

N°:

			/	2	0	2	2
--	--	--	---	---	---	---	---



Université Sultan Moulay Slimane
ECOLE SUPERIEURE DE TECHNOLOGIE
–FKIH BEN SALAH–

Diplôme Universitaire de Technologie

Génie Informatique

Rapport de stage de Fin d'Etudes

Titre :
**APPLICATION DE
GESTION D'IMMEUBLES**

●●●● Réalisé par

Boubcher Issam

●●●● Soutenu le

01/07/2022

devant le jury

composé de :

Président : Prof. Amine Abdellah

Encadrent : Prof. Ait Daoud Rachid

Année Universitaire : **2021/2022**

Remerciements

Au terme de ce travail, je tiens à présenter mes sincères remerciements à **Mr. Lahsen El Bouhali** et à tout le staff de l'Ecole 1337 pour leur pédagogie, leur patience, leur disponibilité, leur dévouement et leurs conseils fructueux. Leurs capacités techniques et leurs compétences sont vraiment incroyables et leur énorme soutien a été crucial pour la réussite de ce travail.

Aussi, je profite de cette occasion pour exprimer ma profonde gratitude à **M. Amine Abdellah**, notre coordinateur de branche, qui a inlassablement fait tout son possible pour garantir le succès de notre formation.

Nos remerciements vont également à **Mr. Rachid Ait Daoud**, Mr. Hassan Faouzi et tous nos professeurs de l'EST Fquih Ben Salah, les excellents professeurs qui nous ont apporté une multitude de connaissances techniques et analytiques et nous ont accompagné tout au long de notre parcours académique.

En outre, nous tenons à remercier les membres du jury supervisant notre soutenance, pour leur temps précieux, et pour toutes leurs remarques et recommandations visant à améliorer encore plus notre projet.

Enfin à tous ceux qui nous ont aidés de près ou de loin à réaliser ce travail, trouvent ici l'expression de notre estime et nos vifs remerciements.

Résumé

Dans le cadre de la création d'une solution qui résout la multitude de problèmes rencontrés par les copropriétaires des immeubles, nous allons concevoir et créer une application web qui centralisera toutes les actions relatives à la gestion des immeubles.

Une solution complète et innovante de gestion de copropriété qui prend en compte toutes les fonctionnalités indispensables à l'activité d'un syndic serait extrêmement bénéfique à tout copropriétaire. Elle permettrait de faciliter l'accès à l'information sur qui, quand et pour combien quelqu'un a contribué à la trésorerie de l'immeuble, ainsi que de fournir des informations claires et détaillées sur la façon dont les fonds apportés sont dépensés. Ce qui assure une transparence totale de la collecte et de la dépense des fonds.

Notre solution, hébergée sur le web sous le lien www.skyline-app.ga dans une infrastructure cloud Heroku, permet aux utilisateurs d'ajouter les immeubles dans lesquels ils possèdent ou louent des appartements, d'inviter leurs copropriétaires à rejoindre l'immeuble, de donner à certains des privilèges d'administrateur sur l'immeuble, ainsi que de gérer la collecte des fonds des copropriétaires et de dépenser ces fonds pour diverses actions liées à l'amélioration de leur espace commun, comme des améliorations de l'immeuble, des réparations, etc.

Abstract

As part of creating a solution that solves the multitude of problems encountered by co-owners, we will conceive and create a Web application which will centralize all the actions relating to managing buildings.

A complete and innovative solution of co-ownership management which takes into account all the functionalities essential to the activity of a syndic would be extremely beneficial to any co-owner. It would allow easy access to information on who, when and how much someone has contributed to the building's treasury, as well as provide clear and detailed information on how the contributed funds are spent. This ensures total transparency of the collection and spending of funds.

Our solution, hosted on the web at www.skyline-app.ga in a Heroku cloud server, allows users to add buildings where they own or rent apartments to the application, invite other co-owners to join the building, give some of them administrator privileges over the building, as well as manage the collection of co-owner funds and spend these funds on various actions related to the improvement of their common space, such as building improvements, repairs, etc.

Table des matières

Présentation École 1337	7
Introduction générale	8
Chapitre 1 : Élaboration de cahier de charges	9
1. Introduction :.....	9
2. Problématique :.....	9
3. Spécification générale :.....	10
3.1. Objectif de projet :.....	10
4. Spécifications fonctionnelles :	10
5. Spécifications d'interface :.....	11
6. Spécifications non fonctionnelles :.....	12
7. Choix d'une application web :	12
8. Conclusion :.....	12
Chapitre 2 : Conceptualisation de l'application	13
1. Introduction :.....	13
2. Le choix d'UML :.....	13
3. Diagrammes UML :	14
3.1 Diagrammes des cas d'utilisation :.....	14
3.2. Diagramme de classes :.....	17
3.3. Diagrammes de séquence :	18
Chapitre 3 : Réalisation de l'application	20
1. Introduction :.....	20
2. L'environnement de travail :.....	20
2.1. Visual Studio Code :.....	20
2.2. Git :.....	21
2.3. GitHub :.....	21
3. Technologies utilisées :.....	22
3.1. Front-end:.....	22
3.2. Back-end :	24
3.3. Base de données :.....	25
3.4. Client AJAX : React Query	27
3.5. Authentification : Auth0	27

3.6. Hébergement : Heroku.....	28
3.7. CDN : Cloudflare.....	28
4. Autres outils :	29
5. L'architecture de l'application :	30
(1) Cryptage de trafic :	30
(2) Établir une connexion à Heroku :.....	31
(3) Communication avec la base de données :	31
(4) Récupération et stockage des photos :	32
6. Optimisations :	32
6.1. Optimisation de sécurité :	32
6.2. Optimisation de performance :	33
6.3. Optimisation de déploiement :	36
7. Présentation de l'application :	37
7.1. Introduction :	37
7.2. Plan de navigation de l'application :	39
7.3. Page Immeubles :	40
7.4. Page paiements (cotisations) :	50
7.5. Page dépenses :	52
7.6. Page Invitations :	53
8. Conclusion :	55
Conclusion Générale et Perspectives	56
Références.....	57

Présentation École 1337



1337 est la première école de codage au Maroc, ouverte 24/7 et accessible à tous sans prérequis de diplôme, ou de connaissance en informatique.

La pédagogie de Treize, Trente-Sept s'articule autour du peer-learning. Un fonctionnement participatif qui permet aux étudiants de libérer leur créativité grâce à l'apprentissage par projet. Elle offre un environnement formidable dans lequel les esprits brillants peuvent prospérer.

C'est ça le secret pour lequel de nombreuses entreprises viennent à l'école pour essayer d'attirer ses étudiants.

Introduction générale

Le monde d'aujourd'hui est un témoin d'un progrès énorme dans différents domaines et plus particulièrement dans le domaine de la technologie de l'information. Ce progrès remarquable pousse les entreprises à trouver des solutions pour automatiser leurs tâches quotidiennes afin de réaliser des produits et des services de manière plus rapide et plus facile.

Le présent travail s'inscrit dans le cadre du projet de stage en vue de l'obtention de Diplôme Universitaire de Technologie (DUT) pour l'année universitaire 2021/2022 en génie informatique, dans lequel nous allons concevoir et implémenter une solution web de Gestion des Immeubles.

Dans ce but, ce rapport englobe toutes les ressources utilisées pendant la conception et le développement pour la réalisation de cette application web.

Le présent document est organisé comme suit :

Le Premier chapitre consiste à élaborer le problème que nous essayons de résoudre, et établir le cahier de charges que nous suivrons pour conceptualiser l'application.

Dans **le deuxième chapitre**, on détaille la partie analyse et conceptualisation de notre application.

Le troisième chapitre présente l'environnement de développement logiciel et les outils que nous avons utilisés pour la réalisation notre application, ainsi que toutes les optimisations que nous avons mises en œuvre. À la fin, nous terminerons par une présentation complète de toutes les fonctionnalités de l'application.

Chapitre 1 :

Élaboration de cahier de charges

1. Introduction :

Cette partie est consacrée pour la présentation de la problématique et le cahier des charges de notre application web qui fait le sujet de ce projet. Un cahier des charges sera un atout considérable dans le processus de création de notre application et dans la réussite du projet.

Au cours de ce chapitre, nous allons introduire notre projet en étudiant son cadre général qui nous a poussé à réaliser cette application.

Notre mission est d'analyser les besoins des **copropriétaires** et de suivre les étapes nécessaires pour élaborer et développer une application répondant à ces nécessités.

2. Problématique :

Les copropriétaires sont confrontés à une multitude de problèmes lorsqu'il s'agit de gérer un espace commun. Tout d'abord, il est difficile pour les copropriétaires de savoir qui habite dans quel appartement, ainsi que leurs informations de contact (numéro de téléphone, adresse e-mail, etc.) et il peut leur falloir un certain temps pour savoir qui est le syndic de l'immeuble.

De plus, il est difficile de savoir quels copropriétaires ont versé la cotisation mensuelle et lesquels ne l'ont pas fait.

Une autre question qui a un impact important sur les interactions entre les copropriétaires et le syndic est de savoir où sont dépensés exactement les fonds de l'immeuble.

À partir de là, nous pouvons déterminer que la **transparence** totale et complète est la plus grande exigence dont les copropriétaires ont besoin. Qui contribue aux fonds et qui ne le fait pas ? comment les fonds sont-ils dépensés ? et combien reste-t-il dans la trésorerie de l'immeuble à un moment précis ? toutes ces questions doivent être résolues.

3. Spécification générale :

Il s'agit de mettre en place un système qui permet :

- Créez des immeubles
- Invitez les utilisateurs aux immeubles.
- Gérez les invitations.
- Gérer les contributions.
- Gérer les dépenses.

3.1. Objectif de projet :

L'idée concerne généralement la mise en place d'une application unifiant tous les processus de colocation. En effet, il est difficile pour tous les habitants d'un immeuble de se rappeler qui habite dans quel appartement, comment les contacter, qui est le syndic de l'immeuble, ainsi que qui contribue à la trésorerie de l'immeuble et qui ne le fait pas, et où sont dépensés ces fonds.

4. Spécifications fonctionnelles :

Un utilisateur peut :

- Créer des immeubles dans l'application, avec leur nom, leur localisation, leur ville, leur surface globale, le nombre d'appartements, et pour chaque appartement, son identifiant et sa surface (ce qui fait de cet utilisateur le Créateur Syndic de l'immeuble.).
- Accepter une invitation à rejoindre un immeuble.
- Refuser une invitation à rejoindre un immeuble.

Un Résident d'un immeuble peut :

- Avoir toutes les fonctionnalités d'un utilisateur, et :
- Inviter d'autres utilisateurs à rejoindre un immeuble (nécessite la validation d'un Administrateur Syndic de même immeuble).
- Voir toutes les informations sur un immeuble.
- Voir le rôle, le nom, l'adresse e-mail et le numéro de téléphone ainsi que les numéros d'appartement de tous les autres résidents.
- Quitter l'immeuble.
- Consulter l'historique complet de toutes les contributions versées à la trésorerie d'un immeuble.
- Consulter l'historique complet de toutes les dépenses et à quoi elles ont servi.

Un Administrateur Syndic d'un immeuble peut :

- Avoir tous les privilèges d'un résident, et :
- Inviter d'autres utilisateurs et les faire monter en administrateurs.
- Envoyer des invitations qui ne nécessitent pas la validation d'un administrateur.
- Ajouter une cotisation d'un résident à l'immeuble, ce qui augmente sa trésorerie.
- Ajouter une dépense à l'immeuble, ce qui diminue sa trésorerie.
-

Un Créateur Syndic d'un immeuble peut :

- Avoir tous les privilèges d'un Administrateur Syndic, et :
- Faire retirer un résident de l'immeuble.
- Retirer les privilèges d'administrateur d'un Administrateur Syndic.
- Désactiver un immeuble (ce qui retire tous ses résidents, mais ne supprime pas l'immeuble lui-même).

5. Spécifications d'interface :

L'interface de l'application doit être facile à comprendre et intuitive. Le design doit être professionnel, clair et moderne. Les couleurs ne doivent pas être excessives et elles doivent être utilisées en fonction de leur teinte pour attirer l'attention de l'utilisateur aux endroits appropriés.

6. Spécifications non fonctionnelles :

Disponibilité :

L'application devra être constamment disponible et accessible à tout moment 24/7.

Sécurité :

L'application devra respecter la confidentialité des données, et doit crypter tout le trafic entrant et sortant du client.

Performance :

L'application doit être avant tout performante. C'est à-dire à travers ses fonctionnalités, elle devra répondre à toutes les exigences des utilisateurs d'une manière optimale.

Temps d'accès acceptable :

Toute interaction avec les applications ne doit pas prendre plus de 10 secondes en présence d'une connexion internet optimale.

Plateforme :

L'application doit être accessible via l'internet par tout navigateur moderne tel que Google Chrome, Microsoft Edge, Safari Browser, Mozilla Firefox et Opera Browser.

7. Choix d'une application web :

Une application web désigne un logiciel applicatif hébergé sur un serveur et accessible via un navigateur web.

Contrairement à un logiciel traditionnel, l'utilisateur d'une application web n'a pas besoin de l'installer sur son ordinateur. Il lui suffit de se connecter à l'application à l'aide de son navigateur. La tendance actuelle est d'offrir une expérience utilisateur et des fonctionnalités équivalentes aux logiciels directement installés sur les ordinateurs, sans avoir besoin d'installer, de gérer et de maintenir l'application sur tous les postes de l'équipe. Pour ces raisons, nous avons décidé de réaliser notre projet sous la forme d'une application web.

8. Conclusion :

L'établissement d'un cahier des charges solide était primordial pour le développement de notre application. Il nous a permis d'avoir un objectif final réaliste mais ambitieux à atteindre, tout en respectant la contrainte de temps.

Chapitre 2 :

Conceptualisation de l'application

1. Introduction :

Dans cette partie nous traitons l'aspect conceptuel de notre application. Pour la conception et la réalisation de cette dernière, nous nous utilisons le formalisme UML basé sur les diagrammes et offrant une flexibilité marquante.

2. Le choix d'UML :

UML (Unified Modeling Language) est le langage de modélisation le plus populaire dans le monde. Il est né de la fusion de plusieurs méthodes existantes auparavant est devenu une référence en termes de modélisation objet, alors il est utilisé dans la majorité des projets logiciels. Le choix d'UML vient pour plusieurs raisons dont :

- L'élaboration des modèles objet, indépendamment de tout langage de programmation, l'UML permet donc de normaliser les concepts objet.
- UML est un support de communication performant : Il cadre l'analyse et facilite la compréhension de représentations abstraites.
- La structuration cohérente des fonctionnalités et des données.

3. Diagrammes UML :

3.1 Diagrammes des cas d'utilisation :

Les diagrammes des cas d'utilisation (Use Case Diagram) constituent la première étape d'analyse UML en modélisant les besoins des utilisateurs, identifiant les grandes fonctionnalités et les limites du système et représentant les interactions entre le système et ses utilisateurs.

Voici les diagrammes de cas d'utilisation que nous avons conçus pour servir de base aux fonctionnalités de notre application :

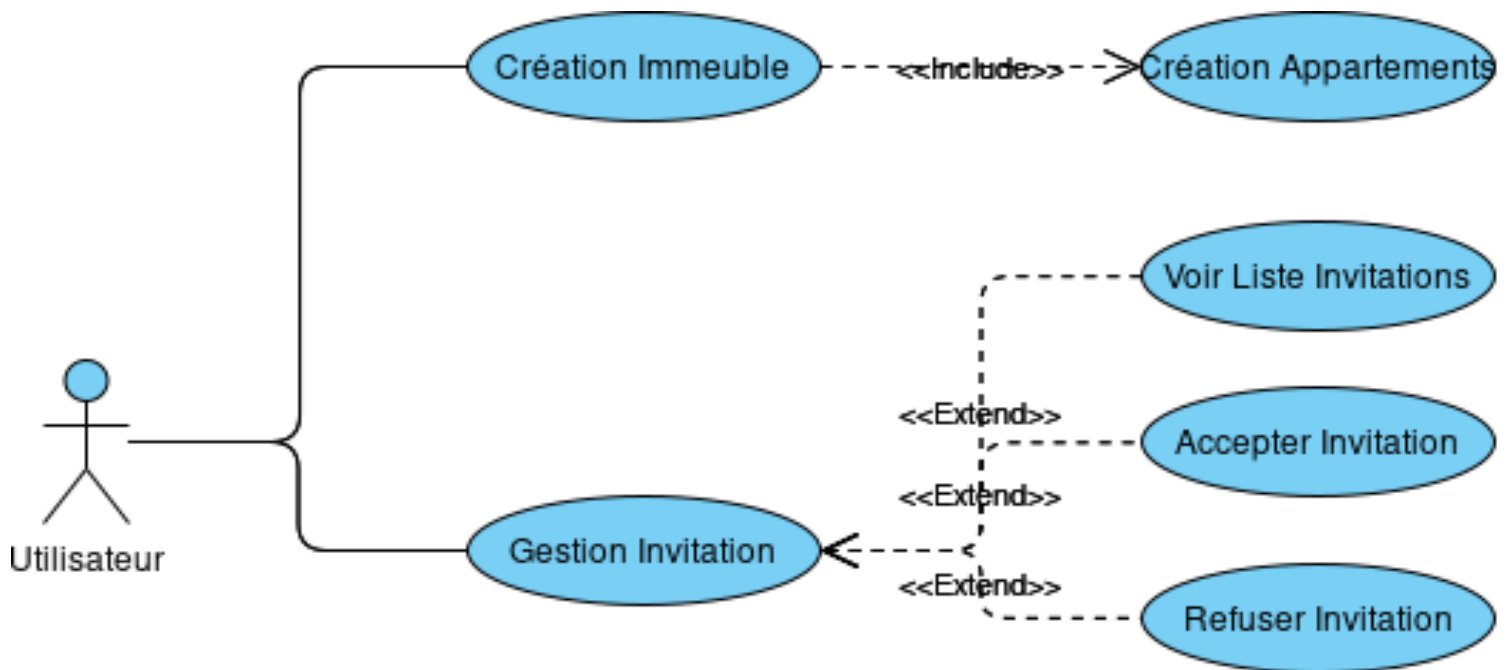


Figure 1 : Diagramme cas d'utilisation d'utilisateur

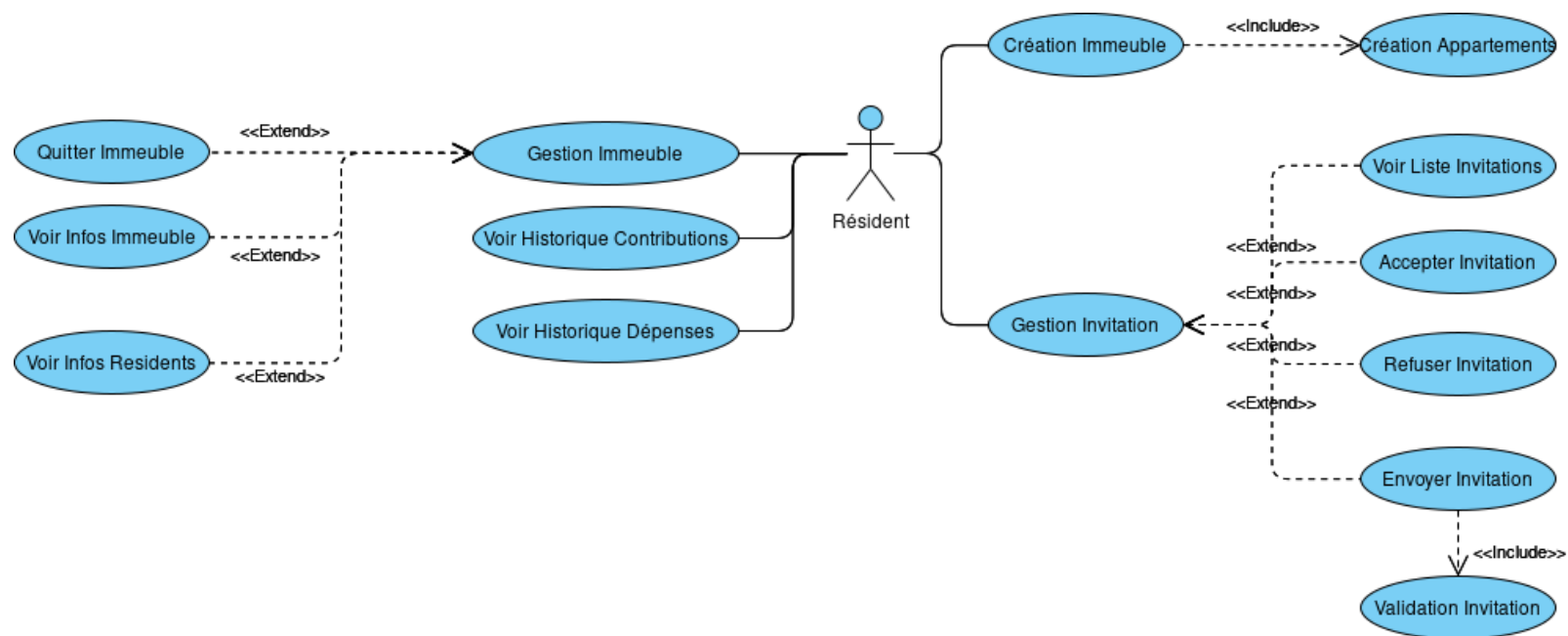


Figure 2 : Diagramme cas d'utilisation d'un résident

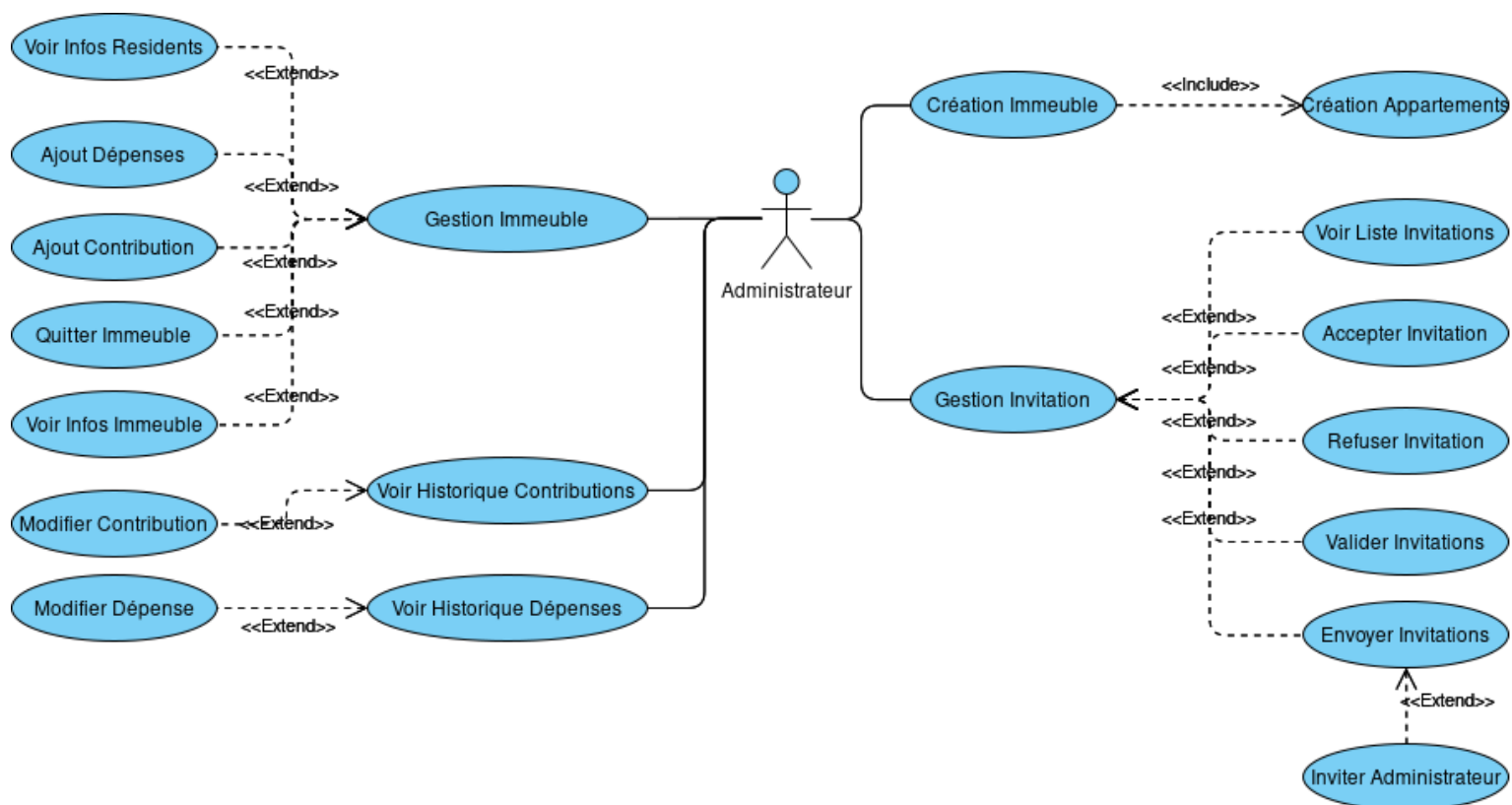


Figure 3 : Diagramme cas d'utilisation d'un administrateur

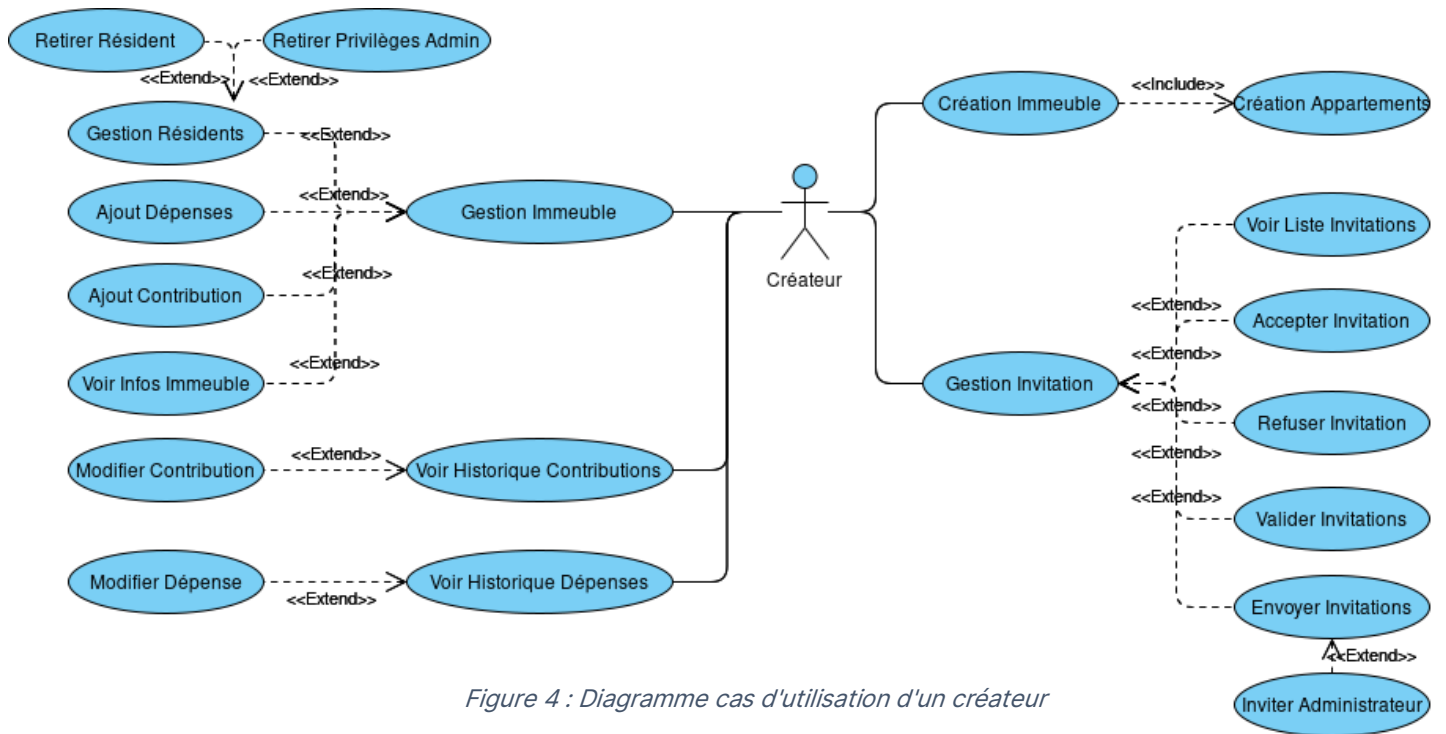


Figure 4 : Diagramme cas d'utilisation d'un créateur

3.2. Diagramme de classes :

Le diagramme de classes permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Et il nous permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

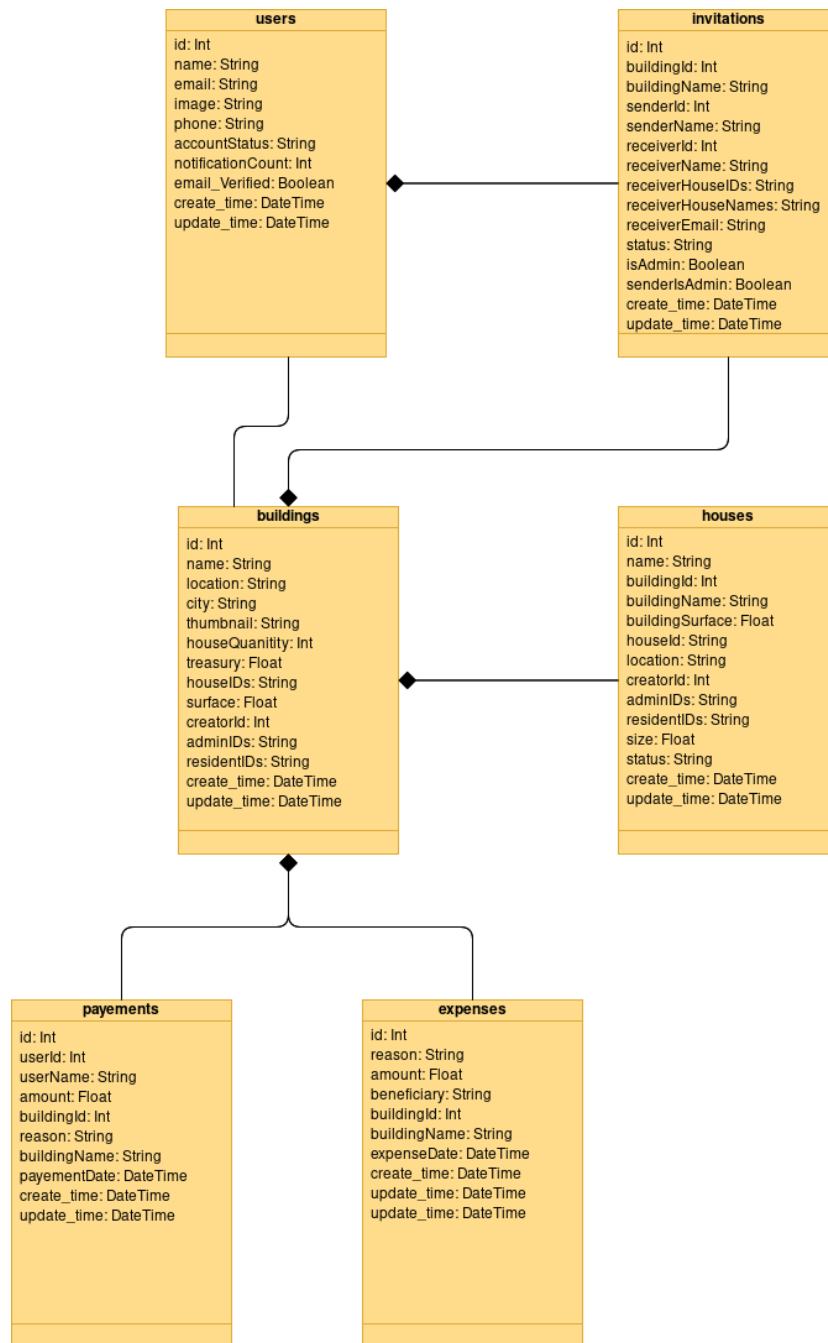


Figure 5: Diagramme de classes

3.3. Diagrammes de séquence :

Les diagrammes de séquence sont une solution de modélisation dynamique en langage UML, ils se concentrent plus précisément sur les lignes de vie, les processus et les objets qui vivent simultanément, et les messages qu'ils échangent entre eux pour exercer une fonction avant la fin de la ligne de vie.

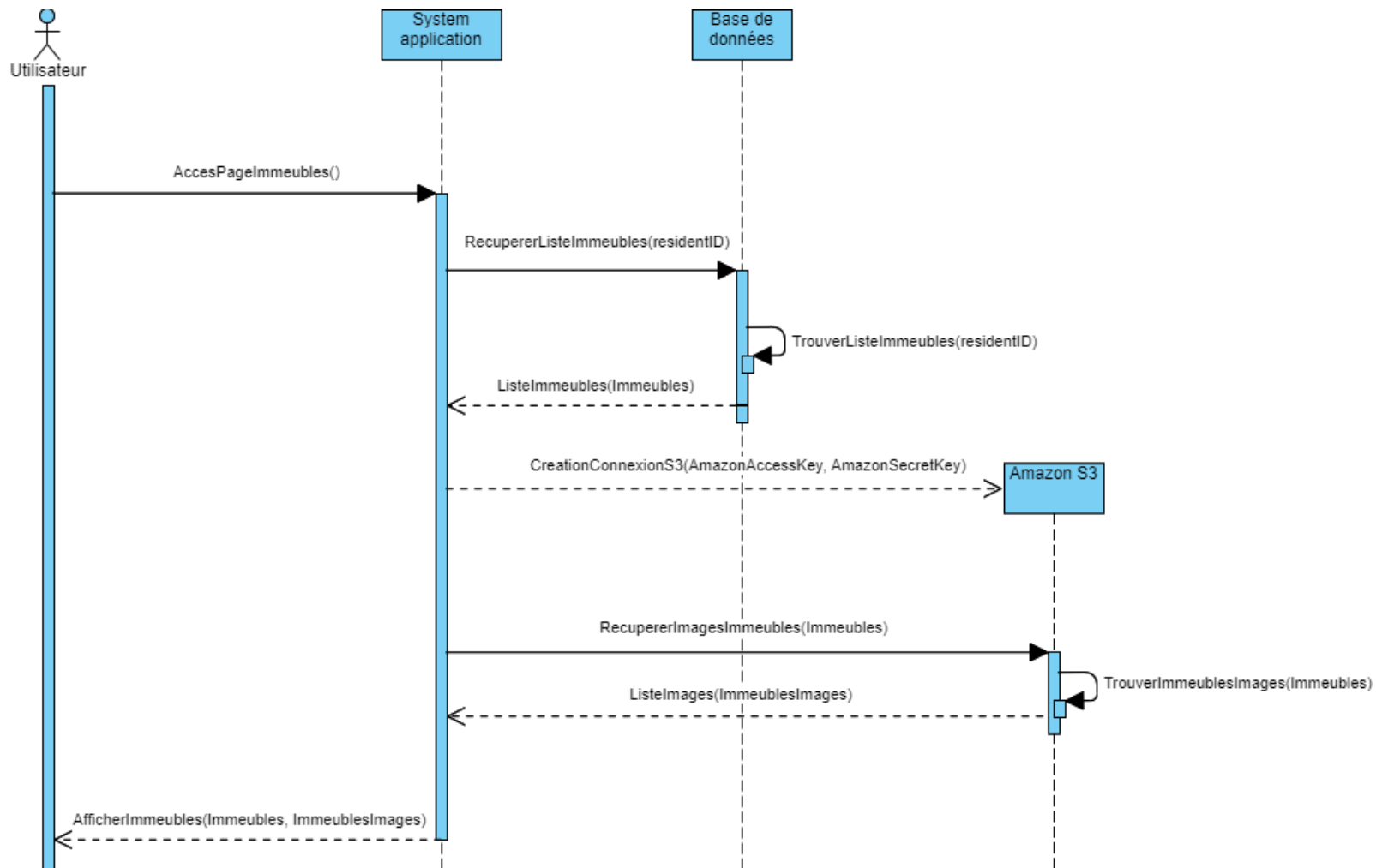


Figure 6 : Diagramme de séquence d'accès au page Immeubles

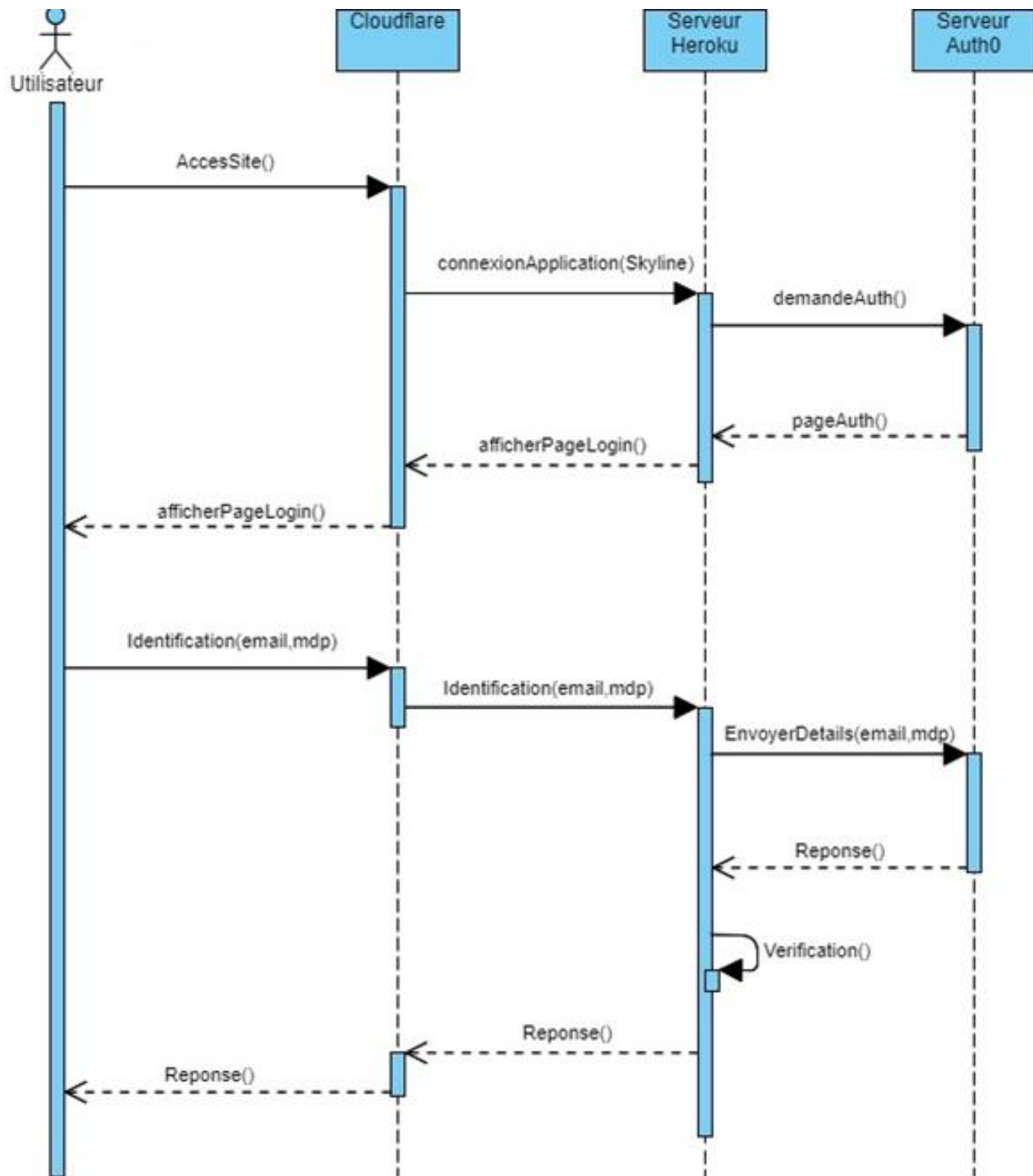


Figure 7 : Diagramme de sequence d'authentification

Chapitre 3 :

Réalisation de l'application

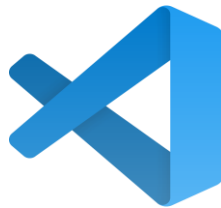
1. Introduction :

Au cours de cette partie, nous allons aborder l'ensemble des technologies que nous utilisons pour répondre aux besoins fonctionnels ainsi que les optimisations que nous avons faite pour améliorer et solidifier l'application, et nous terminerons par une visite détaillée de toutes les fonctionnalités que nous avons mises en place.

2. L'environnement de travail :

Dans cette partie, nous présentons les différents outils nécessaires pour le développement de notre application.

2.1. Visual Studio Code :



Visual studio code ou VS Code est un éditeur de code développé par Microsoft en 2015. Il supporte un très grand nombre de langages grâce à des extensions, la complétion intelligente du code, la coloration syntaxique, le débogage et les commandes git.

2.2. Git :



Git est un logiciel permettant de suivre les modifications apportées à un ensemble de fichiers. Il est généralement utilisé pour coordonner le travail des programmeurs qui développent en collaboration le code source pendant le développement de logiciels.

2.3. GitHub :



GitHub est un fournisseur d'hébergement Internet pour le développement de logiciels et le contrôle de version utilisant Git. Il offre les fonctionnalités de contrôle de version distribué et de gestion du code source de Git, ainsi que ses propres fonctionnalités. Nous avons utilisé Github comme répertoire central pour notre code. Et aussi parce qu'il nous permet de déployer automatiquement notre code sur Heroku, nous en parlerons dans la section optimisation du déploiement.

3. Technologies utilisées :

3.1. Front-end:

3.1.1. HTML:



HyperText Markup Language est un langage de balisage qui a été créé par Tim Berners-Lee en 1991. Il nous permet de définir les différents contenus d'une page.

Le balisage indique au navigateur web comment présenter à l'utilisateur les mots et les images d'une page web sur l'internet. Bien que chaque code de balisage individuel soit un élément à proprement parler, on les appelle communément des balises. Certains éléments, présentés sous forme de paires, indiquent le début et la fin de l'effet d'affichage.

3.1.2. CSS :



Cascading StyleSheets (feuilles de styles en cascade) est un langage de styles, il a été créé par Håkon Wium Lie en 1996, soit 5 ans après le HTML. Le CSS vient résoudre un problème bien différent du HTML.

En effet, le HTML sert à définir les différents éléments d'une page, et leur donner du sens. Le CSS, lui, va servir à mettre en forme les différents contenus définis par le HTML en leur appliquant des styles.

3.1.3 JavaScript :



JavaScript désigne un langage de développement informatique, il a été créé par Brendan Eich en 1995. Ce qui rend Javascript spécial, c'est que tous les navigateurs modernes le supportent. Ce qui signifie que l'exécution des tâches est opérée par le navigateur lui-même, sur l'ordinateur de l'utilisateur, sans devoir installer un compilateur.

3.1.4. ReactJS:



ReactJS est une bibliothèque JavaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage (Single-page Application), via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.

3.1.5. NextUI:



NextUI est une bibliothèque de composants d'interface utilisateur (UI Component Library), qui permet d'utiliser de composants magnifiques dans la réalisation d'applications.

Nous avons utilisé les composants NextUI dans toutes les parties de notre application.

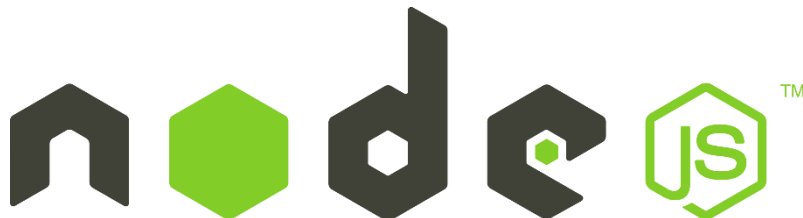
3.1.6. Recoil:



Recoil est une bibliothèque de gestion d'état (State Management Library) dans les applications React. Elle permet une gestion globale de l'état avec le concept d'Atomes et Sélecteurs.

3.2. Back-end :

3.2.1 NodeJS :



Node.js est un environnement d'exécution JavaScript open source, multiplateforme et back-end qui fonctionne sur le moteur V8 et exécute le code JavaScript en dehors d'un navigateur Web.

Node.js permet aux développeurs d'utiliser JavaScript pour écrire des outils en ligne de commande et pour l'exécution de scripts côté serveur afin de produire du

contenu de page Web dynamique avant que la page ne soit envoyée au navigateur Web de l'utilisateur.

Par conséquent, Node.js représente un paradigme de "JavaScript partout", unifiant le développement d'applications Web autour d'un seul langage de programmation, plutôt que de langages différents pour les scripts côté serveur et côté client.

3.2.2 Next.js :



Next.js est un cadre de développement web open-source construit au-dessus de Node.js et permettant aux applications web basées sur React d'offrir des fonctionnalités telles que le rendu côté serveur et la génération de sites web statiques.

Alors que les applications React traditionnelles ne peuvent rendre leur contenu que dans le navigateur côté client, Next.js étend cette fonctionnalité aux applications rendues côté serveur.

3.3. Base de données :

3.3.1. MySQL:



MySQL (My Structured Query Language), désigne un type de base de données relationnelle la plus utilisée dans le monde. MySQL est un lieu de stockage et d'enregistrement des données. Il est alors ensuite possible, via une requête SQL, d'exécuter des requêtes pour récupérer, sauvegarder, mettre à jour et supprimer des données à partir d'une base de données de façon très rapide.

Nous avons utilisé PlanetScale pour créer une base de données MySQL dans le nuage, ce qui nous permet de gérer et de contrôler facilement notre base de données, tout en la rendant accessible 24/7 et en tout lieu.

3.3.2. Prisma:



Prisma est un ORM (Object Relational Mapper) puissante et open source. Il fournit un client global pour la connexion et l'interrogation de la base de données. Nous avons choisi d'utiliser Prisma ORM pour écrire le schéma complet de notre base de données, ainsi que pour rédiger toutes nos requêtes.

3.3.3 PlanetScale:



Les bases de données PlanetScale sont conçues pour les développeurs et leurs flux de travail. PlanetScale est basé sur MySQL et utilise un principe innovant de Branches (main, dev ...).

3.3.4. Amazon AWS S3 :



Amazon Simple Storage Service (Amazon S3) est un service de stockage en nuage basé sur le Web, évolutif et à haut débit. Ce service est élaboré pour la sauvegarde

et l'archivage en ligne de données et d'applications sur Amazon Web Services (AWS). Amazon S3 a été créé pour faciliter l'informatique à l'échelle du Web pour les développeurs.

Nous avons utilisé S3 pour héberger les images de notre application, ceci est expliqué avec plus de détails dans la section d'optimisations.

3.4. Client AJAX : React Query



React Query est une bibliothèque de récupération de données pour React, qui facilite la récupération, la mise en cache, la synchronisation et la mise à jour de l'état du serveur dans les applications React.

3.5. Authentification : Auth0



Auth0 est une solution sécurisée et flexible pour ajouter des services d'authentification et d'autorisation aux applications.

C'est le leader du secteur de l'authentification. Il fournit une vaste palette d'outils pour faciliter l'accès aux plus hauts niveaux de sécurité et de cryptage dans des applications de qualité professionnelle. Ainsi que l'intégration facile avec les fournisseurs OAuth (comme Google, Facebook, Twitter, etc.).

Au lieu de stocker des informations d'identification importantes et cruciales sur notre base de données PlanetScale, nous avons choisi d'utiliser Auth0 car il nous offre une flexibilité et une sécurité exceptionnelles en matière d'authentification des utilisateurs.

3.6. Hébergement : Heroku



Heroku est une plateforme en cloud supportant plusieurs langages de programmation. Il est en développement depuis juin 2007. Et il est compatible avec PHP, Java, Ruby, Node.js, Python et Go.

Heroku est un excellent fournisseur d'hébergement pour tout type d'application qui doit exister dans le cloud, et pour cela, nous l'avons choisi comme service d'hébergement pour notre application.

3.7. CDN : Cloudflare



Cloudflare est une suite intégrée de sécurité et de performance pour les applications basées sur le Web. Cloudflare propose des solutions sophistiquées basées sur des services de sécurité et d'amélioration des performances, comme la protection contre les attaques de déni de service (DDoS), le routage automatique HTTPS, un certificat SSL gratuit, le cryptage du trafic, et l'optimisation des fichiers Javascript et CSS.

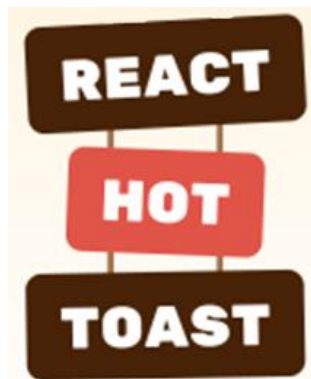
4. Autres outils :

4.1. Freenom:



Freenom est un service d'enregistrement de noms de domaine. Nous avons utilisé ce service pour obtenir le domaine www.skyline-app.ga pour notre application web.

4.2. React Hot Toast:



React Hot Toast est une bibliothèque légère pour ajouter des notifications à une application React. Nous avons utilisé React Hot Toast pour rendre notre application interactive, où les actions ont des réponses visuelles.

5. L'architecture de l'application :

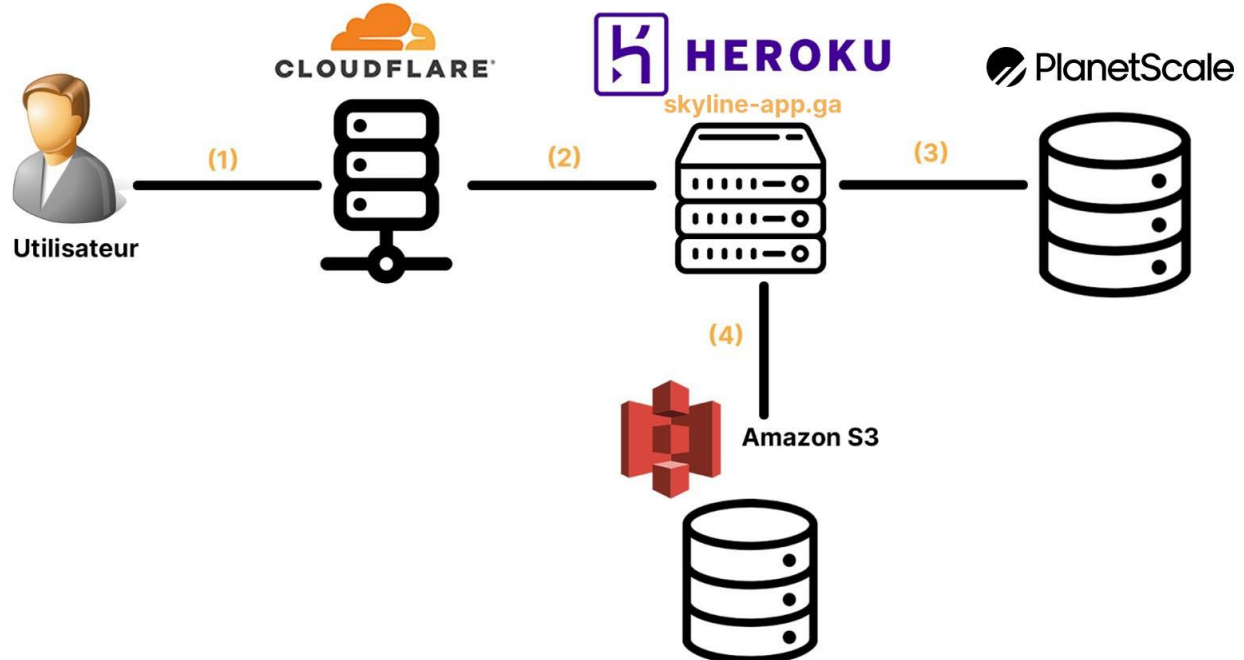


Figure 8: L'architecture de l'application

(1) Cryptage de trafic :

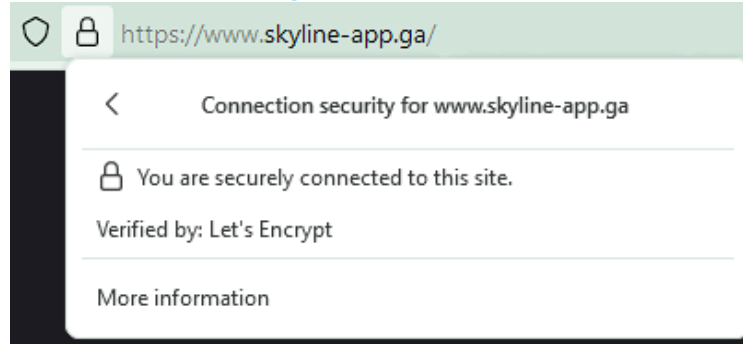


Figure 9: Connection sécurisée avec notre application

Dans cette étape, le client et Cloudflare établissent une poignée de main TLS (TLS Handshake) afin de pouvoir communiquer en toute sécurité.

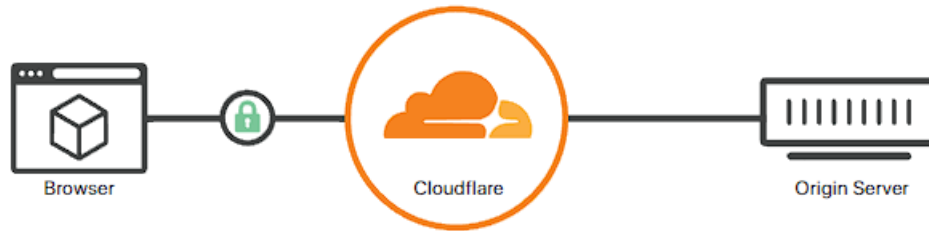


Figure 10: Cryptage de trafic depuis et vers Cloudflare

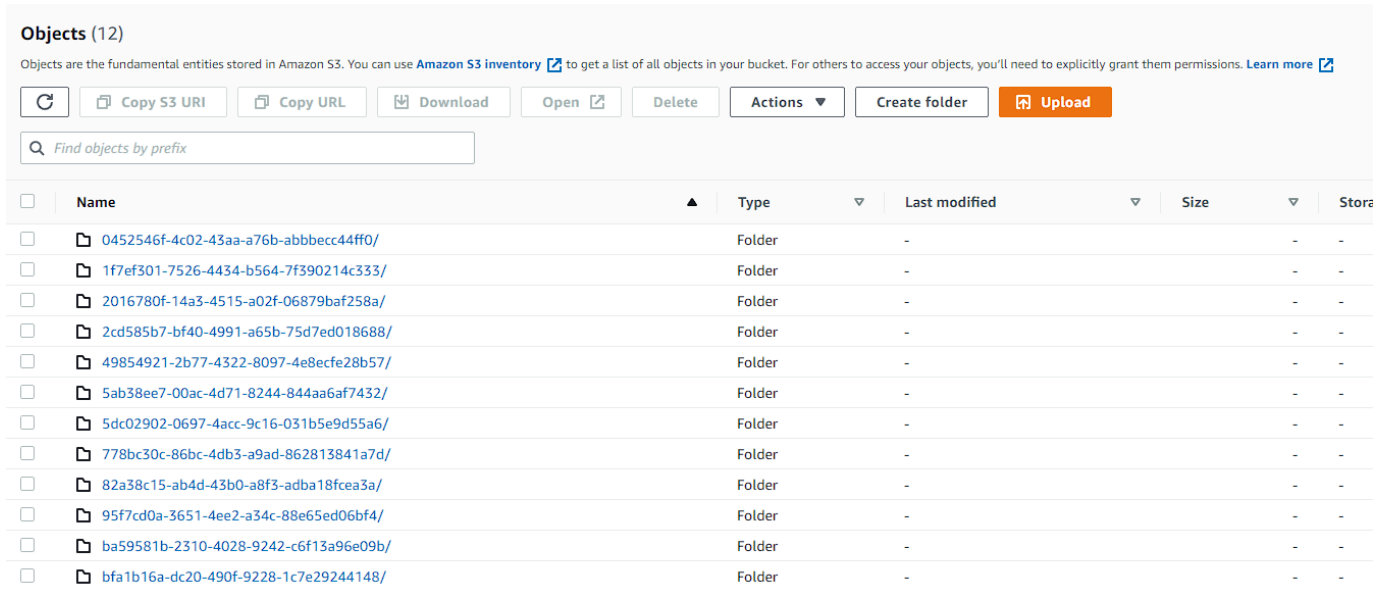
(2) Établir une connexion à Heroku :

Dans cette étape, Cloudflare agit comme un intermédiaire entre le client et les serveurs Heroku, il trouve la route la plus courte entre le client et ses serveurs et utilise cette route pour tout le trafic prochain. Il accélère également le temps de chargement des pages en compressant les fichiers Javascript et CSS.

(3) Communication avec la base de données :

Notre application communique avec la base de données hébergée dans PlanetScale pour gérer l'authentification des utilisateurs, l'insertion, la mise à jour et la suppression des données.

(4) Récupération et stockage des photos :



<input type="checkbox"/>	Name	Type	Last modified	Size	Storage
<input type="checkbox"/>	0452546f-4c02-43aa-a76b-abbbecc44ff0/	Folder	-	-	-
<input type="checkbox"/>	1f7ef301-7526-4434-b564-7f390214c333/	Folder	-	-	-
<input type="checkbox"/>	2016780f-14a3-4515-a02f-06879baf258a/	Folder	-	-	-
<input type="checkbox"/>	2cd585b7-bf40-4991-a65b-75d7ed018688/	Folder	-	-	-
<input type="checkbox"/>	49854921-2b77-4322-8097-4e8ecfe28b57/	Folder	-	-	-
<input type="checkbox"/>	5ab38ee7-00ac-4d71-8244-844aa6af7432/	Folder	-	-	-
<input type="checkbox"/>	5dc02902-0697-4acc-9c16-031b5e9d55a6/	Folder	-	-	-
<input type="checkbox"/>	778bc30c-86bc-4db3-a9ad-862813841a7d/	Folder	-	-	-
<input type="checkbox"/>	82a38c15-ab4d-43b0-a8f3-adba18fcea3a/	Folder	-	-	-
<input type="checkbox"/>	95f7cd0a-3651-4ee2-a34c-88e65ed06bf4/	Folder	-	-	-
<input type="checkbox"/>	ba59581b-2310-4028-9242-c6f13a96e09b/	Folder	-	-	-
<input type="checkbox"/>	bfa1b16a-dc20-490f-9228-1c7e29244148/	Folder	-	-	-

Figure 11: Notre tableau de bord de AWS S3 contenant les images des utilisateurs

Après avoir vérifié les privilèges de l'utilisateur authentifié, notre application s'authentifie avec notre stockage S3, puis demande des images spécifiques (selon les besoins du page).

6. Optimisations :

6.1. Optimisation de sécurité :

Comme nous l'avons mentionné dans la section précédente, tout le trafic entrant et sortant de l'utilisateur vers Cloudflare est crypté et sécurisé.

De plus, la récupération des images nécessite un identifiant de 36 chiffres et caractères, cet identifiant ne peut être généré que par notre application.

Et enfin, toute l'authentification est effectuée par Auth0, qui fournit une sécurité et un cryptage de très haut niveau pour traiter en toute sécurité des détails confidentiels comme les mots de passe des utilisateurs.

Le choix d'Auth0 garantit totalement la sécurité de l'utilisateur car, ce faisant, les mots de passe ne sont plus stockés dans notre base de données, ce qui signifie qu'en cas de piratage de la base de données, les hackers n'auront jamais accès aux mots de passe hachés.

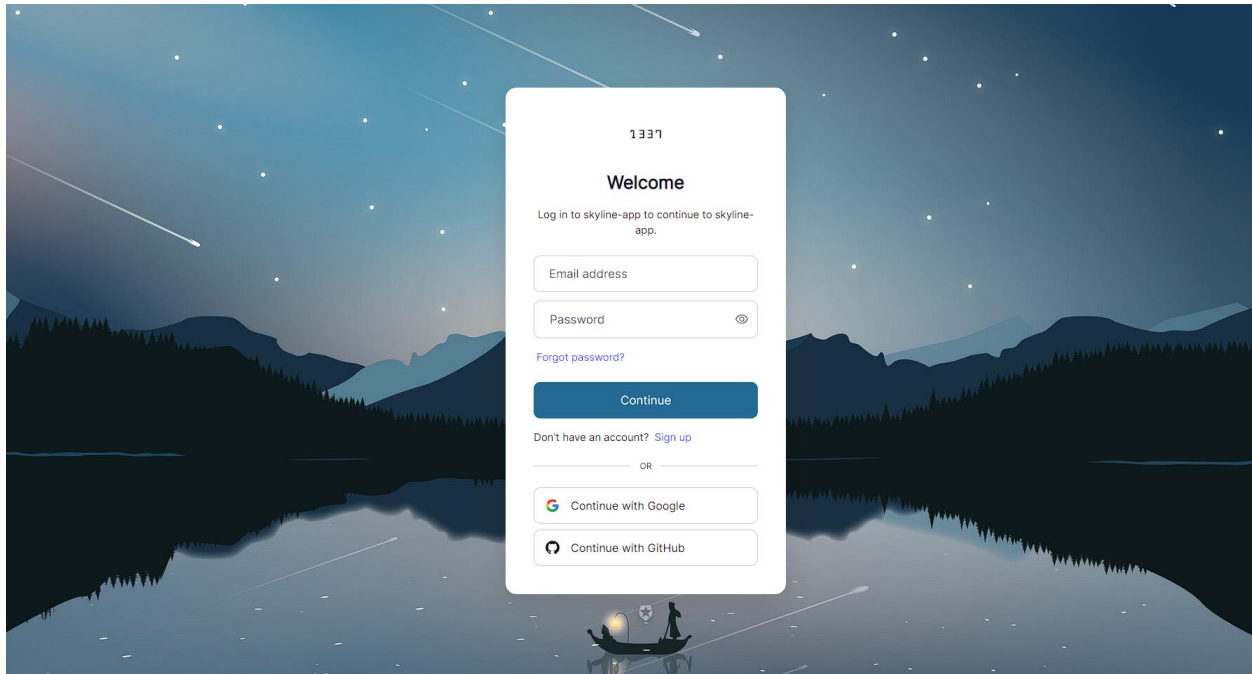


Figure 12: Notre page d'authentification utilisant Auth0

6.2. Optimisation de performance :

La principale raison pour laquelle nous avons opté pour Next.js est qu'il s'agit du meilleur framework de JavaScript en termes de performances.

Next.js propose plusieurs types de rendu de page, chaque choix étant utilisé dans un but précis :

Rendement côté serveur (SSR) :

Avec le rendu côté serveur, le HTML de la page est généré sur un serveur pour chaque requête. Les instructions JavaScript pour rendre la page interactive sont ensuite envoyés au client.

Génération de sites statiques (SSG) :

Avec la génération de sites statiques, le HTML est généré sur le serveur, mais contrairement au rendu côté serveur, il n'y a pas de serveur au moment de l'exécution. Au lieu de cela, le contenu est généré une fois, au moment de la construction, lorsque l'application est déployée, et le HTML est stocké dans un CDN et réutilisé pour chaque requête.

Rendement côté client (CSR) :

Générer statiquement (pré-rendu) les parties de la page qui ne nécessitent pas de données externes.

Lorsque la page se charge, récupérer les données externes du client à l'aide de JavaScript et remplir les parties restantes.

Notre application est une application de type "dashboard", ce qui signifie que tout le contenu de l'application se trouve derrière une page de login et qu'il n'est pas nécessaire des optimisations des moteurs de recherche (SEO). C'est pourquoi nous avons choisi le CSR (Client-Side Rendering) car c'est le type de rendu le plus approprié pour notre application.

Avec CSR, tout le HTML, le CSS et le JavaScript basique de la page sont générés complètement lors du déploiement (appelé Build-Time), et tout ceci est mis en cache dans le CDN Cloudflare (discuté dans 3.7.). Ce qui rend la diffusion du contenu incroyablement rapide, car avec le CDN, il n'y a pas besoin de contacter le serveur Heroku à chaque fois pour obtenir le HTML. Ceci réduit considérablement les coûts de serveur.

Après que le client ait chargé le HTML et le CSS pré-générés (appelé Run time), JavaScript prend le relais et charge tout le contenu spécifique à l'utilisateur à partir de la base de données en utilisant des requêtes AJAX, qui envoie une requête et affiche dynamiquement les résultats sans avoir besoin de recharger toute la page.

De plus, notre application est une SPA (Single Page Application), ce qui signifie que le fait de passer d'une page à l'autre ne rafraîchit pas complètement le site Web. Au lieu de cela, elle masque dynamiquement le contenu de la page actuelle et affiche l'autre tout en conservant les éléments persistants tels que la barre de navigation affichés à tout moment.

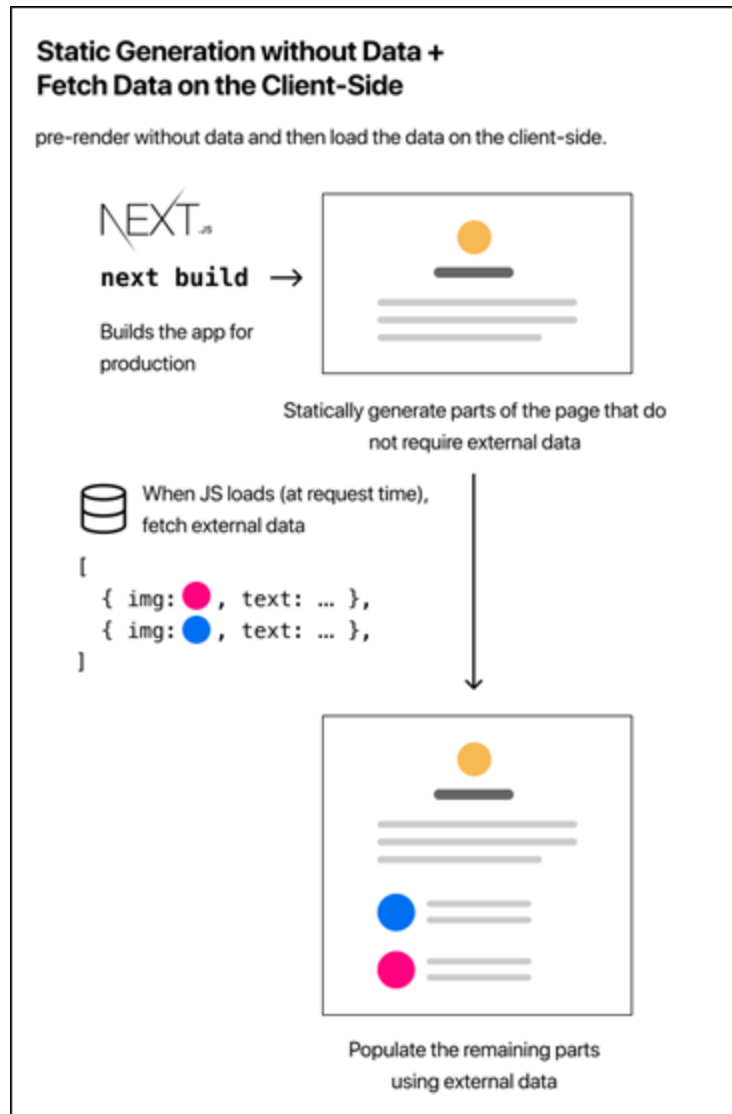


Figure 13: Diagramme expliquant l'architecture de rendement côté client (CSR).

6.3. Optimisation de déploiement :

Nous avons configuré notre environnement de déploiement de telle sorte que dès que nous poussons un commit vers notre répertoire GitHub, Heroku détecte immédiatement que le répertoire a été mis à jour et télécharge automatiquement tous les fichiers modifiés, puis, il redéploie le répertoire vers notre instance. En cas d'échec du déploiement, Heroku nous envoie un courriel pour nous en informer et nous fournit les journaux de compilation complets.

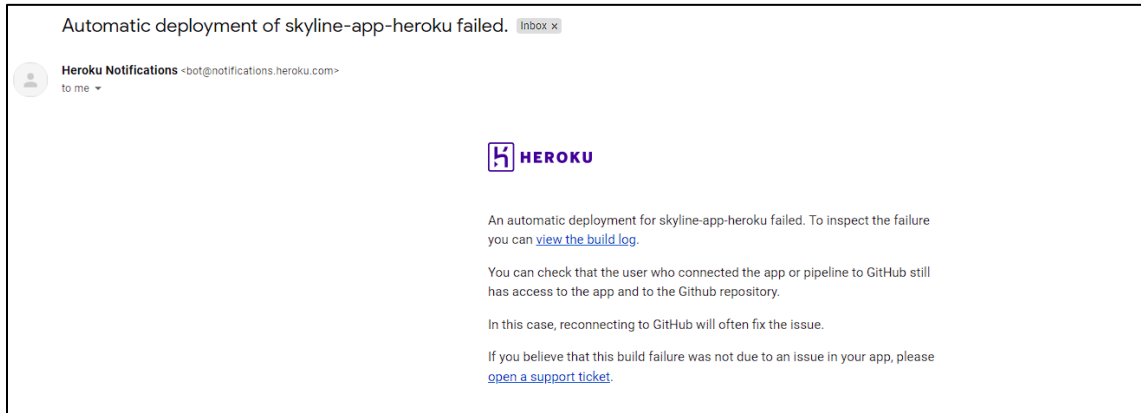


Figure 14: Exemple d'email nous informant de l'échec d'un déploiement automatique

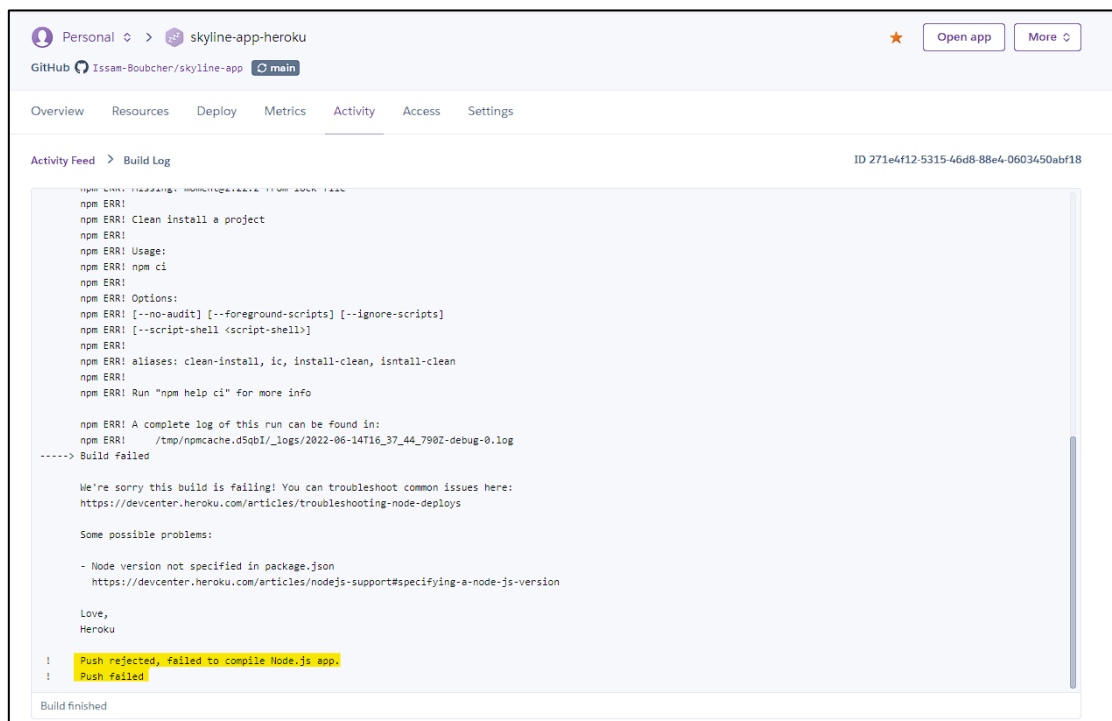


Figure 15: Exemple d'un journal de compilation qui a échoué

Disposer de cette chaîne de déploiement est extrêmement précieux, car cela nous permet de déployer les mises à jour et les correctifs critiques de manière simple et beaucoup plus rapide qu'un déploiement manuel, et nous alerte également par email en cas d'échec.

7. Présentation de l'application :

7.1. Introduction :

Lorsqu'une nouvelle personne s'inscrit dans notre application, elle est considérée comme un **Utilisateur**, au départ, cet utilisateur peut soit créer un tout nouvel immeuble, soit accepter une invitation à un immeuble (s'il a été invité).

Lorsque l'utilisateur crée un immeuble, il devient le **Créateur Syndic** de l'immeuble, le créateur est le **Résident** avec le plus haut niveau de privilèges sur un immeuble, le créateur doit inviter d'autres résidents à l'immeuble, le créateur peut également choisir si le résident invité serait un **Administrateur Syndic**.

Le créateur ne doit accorder le rôle d'administrateur qu'aux résidents en qui il a confiance. En effet, le rôle d'administrateur permet de lancer des invitations sans avoir besoin de les valider, alors que les invitations envoyées par les résidents normaux doivent être validées par un résident ayant des privilèges d'administrateur ou de créateur. Nous reviendrons plus tard en détail sur le concept des rôles et invitations.




	 Créateur Syndic	 Administrateur Syndic	 Résident
Visualiser infos résidents			
Inviter résidents			
Inviter résidents comme adminstrateurs			
Valider Invitations			
Ajout cotisation			
Ajout dépense			
Retirer privilège administrateur			
Retirer résidents			

Figure 16: Privilèges des différents rôles des résidents

7.2. Plan de navigation de l'application :

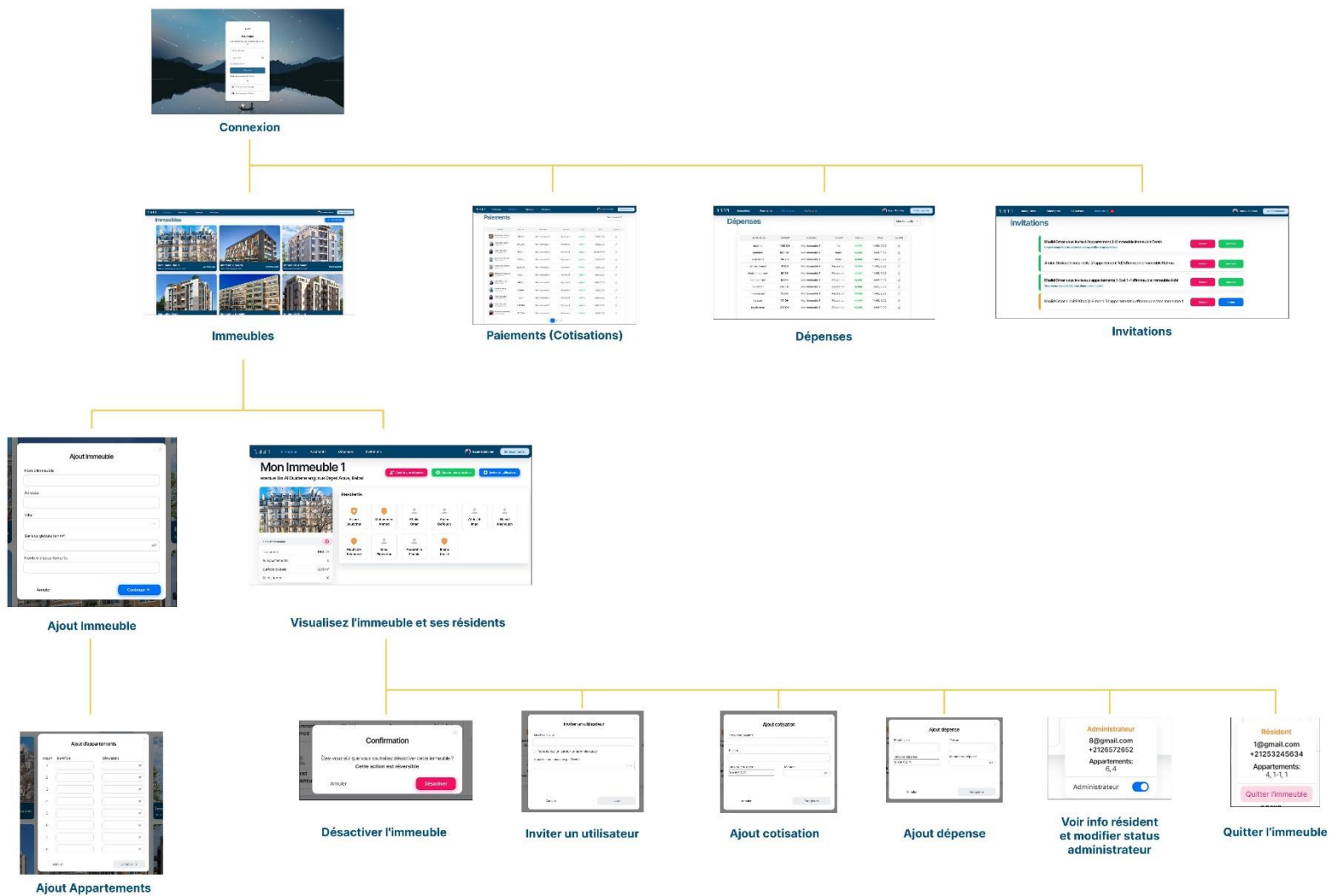


Figure 17: Arborescence de l'application

7.3. Page Immeubles :

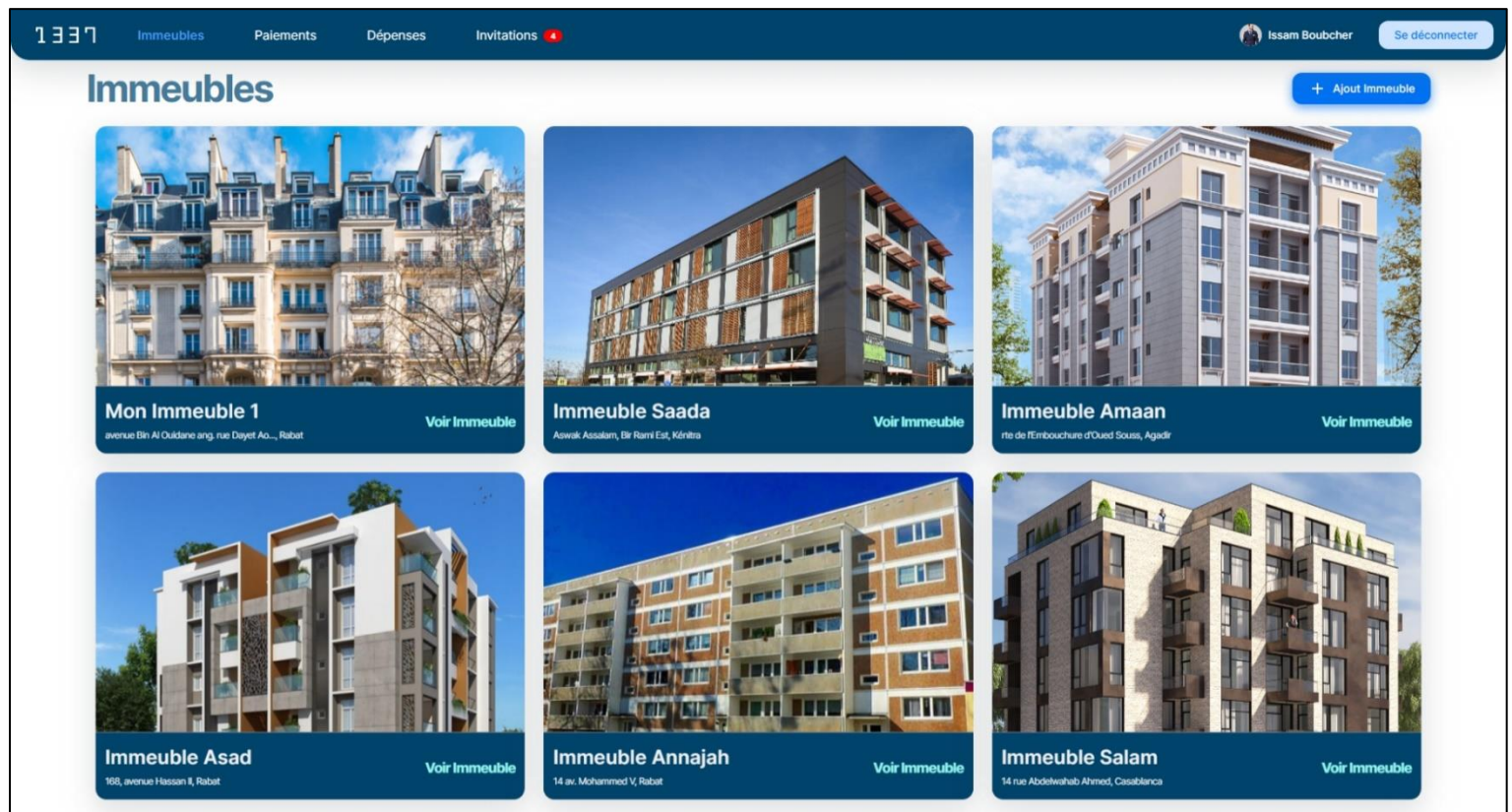
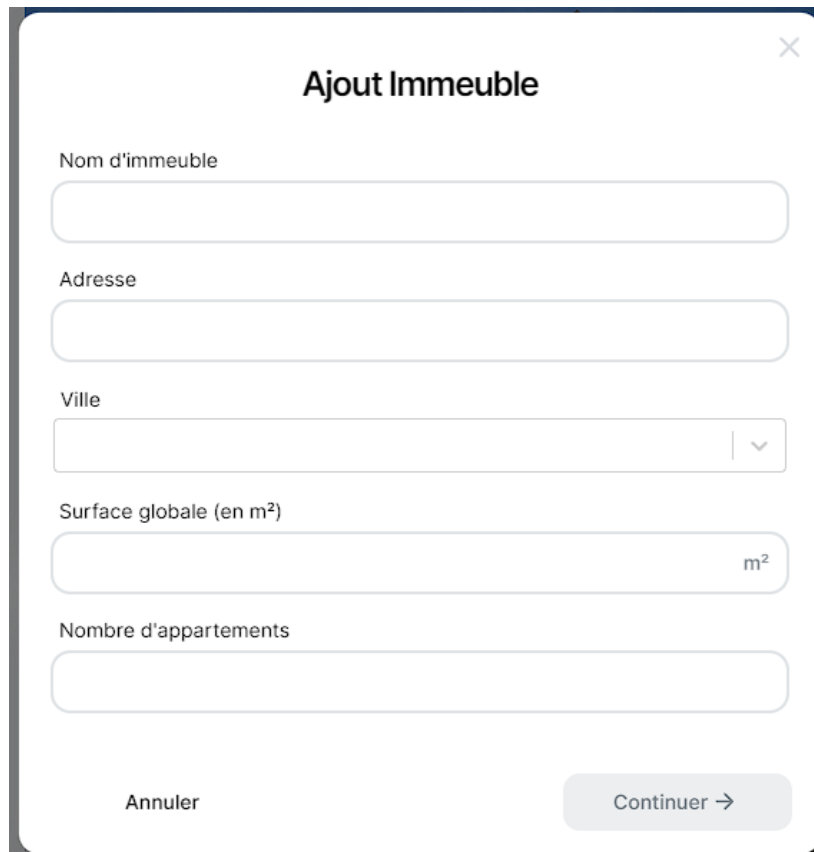


Figure 18: Page Immeubles

Il s'agit de la page des immeubles, où l'utilisateur peut visualiser tous les immeubles dans lesquels il est résident. L'utilisateur peut y voir l'image de l'immeuble, son nom, son adresse et sa ville.

Dans cette page, l'utilisateur peut également créer un immeuble :



A screenshot of a mobile application window titled "Ajout Immeuble". The window has a close button (X) in the top right corner. It contains five input fields: "Nom d'immeuble", "Adresse", "Ville" (with a dropdown arrow), "Surface globale (en m²)" (with a "m²" unit label), and "Nombre d'appartements". At the bottom, there are two buttons: "Annuler" and "Continuer →".

Figure 19: Fenêtre d'ajout immeuble

En ajoutant un nouvel immeuble, l'utilisateur devient le **Créateur Syndic** de l'immeuble.



A screenshot of a mobile application window titled "Ajout d'appartements". The window has a close button (X) in the top right corner. It contains a table with four rows for adding apartments. The table has three columns: "Appart.", "Identifiant", and "Dimensions". Each row has input fields for the "Identifiant" and "Dimensions" (with a "m²" unit label). At the bottom, there are two buttons: "Annuler" and "Continuer →".

Appart.	Identifiant	Dimensions
1	<input type="text"/>	<input type="text"/> m²
2	<input type="text"/>	<input type="text"/> m²
3	<input type="text"/>	<input type="text"/> m²
4	<input type="text"/>	<input type="text"/> m²

Figure 20: Fenêtre d'ajout d'appartements d'immeuble

L'utilisateur peut spécifier l'identifiant de l'appartement selon le système d'identification qu'il utilise, par exemple, un immeuble peut avoir ses appartements simplement identifiés **par un numéro** (1,2,3 ...) ou **par étage et numéro** (1-1, 1-2, 2-1, 2-2, 3-1 ...) ou **par étage, numéro et direction** (1g-4, 2d-6 ... ici "g" se réfère à gauche et "d" se réfère à droite, donc 2d-6 signifie : "étage 2, droite, appartement 6").

À partir de cela, nous pouvons clairement voir qu'il y a une multitude de façons d'identifier l'emplacement d'un appartement spécifique.

C'est pourquoi nous avons décidé d'utiliser un système d'identification caché pour l'architecture back-end, tout en ne révélant à l'utilisateur que le système d'identification utilisé dans son immeuble exact.

Après avoir rempli les informations sur l'appartement, le créateur doit choisir les appartements qu'il possède :



Figure 21: Sélection d'appartements de Créateur

Nous aimerions ici souligner l'outil que nous utilisons pour la fonctionnalité de sélection multiple, il s'appelle React-Select et il est puissant et hautement personnalisable.



En ce qui concerne **l'attribution des appartements**, 2 règles ont été mises en place, ces règles régissent toute la logique d'attribution des appartements aux résidents, ces règles sont les suivantes :

- Un résident peut posséder plus d'un appartement. Un appartement peut appartenir à plus d'un seul résident (relation **n-n**).
- Aucun utilisateur ne peut être résident d'un immeuble dans lequel il n'est attribué à aucun appartement.

Ces deux règles ont permis de solidifier la réalisation de la logique back-end et de s'assurer qu'il n'y aura jamais de cas particulier susceptible de provoquer des bogues plus tard.

Lorsque l'utilisateur clique sur un immeuble, il est dirigé vers la page de détails de cet immeuble :

The screenshot displays the 'Mon Immeuble 1' details page. The header bar contains navigation links: 'Immeubles', 'Palements', 'Dépenses', and 'Invitations' (with a red notification badge). The user 'Issam Boubcher' is logged in, with a 'Se déconnecter' button. The main title is 'Mon Immeuble 1' with the address 'avenue Bin Al Ouidane ang. rue Dayet Aoua, Rabat'. Below the title are three action buttons: 'Ajouter une dépense', 'Ajouter une cotisation', and 'Inviter un utilisateur'. A large image of the building is shown on the left. Below it is a 'Fiche d'information' section with the following data:

Fiche d'information	
Trésorerie:	884 DH
N. appartements:	8
Surface globale:	1200 m²
N. résidents:	10

To the right of the image is a 'Residents' section displaying 10 user avatars and names in a grid:

- Issam Boubcher
- Mohammed Ahmed
- Khalid Omar
- Amine Deriouch
- Abdellah Imad
- Hamid Machouch
- Mouhcine Belahmed
- Imad Elmokhtar
- Abderahim Elnahiri
- Imane Khalid

Figure 22: Page des détails d'immeuble

Ici, l'utilisateur peut voir toutes les informations concernant un certain immeuble ainsi que plus d'informations sur chaque résident de l'immeuble, tous les résidents ont une icône au-dessus de leur nom indiquant leur rôle :



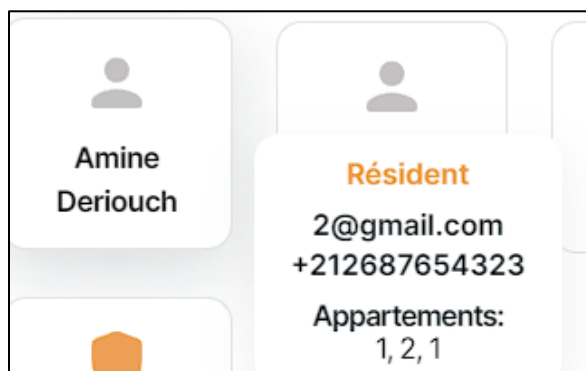
Figure 23: Les différents rôles dans un immeuble

Tous les créateurs sont des administrateurs et des résidents.

Tous les administrateurs sont des résidents.

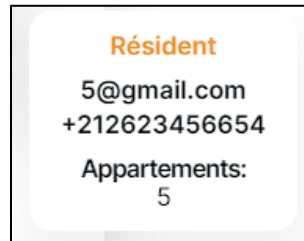
Le contraire n'est pas toujours vrai.

Lorsque la carte d'un résident est cliquée, cela montre plus d'informations sur ce résident spécifique, comme son rôle (Créateur/Administrateur/Résident), son adresse e-mail, son numéro de téléphone, et les appartements qu'il possède dans cet immeuble (notée que les identifiants d'appartement qui sont montrés ici sont ceux fournis par le créateur de l'immeuble au moment de création d'immeuble, pas l'identifiant caché de l'appartement. Cela garantit la clarté et la cohérence de la visualisation des informations, puisque la représentation numérique de l'appartement correspond toujours à l'identité de l'appartement réel) :



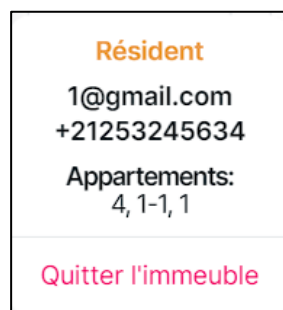
Cette fiche est dynamique, car son contenu change en fonction de la personne qui l'a cliquée, et de qui elle concerne. Voici toutes les variations de la carte, ainsi que les circonstances spécifiques qui ont conduit à cette variation exacte :

- **Résident clic sur un autre résident :**



Seulement les informations de ce résident sont visibles, aucune action ici n'est possible.

- **Résident clic sur lui-même :**



Le résident peut voir ses informations, et il a la possibilité de quitter l'immeuble.

- **Administrateur clic sur un résident :**



L'administrateur peut retirer un résident de l'immeuble.

- **Créateur clic sur un administrateur/résident :**

Administrateur
test@gmail.com
+21262345676
Appartements:
3

Administrateur
☒

Retirer cet utilisateur de l'immeuble

Le créateur peut ajouter ou retirer le rôle d'administrateur à un utilisateur après qu'il ait rejoint l'immeuble. Le créateur peut également retirer le résident de l'immeuble.

Comme vous pouvez le constater, différentes actions apparaissent à l'utilisateur, en fonction de son rôle dans l'immeuble ainsi que du contexte de l'action.

Cette idée s'étend à de nombreux autres aspects de l'application, nous allons en aborder certains dès maintenant, tandis que nous en aborderons d'autres plus tard. A titre d'exemple, seuls les administrateurs et les créateurs peuvent ajouter des dépenses ou des cotisations :



Figure 24: Actions que les créateurs et les administrateurs peuvent effectuer sur un immeuble.



Figure 25: Actions que les résidents peuvent effectuer sur un immeuble.

Comme vous le remarquez, les fonctions "Ajouter une cotisation" et "Ajouter une dépense" sont **cachées** aux **résidents normaux**.

Nous parlerons plus tard des paiements, des dépenses et de la trésorerie de l'immeuble.

Passons à un autre sujet : **Système d'Invitations** :

Tous les résidents d'un immeuble peuvent inviter d'autres résidents à rejoindre l'immeuble. Nous avons décidé d'accorder à tous les résidents d'une immeuble la possibilité d'effectuer cette action car elle permet une intégration plus rapide des nouveaux résidents dans l'immeuble, puisqu'il n'est pas nécessaire d'attendre qu'un administrateur ou le créateur de l'immeuble envoie lui-même toutes les invitations. Les invitations sont de 2 types, celles effectuées par un administrateur/créateur, et celles effectuées par un résident :

- **Invitation effectuée par un administrateur/créateur** : Ce type d'invitation est direct, une fois l'invitation est envoyée, le système attend la réponse du destinataire, qui peut soit accepter soit refuser l'invitation. S'il choisit de l'accepter, le destinataire devient un résident de cet immeuble, il sera propriétaire d'un appartement ou d'un ensemble d'appartements qui sont spécifiés lors de l'envoi de l'invitation. Il faut noter qu'avec ce type d'invitation, il est possible de faire en sorte que le destinataire devienne instantanément un administrateur lorsqu'il accepte l'invitation, le destinataire en sera informé au préalable dans l'invitation elle-même.

Inviter un utilisateur

Email de l'invité

destinataire@gmail.com

☐ Rendez cet utilisateur un administrateur

Appartements occupés par l'invité

3 x 4 x

Annuler Inviter

Figure 26 : Formulaire d'invitation d'utilisateurs à l'immeuble (créateur/administrateur)

- **Invitation effectuée par un résident** : Tous les résidents ont la possibilité d'inviter d'autres personnes dans l'immeuble, mais après que le destinataire ait accepté l'invitation, celle-ci doit être validée manuellement par un autre résident du même immeuble ayant le rôle d'administrateur ou de créateur. Cette façon de gérer les invitations garantit un processus d'invitation rapide et libre, tout en gardant la sécurité et la sûreté d'avoir l'entrée fermée derrière la validation d'un l'utilisateur élevé. Un exemple de ceci sera fourni ultérieurement. L'expéditeur et le destinataire sont tous deux informés de cette restriction. Il faut noter qu'avec ce type d'invitation, il n'est pas possible de mettre à la disposition du
- destinataire le rôle d'administrateur, car cela conduirait les résidents normaux à pouvoir donner à d'autres des privilèges qu'ils n'ont pas eux-mêmes.

The modal form is titled "Inviter un utilisateur" and includes a close button in the top right corner. It contains the following elements:

- A label "Email de l'invité" above a text input field containing "destinataire@gmail.com".
- An orange warning message: "ⓘ Un administrateur de cet immeuble va vérifier cette invitation."
- A label "Appartements occupés par l'invité" above a multi-select dropdown menu.
- The dropdown menu shows two selected items: "1 x" and "3 x", with a clear button (x) and a dropdown arrow (v) on the right.
- At the bottom, there are two buttons: "Annuler" (grey) and "Inviter" (blue).

Figure 27 : Formulaire d'invitation d'utilisateurs à l'immeuble (résident)

Enfin, l'immeuble peut être désactivé, ce qui retire tous ses résidents et administrateurs, laissant le créateur de l'immeuble comme seul résident.

The modal form is titled "Confirmation" and includes a close button in the top right corner. It contains the following elements:

- The question: "Êtes-vous sûr que vous souhaitez désactiver cette immeuble ?"
- The note: "Cette action est réversible"
- At the bottom, there are two buttons: "Annuler" (grey) and "Désactiver" (pink).

Figure 28 : Confirmation de la désactivation de l'immeuble

Les immeubles ne peuvent jamais être supprimés définitivement, ils peuvent seulement être désactivés (et aussi réactivés).

C'est intentionnel, car la suppression d'un immeuble implique la **suppression de l'historique complet de tous les paiements et dépenses** effectués dans cet immeuble, ce qui ne devrait **jamais** se produire.

7.4. Page paiements (cotisations) :

Le devoir principal du syndic de l'immeuble est de collecter une contribution mensuelle de tous les résidents, et de dépenser cet argent pour les réparations importantes, ainsi que les frais récurrents.

La tenue d'un registre public de toutes les contributions, de la somme d'argent versée, de la date et de l'auteur de la contribution est vraiment importante pour assurer une **transparence complète et totale**, ce qui est l'objectif principal de notre application.

Ici, les résidents ayant le rôle d'administrateur ou de créateur peuvent ajouter un paiement. Ils doivent choisir quel résident paie, la raison de ce paiement, la date à laquelle ce paiement a eu lieu, ainsi que son montant.

The image shows a mobile application form titled "Ajout cotisation" with a close button (X) in the top right corner. The form contains several input fields: a dropdown menu for "Personne payante" with "Issam Boubcher" selected; a text field for "Raison" containing "Contribution mensuelle"; a date field for "Date de cotisation" with "15/06/2022"; and a numeric field for "Montant" with "120" and a currency selector set to "DH". At the bottom, there are two buttons: "Annuler" and "Enregistrer".

Figure 29 : Formulaire d'ajout d'une cotisation (paiement)

La page des paiements met en évidence tous les paiements qui ont eu lieu dans les immeubles dont l'utilisateur est résident.

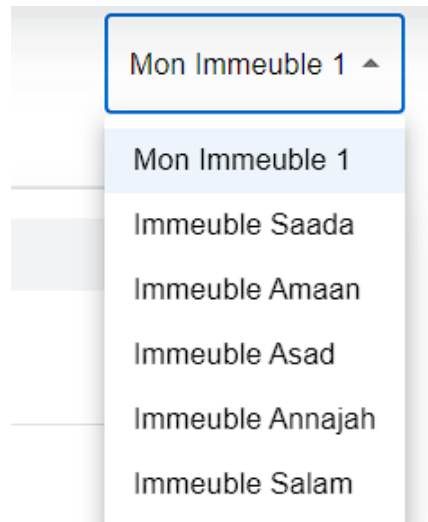


Figure 30 : Menu de sélection des immeubles dans lesquels l'utilisateur est résident

Toutes les informations de cette page sont classées en fonction des immeubles dont elles font partie.

Si l'utilisateur fait partie de plusieurs immeubles, il peut visualiser les paiements qui ont eu lieu sur chacun des immeubles en sélectionnant dans le menu ci-dessus.

1337

Immeubles

Palements

Dépenses

Invitations

Issam Boucher

Se déconnecter

Palements

Mon Immeuble 1

RÉSIDENT	MONTANT	IMMEUBLE	RAISON	STATUS	DATE
<div><div></div><div>Mohammed Ahmed</div><div>test3@gmail.com</div></div>	120 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022
<div><div></div><div>Hamid Machouch</div><div>5@gmail.com</div></div>	540 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022
<div><div></div><div>Issam Boucher</div><div>test4@gmail.com</div></div>	760 DH	Mon Immeuble 1	Payement	SUCCES	06/06/2022
<div><div></div><div>Abderahim Elnahiri</div><div>7@gmail.com</div></div>	100 DH	Mon Immeuble 1	Payement	SUCCES	16/05/2022
<div><div></div><div>Abderahim Elnahiri</div><div>7@gmail.com</div></div>	2200 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022
<div><div></div><div>Mouhcine Belahmed</div><div>test@gmail.com</div></div>	770 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022
<div><div></div><div>Hamid Machouch</div><div>5@gmail.com</div></div>	120 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022
<div><div></div><div>Abdellah Imad</div><div>3@gmail.com</div></div>	80 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022
<div><div></div><div>Issam Boucher</div><div>test4@gmail.com</div></div>	15 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022
<div><div></div><div>Issam Boucher</div><div>test4@gmail.com</div></div>	1141 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022
<div><div></div><div>Mouhcine Belahmed</div><div>test@gmail.com</div></div>	430 DH	Mon Immeuble 1	Payement	SUCCES	14/06/2022

1

2

>

Figure 31: Page paiements

Cette page détaille toutes les informations concernant chaque paiement qui a eu lieu dans un immeuble. Il est primordial que cette page (et la page de dépenses)

soit **accessible à tous les résidents** d'un immeuble, **pas seulement** à ses administrateurs et à son créateur.

L'établissement d'une liste de toutes les contributions, de leur montant, de la date à laquelle elles ont été versées et de leur auteur est la clé d'une **transparence totale**, mais comment pouvons-nous élever ce concept ?

Avec **La Trésorerie de l'Immeuble**. C'est une représentation digitale du montant exact des fonds dont disposent les administrateurs de l'immeuble à ce moment précis. La trésorerie s'incrémente automatiquement lorsqu'une nouvelle contribution est notée et diminue automatiquement lorsqu'une dépense est notée. Ce concept ajoute une nouvelle couche de transparence. En effet, notre application ne rend pas seulement responsables ceux qui ne paient pas leur contribution. Mais elle rend également responsables les administrateurs de l'immeuble, car le montant des fonds de l'immeuble qu'ils détiennent à un moment donné doit toujours être exactement équivalent au montant mentionné dans l'application.

Cela décourage fortement tout administrateur d'immeuble de détourner des fonds, et augmente la confiance des résidents dans le fait que leur contribution ne sera dépensée qu'au profit de l'immeuble.

7.5. Page dépenses :

Les administrateurs de l'immeuble peuvent dépenser les fonds obtenus pour les dépenses liées à l'immeuble, telles que les travaux de maintenance et de réparation.

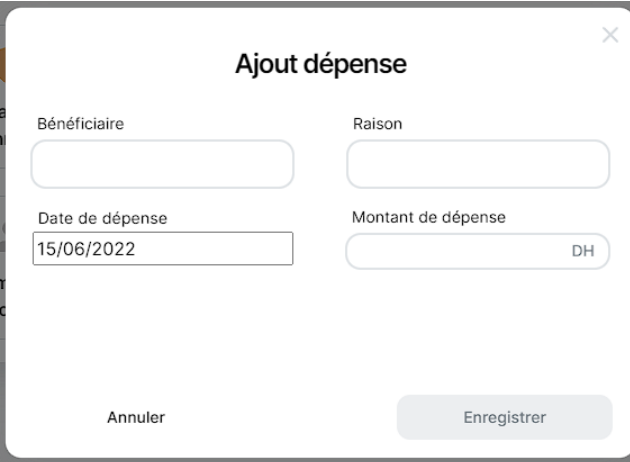
Le formulaire est intitulé "Ajout dépense" et se présente sous la forme d'une fenêtre modale avec un bouton de fermeture en haut à droite. Il est divisé en quatre sections de saisie : "Bénéficiaire" (un champ de texte vide), "Raison" (un champ de texte vide), "Date de dépense" (un champ de date contenant "15/06/2022") et "Montant de dépense" (un champ de texte avec "DH" à droite). En bas à gauche, il y a un bouton "Annuler", et en bas à droite, un bouton "Enregistrer".

Figure 32: Formulaire d'ajout d'une dépense

Voici la page des dépenses, où tous les résidents peuvent voir toutes les dépenses que les administrateurs de l'immeuble ont notées, avec des détails tels que le bénéficiaire de cette dépense, la raison de cette dépense et son coût.

BÉNÉFICIAIRE	MONTANT	IMMEUBLE	RAISON	STATUS	DATE
tech inc	1400 DH	Mon Immeuble 1	Fix	succès	14/06/2022
plombier	600 DH	Mon Immeuble 1	Repair	succès	14/06/2022
assistance	760 DH	Mon Immeuble 1	Repair	succès	10/06/2022
ahmed hamid	70 DH	Mon Immeuble 1	Réparation	succès	14/06/2022
khalid mouhcine	60 DH	Mon Immeuble 1	Réparation	succès	14/06/2022
bremade inc	54 DH	Mon Immeuble 1	Réparation	succès	14/06/2022
fix center	230 DH	Mon Immeuble 1	Réparation	succès	14/06/2022
fix company	33 DH	Mon Immeuble 1	Réparation	succès	14/06/2022
fix corp	111 DH	Mon Immeuble 1	Réparation	succès	14/06/2022
repair omar	222 DH	Mon Immeuble 1	Réparation	succès	14/06/2022

Figure 33: Page dépenses

7.6. Page Invitations :

Nous avons déjà discuté en détail le système d'invitation et les 2 types d'invitations (voir 7.3. de ce chapitre).

La page des invitations est la page où l'utilisateur peut voir tous les immeubles auxquels il a été invité, ainsi que l'identité de son expéditeur et la ou les appartement(s) dont il sera propriétaire (notez que l'utilisateur n'est pas invité à l'immeuble lui-même, mais à un appartement spécifique ou à un ensemble d'appartements dont il sera propriétaire, ceci en raison de la règle selon laquelle aucun utilisateur ne peut être résident d'un immeuble sans être attribué à un appartement de cet immeuble).

Lorsque l'utilisateur a une invitation qui l'attend, nous avons fait en sorte que la barre de navigation de l'application ait une petite barre rouge qui est persistante dans l'ensemble de l'application. Cette barre contient le nombre de notifications qui attendent intervention de l'utilisateur. Ce nombre diminue pour chaque notification

en attente qui a été résolue. Lorsque l'utilisateur résout toutes les notifications, la barre rouge disparaît. Cette fonctionnalité a nécessité l'utilisation de React, Recoil et React Query.



Figure 34: Exemple de 4 invitations en attente

Voici la page des invitations, où se trouvent toutes les invitations que l'utilisateur a reçues, ou celles qui attendent son autorisation :

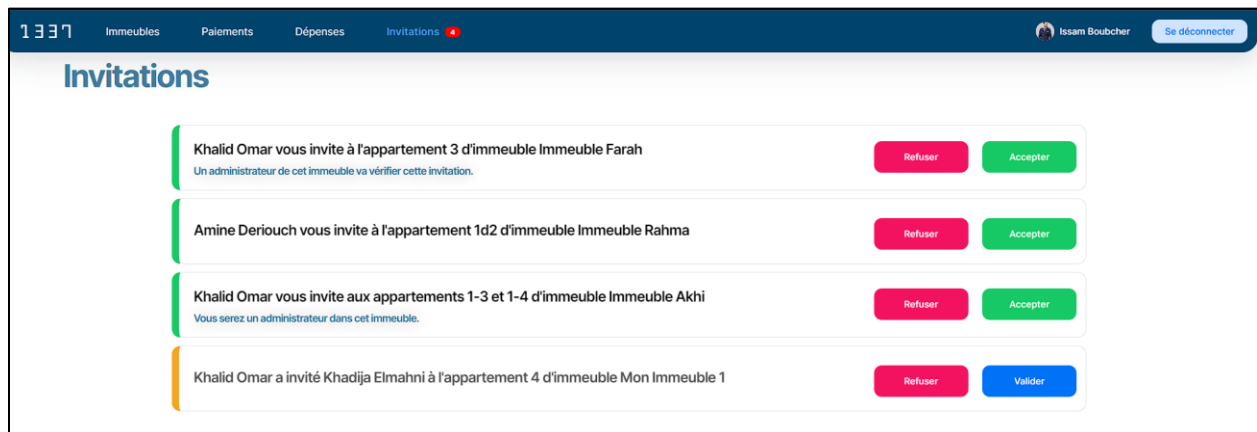


Figure 35: Page d'invitations

L'exemple ci-dessus montre les 4 variantes possibles d'invitations :

- **La première**, est une invitation effectuée par un résident d'Immeuble Farah vers cet utilisateur, lorsque cette invitation est acceptée, elle devra être approuvée par l'un des admins de l'Immeuble Farah avant que l'utilisateur ne devienne un résident de cet immeuble.
- **La seconde**, est une invitation effectuée par l'un des admins d'Immeuble Rahma, dès que l'utilisateur accepte cette invitation, il devient un résident de l'Immeuble Rahma. Aucune validation n'est nécessaire (rappelons que tout ceci est expliqué en détail en 7.3.).

- **La troisième**, est une invitation réalisée par l'un des administrateurs d'Immeuble Akhi, dans celle-ci, dès que l'utilisateur accepte l'invitation, il devient un administrateur dans ce bâtiment.
- **La quatrième**, n'est pas une invitation destinée à l'utilisateur lui-même, au contraire, elle est destinée à Khadija Elmahni, mais parce que la personne qui l'a invitée, Khalid Omar, n'est pas un administrateur de Mon Immeuble 1. Cette invitation nécessite une validation. Et cet utilisateur est l'un des administrateurs de Mon Immeuble 1, donc il peut choisir de valider ou de refuser cette invitation.

Notez que la différence entre une invitation et une demande de validation est la barre à gauche, si elle est **verte**, cela signifie que **l'utilisateur est invité** à un bâtiment, si elle est **orange**, cela signifie que **l'utilisateur doit valider** (ou refuser) une invitation à un bâtiment dans lequel il est administrateur.

8. Conclusion :

Dans ce chapitre, nous avons présenté l'environnement logiciel utilisé lors de développement du site web. Puis nous avons présenté l'application avec toutes ses caractéristiques afin de donner un aperçu du processus de création de cette application.

Conclusion Générale et Perspectives

Ce rapport présente mon projet de stage, j'étais censé réaliser une application de gestion des immeubles, avec la plus grande transparence au cœur de ma mission. Grâce à ce projet, j'ai développé beaucoup de nouvelles connaissances et de savoir-faire dans le domaine du développement d'applications web.

En ce qui concerne l'avenir, j'aimerais développer d'autres outils pour visualiser le flux complet des paiements et des impayés.

J'aimerais également améliorer la fonctionnalité des dépenses, en ajoutant l'option d'ajouter des photos avant et après chaque réparation, ainsi que permettre des discussions communautaires, où les résidents peuvent signaler des problèmes et discuter de leur état directement dans l'application. Cela augmenterait encore la transparence et l'expérience globale de la copropriété.

Je garde un excellent souvenir de ce projet. C'était une très bonne expérience et très encourageante pour moi à l'avenir.

Références

React - The full course: <https://fireship.io/courses/react>

Next.js & React - The Complete Guide: <https://www.udemy.com/course/nextjs-react-the-complete-guide>

React - The Complete Guide: <https://www.udemy.com/course/react-the-complete-guide-incl-redux>

Documentation NextUI: <https://nextui.org/docs>

Reactiflux Community: <https://discord.com/invite/reactiflux>

Stack Overflow: <https://stackoverflow.com>

Documentation Auth0: <https://auth0.com/docs>