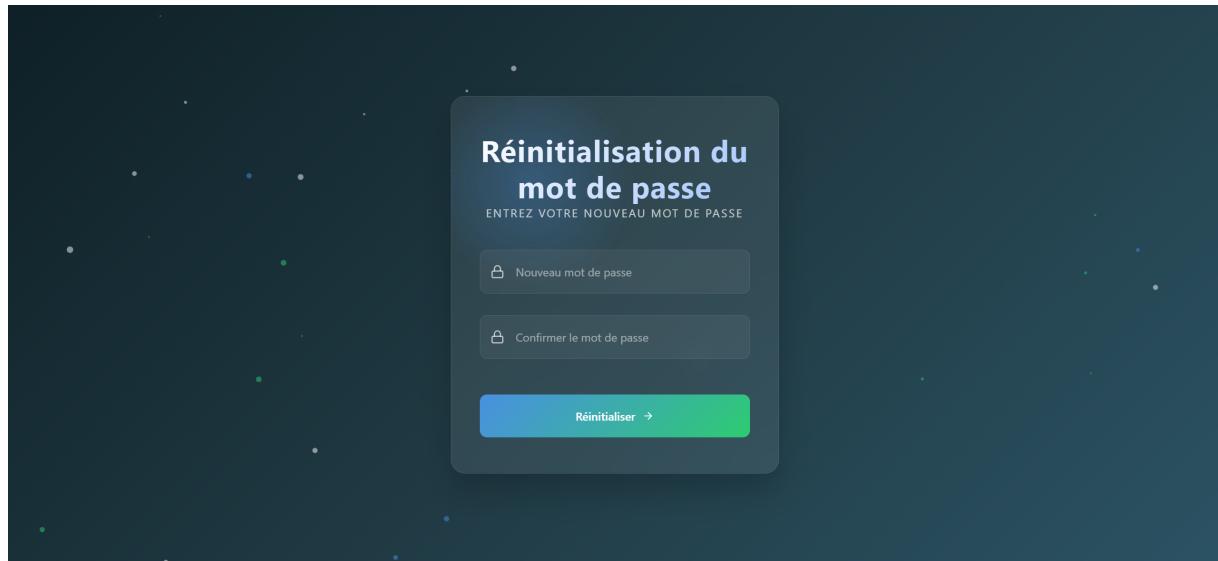


Figure 37 – interface mot de passe oublier

l'utilisateur accéder au lien et entrer le nouveau mot de passe

3.2.2 Interface utilisateur

Espace personnalisé avec accès rapide aux principales fonctionnalités. Design adapté aux enfants (couleurs vives, icônes).

Figure 38 – interface de fiche éducatives

Affichage visuel type carte colorées simple et adequat pour les enfants

Figure 39 – interface suivi de glycémie

Graphique d'évolution temporelle + formulaire de saisie.

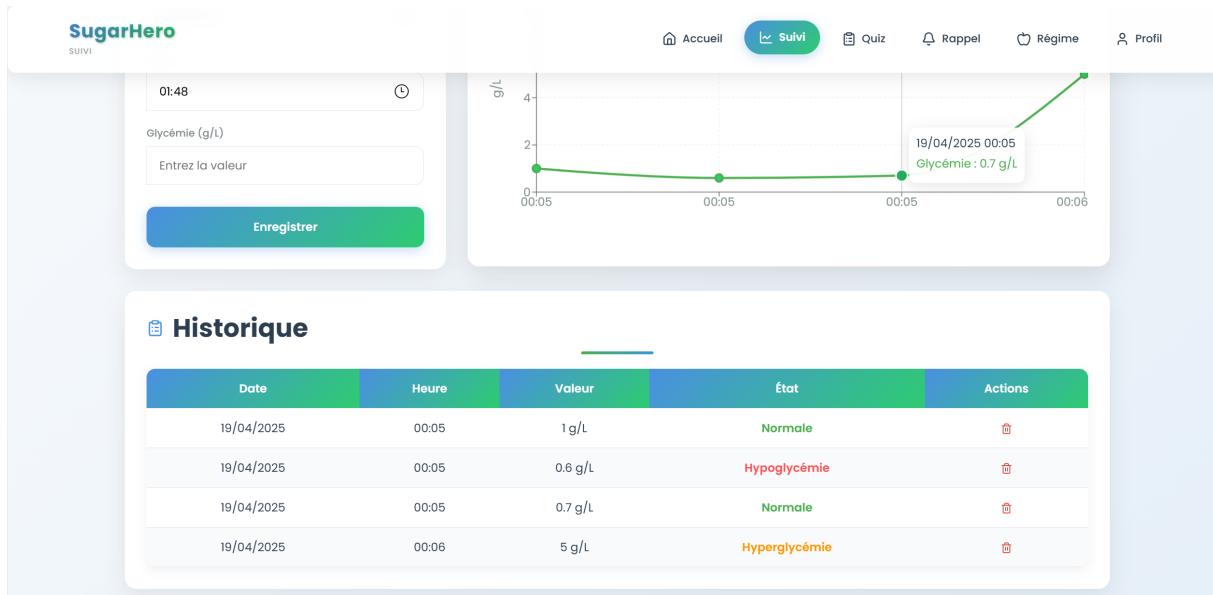


Figure 40 – interface suivi de glycémie

historique des mesures avec Date,Heure ,valeur et Alertes visuelles .

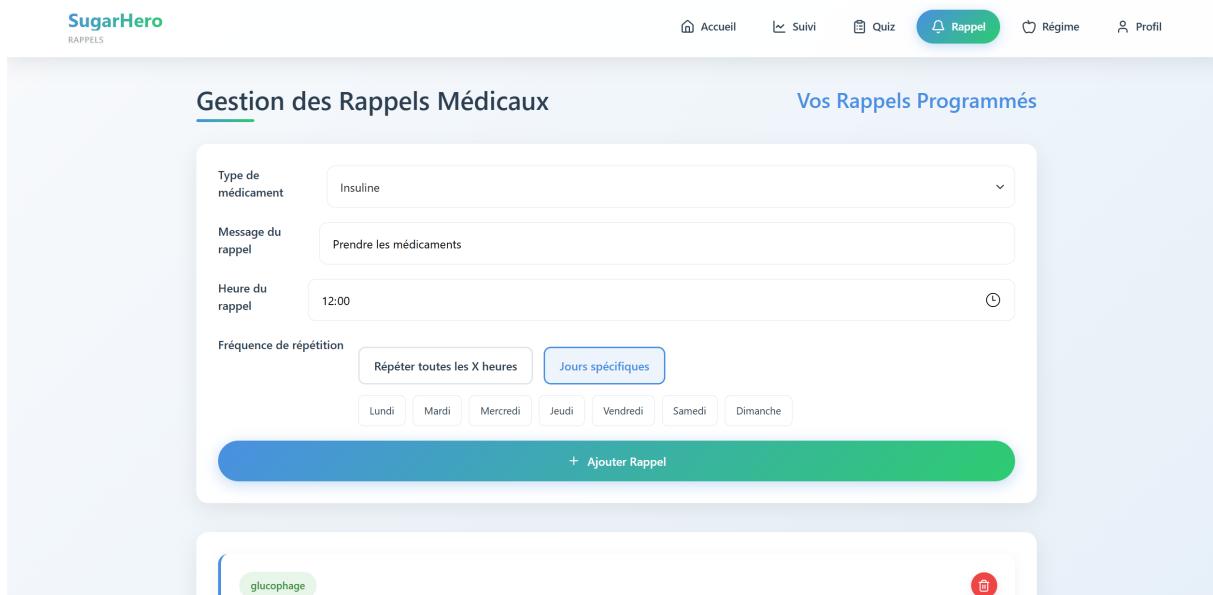


Figure 41 – interface de rappel

système de rappels avec Options de personnalisation (fréquence, type, notifications).

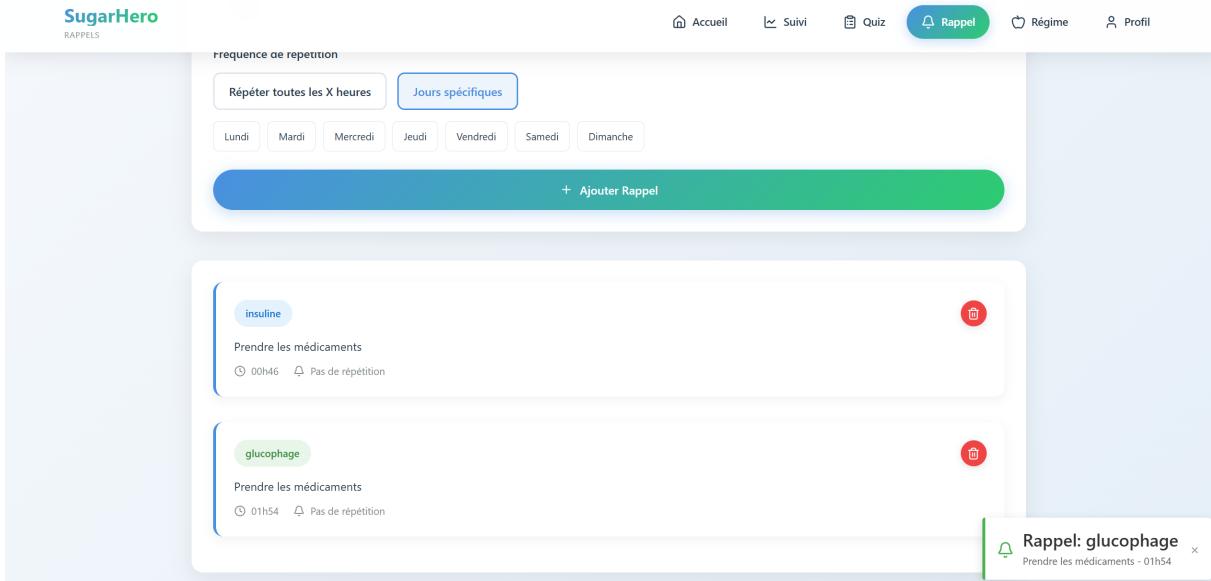


Figure 42 – interface de rappel

le notification s'affiche

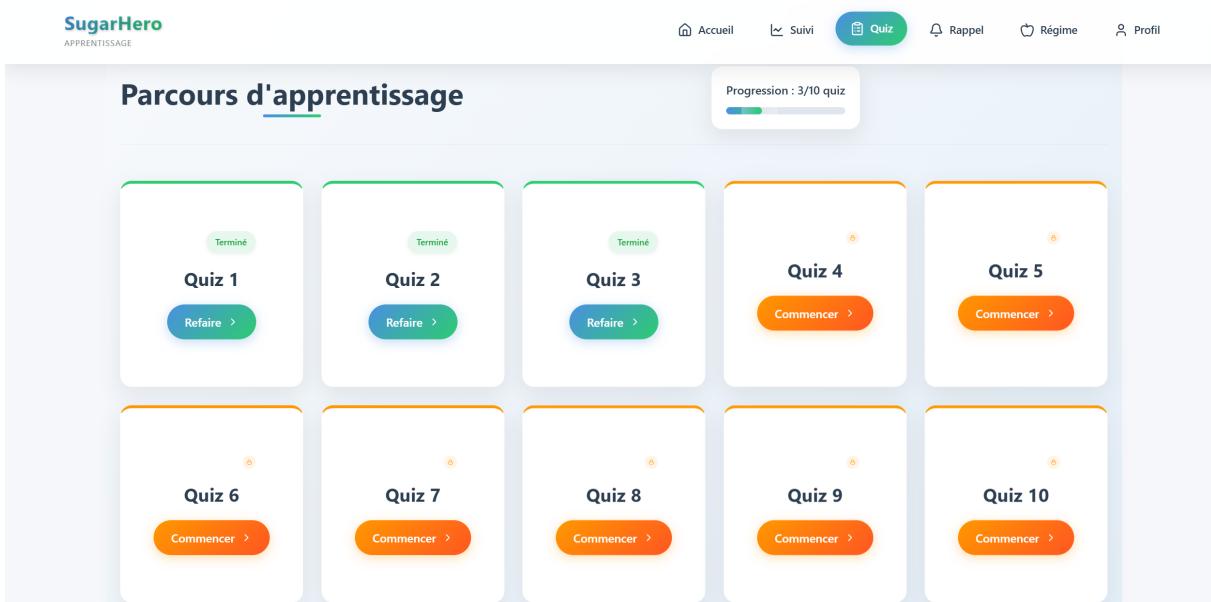


Figure 43 – interface de quiz

Affichage visuel de 10 quiz avec code couleur :

Quiz terminés : marqués en vert avec icône de validation

Quiz disponibles : en orange avec indicateur de progression



Figure 44 – interface de quiz

Réponse correcte : Affichée en vert

Réponse incorrecte : Affichée en rouge

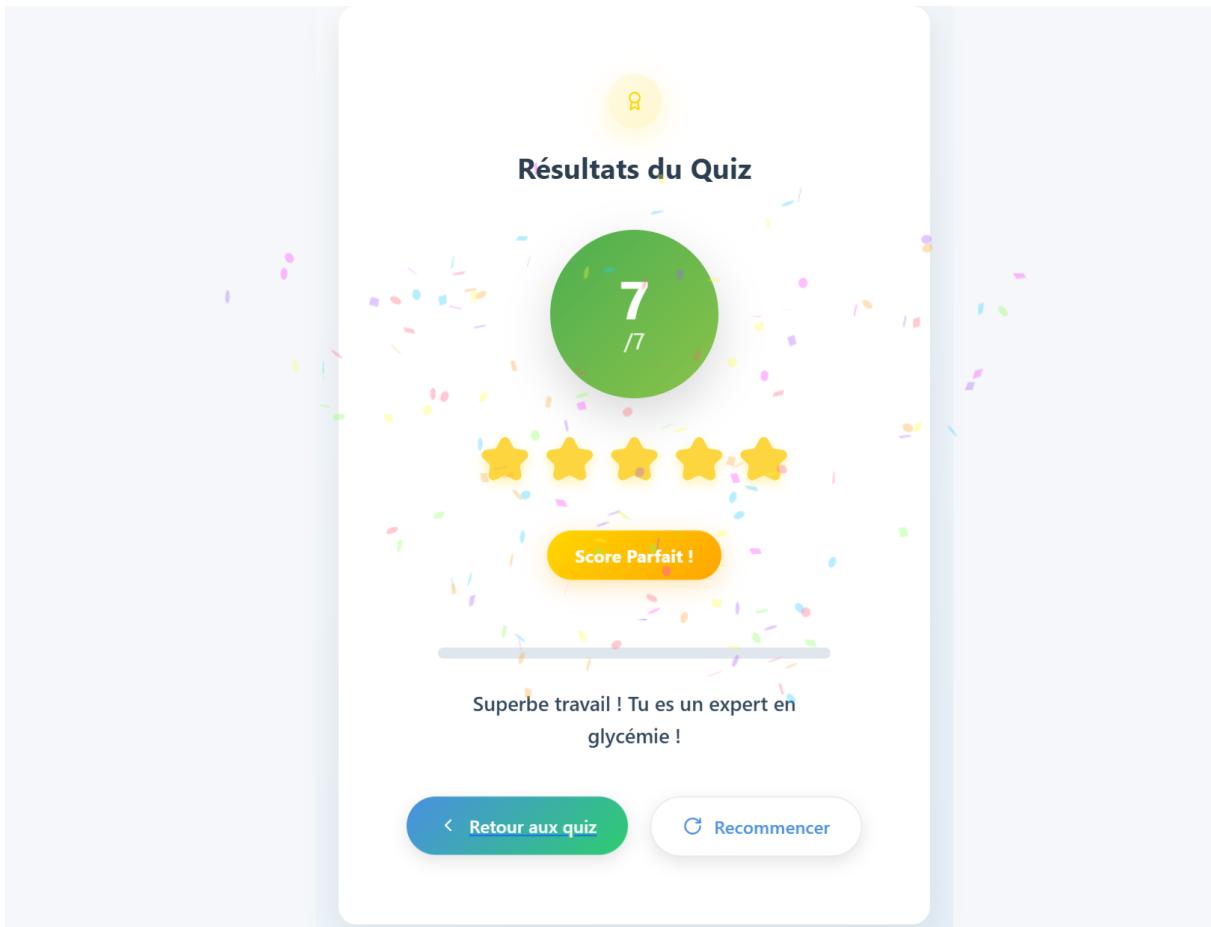


Figure 45 – interface de quiz

resultat de quiz avec des récompenses et une message de motivation

The screenshot shows the SugarHero mobile application interface. At the top, there is a navigation bar with icons for Accueil (Home), Suivi (Follow-up), Quiz (Quiz), Rappel (Reminder), Régime (Diet), and Profil (Profile). Below the navigation bar, there are two tabs: "Repas" (Meals) and "Exercices" (Exercises). The "Repas" tab is selected, indicated by a blue background.

The main area is titled "Nom du repas" (Name of the meal) and contains a text input field with placeholder text "Petit-déjeuner, Déjeuner...". Below this are three input fields: "Heure" (Time) with a dropdown menu showing "00:00", "Calories (kcal)" with the value "250", and "Glucides (g)" with the value "30". A green button labeled "+ Ajouter Repas" (Add Meal) is located below these fields.

A section titled "Vos repas aujourd'hui" (Your meals today) displays a single entry: "07:00 dejenur 150 kcal 30g glucides". To the right of this entry is a red circular delete icon with a white trash symbol.

Figure 46 – interface régime suivi repas

suivi de repas

The screenshot shows the SugarHero mobile application interface. At the top, there is a navigation bar with icons for Accueil (Home), Suivi (Follow-up), Quiz (Quiz), Rappel (Reminder), Régime (Diet), and Profil (Profile). Below the navigation bar, there are two tabs: "Repas" (Meals) and "Exercices" (Exercises). The "Exercices" tab is selected, indicated by a blue background.

The main area is titled "Nom de l'exercice" (Name of the exercise) and contains a text input field with placeholder text "Marche, Natation...". Below this are three input fields: "Durée (min)" (Duration) with the value "30", "Heure" (Time) with a dropdown menu showing "00:00", and "Calories (kcal)" with the value "200". A green button labeled "+ Ajouter Exercice" (Add Exercise) is located below these fields.

A section titled "Vos exercices aujourd'hui" (Your exercises today) displays a single entry: "16:50 Marche 30 min 150 kcal brûlées". To the right of this entry is a red circular delete icon with a white trash symbol.

Figure 47 – interface régime suivi activité physique

suivi d'activités physiques

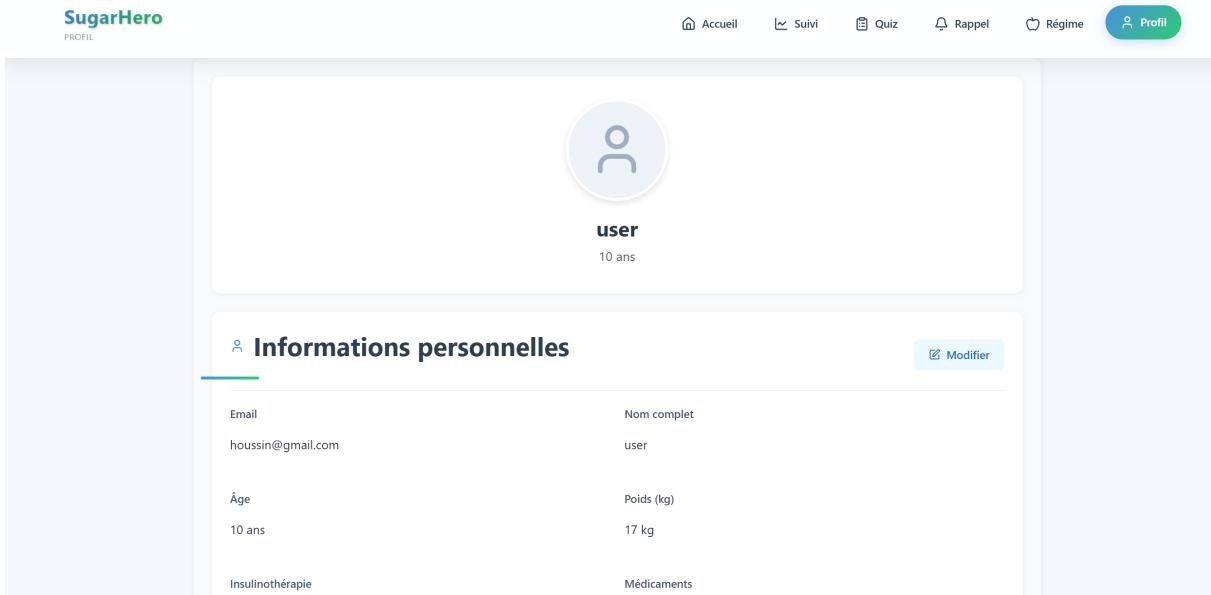


Figure 48 – interface de profil

Page personnel regroupant toutes les données. Éditeur simple pour mettre à jour les informations.

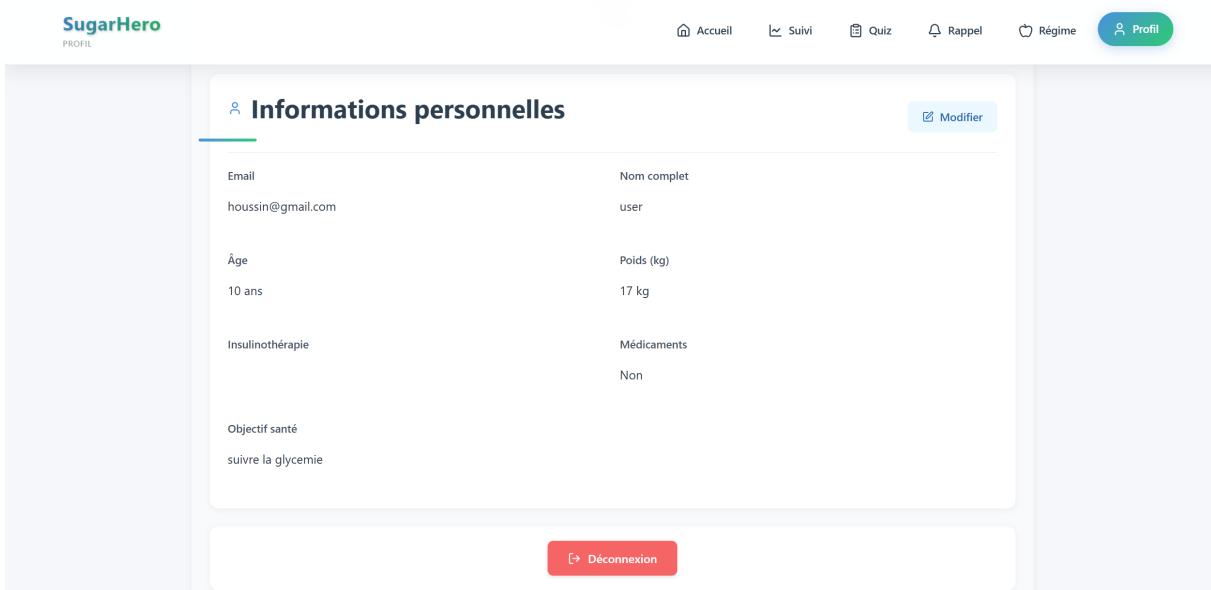


Figure 49 – interface de profil

un button de deconnexion qui redirige au page d'authentification

3.2.3 Interface administrateur

Menu d'accès rapide à toutes les fonctionnalités de gestion.

Figure 50 – interface gestion des fiches éducatives

l'admin peut modifier ou supprimer les cards éducatifs

Figure 51 – interface gestion des fiches éducatives

l'admin peut ajouter un cards éducatifs en remplir cette formulaire

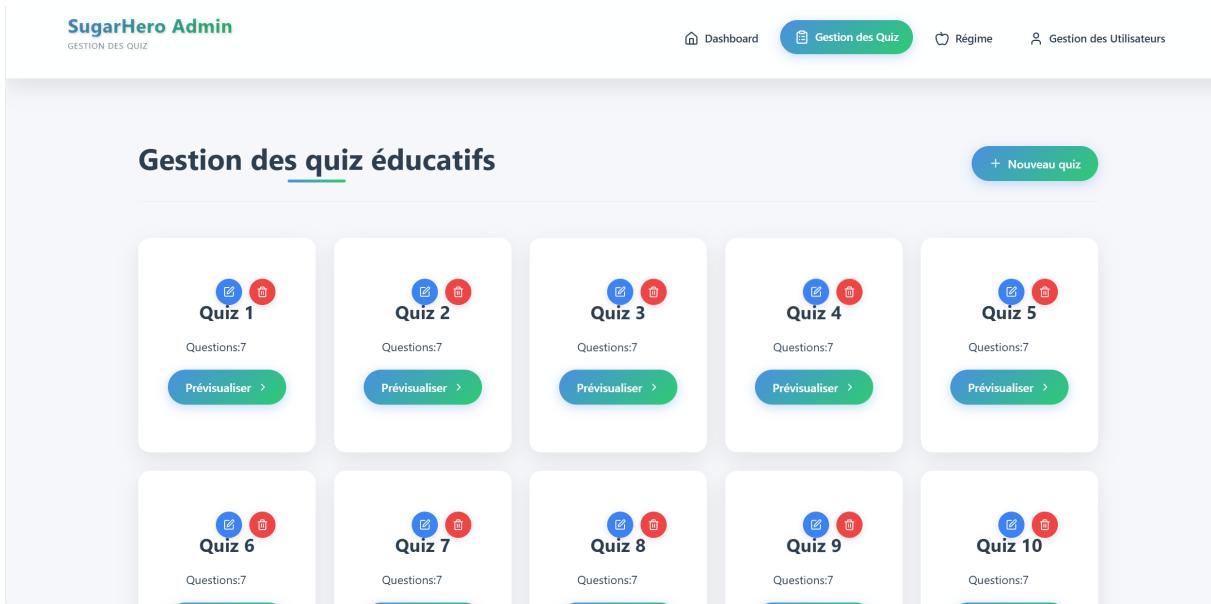


Figure 52 – interface gestion des quiz

l'admin peut modifier ou supprimer ou ajouter des quiz éducatifs

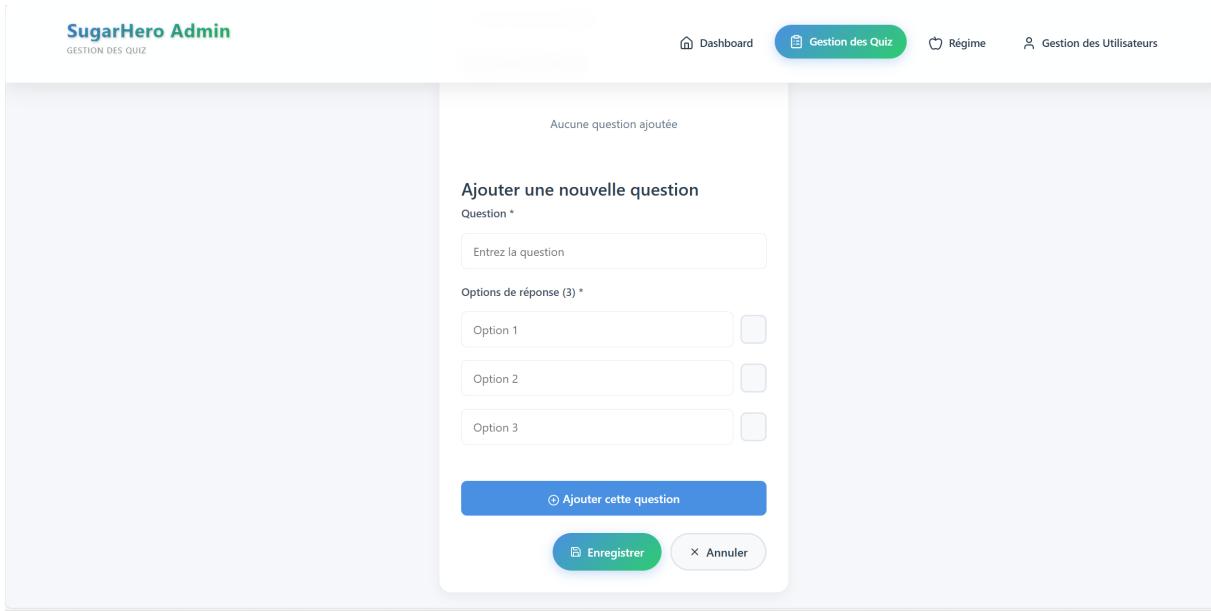


Figure 53 – interface gestion des quiz ajouter un quiz

en remplissons cette formulaire une nouvelle quiz éducatif s'affiche dans le page de quiz

SugarHero Admin
GESTION DES QUIZ

[Dashboard](#) [Gestion des Quiz](#) [Régime](#) [Gestion des Utilisateurs](#)

Questions (7)

- Question 1
Laquelle correspond à la définition de la glycémie ?
Réponse correcte: La quantité de sucre dans le sang
- Question 2
Pourquoi doit-on surveiller la glycémie ?
Réponse correcte: Pour éviter qu'elle soit trop haute ou trop basse
- Question 3
Quel est le meilleur moment pour mesurer sa glycémie ?
Réponse correcte: Toujours au même moment selon les conseils du médecin
- Question 4
Que se passe-t-il si la glycémie est trop basse ?
Réponse correcte: On a faim et froid
- Question 5
Que se passe-t-il si la glycémie est trop haute ?

Figure 54 – interface gestion des quiz modifier un quiz

l'utilisateur peut modifier les quiz éducatifs

SugarHero Admin
PANEL D'ADMINISTRATION

[Dashboard](#) [Gestion des Quiz](#) [Régime](#) [Gestion des Utilisateurs](#)

Gestion des Utilisateurs

+ Ajouter un utilisateur

| EMAIL | CRÉÉ LE | ACTIONS |
|-----------------------|------------|---------|
| mourad@gmail.com | 15/04/2025 | |
| charefines4@gmail.com | 15/04/2025 | |
| arsalen@gmail.com | 12/04/2025 | |
| ayhem@gmail.com | 13/04/2025 | |
| nessrine@gmail.com | 14/04/2025 | |
| ahmed@gmail.com | 14/04/2025 | |

Figure 55 – interface gestion des utilisateurs

l'admin peut modifier ou supprimer un utilisateur

| EMAIL | CRÉÉ LE | ACTIONS |
|-----------------------|------------|---------|
| mourad@gmail.com | 15/04/2025 | |
| charefines4@gmail.com | 15/04/2025 | |

Figure 56 – interface gestion des utilisateurs”ajouter un utilisateur

l’admin ajoute une nouvelle utilisateur en remplissons cette formulaire

Conclusion

Ce chapitre a été consacré à la modélisation UML de Sugarhero dont y compris les différents diagrammes. Dans le chapitre suivant, nous allons présenter l’environnement de travail, les outils de développement utilisé, ainsi la mise à l’essai de ce système.

Conclusion Générale

Dans le cadre de ce projet, nous avons développé une application web complète dédiée à la gestion du diabète chez les enfants. Notre solution combine une interface conviviale et ludique avec des fonctionnalités avancées de suivi médical, offrant ainsi un outil complet pour les jeunes patients et leurs familles.

Les principaux points forts de notre application incluent :

- Une interface intuitive spécialement conçue pour les enfants
- Des fonctionnalités de suivi en temps réel de la glycémie
- Un système de rappels personnalisables pour les traitements
- Des quiz éducatifs interactifs pour apprendre en s'amusant
- Une architecture technique robuste et évolutive

Ce projet nous a permis de mettre en pratique nos connaissances en développement web tout en répondant à un besoin concret dans le domaine de la santé. Les technologies modernes que nous avons utilisées (React, Node.js, MongoDB) nous ont offert une grande flexibilité dans la réalisation de nos objectifs.

Pour l'avenir, nous envisageons plusieurs améliorations possibles :

- Intégration d'algorithmes d'intelligence artificielle pour des recommandations plus personnalisées
- Développement d'une application mobile native
- Ajout de fonctionnalités de partage sécurisé des données avec les professionnels de santé
- Extension du système de gamification pour renforcer la motivation des jeunes utilisateurs

Ce projet représente une première étape prometteuse dans le développement d'outils numériques adaptés aux besoins spécifiques des enfants atteints de maladies chroniques. Nous sommes convaincus que notre application pourra apporter une réelle valeur ajoutée dans la gestion quotidienne du diabète chez les jeunes patients.

Webographie

Références

- [1] OpenClassrooms. *Ultra fast applications using Node.js.*
<https://openclassrooms.com/en/courses/applications-using-node-js>
- [2] TutorialsPoint. *Angular Tutorials.*
<https://www.tutorialspoint.com/angular/>
- [3] OpenClassrooms. *Réalisez une application web avec Angular.*
<https://openclassrooms.com/fr/courses/application-web-avec-angular>
- [4] MongooseJS. *Documentation.*
<https://mongoosejs.com/docs/guide.html>
- [5] Stack Overflow.
<https://stackoverflow.com/>
- [6] MongoDB. *Official Documentation.*
<https://docs.mongodb.com/manual/>
- [7] YouTube. *Node.js Tutorial Series.*
https://www.youtube.com/watch?v=OoXYLzuucwE&list=PL55RiY5tL51q4DB63KBnygU6opNPfk_q
- [8] Wikipedia. *Diagramme de Gantt.*
https://fr.wikipedia.org/wiki/Diagramme_de_Gantt
- [9] Wikipedia. *Angular.*
<https://fr.wikipedia.org/wiki/Angular>
- [10] Medium. *Learn React.js Basics in 3 Minutes.*
<https://medium.com/@madhupathy/learn-basics-of-react-js>
- [11] Angular. *Official Documentation.*
<https://angular.io/docs>
- [12] YouTube. *Angular Tutorial Series.*
https://www.youtube.com/watch?v=dDDy2dYqzLA&list=PLZ80Hu7iFqL9WjWk3NdrtfTzM_MK5-jXq