|  |  |  |
| --- | --- | --- |
|  | Université Sultan Moulay Slimane Ecole Supérieure de Technologie – FBS |  |

**Rapport de Projet de Stage**

Présenté en vue de l’obtention du

Diplôme Universitaire De Technologie

Filière : Génie Informatique

Présenté et Soutenu par :

**Mr. Issam BOUBCHER**

Encadré par : **Mr. Lahsen EL BOUHALI**



Sous la Thématique :

|  |
| --- |
| **APPLICATION SKYLINE**  **DE GESTION D’IMMEUBLES** |

Soutenu le xx/06/2022 devant la commission d’examen composée de :

Pr. Abdellah Amine Professeur à l’Ecole Supérieure de Technologie de Fkih Ben Salah

Pr. Rachid Ait Daoud Professeur à l’Ecole Supérieure de Technologie de Fkih Ben Salah

Pr. Hassan Faouzi Professeur à l’Ecole Supérieure de Technologie de Fkih Ben Salah

Pr. Naima Soukher Professeur à l’Ecole Supérieure de Technologie de Fkih Ben Salah

# 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. Merci beaucoup monsieur !

Nos remerciements vont également à 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 soutenu, pour leur temps précieux, et pour toutes leurs remarques et recommandations visant à améliorer encore plus notre projet. Que les membres du jury trouvent ici l’expression de nos reconnaissances pour avoir accepté d’évaluer notre travail.

Enfin que 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](http://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](http://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

[Remerciements 2](#_Toc105881237)

[Résumé 3](#_Toc105881238)

[Abstract 4](#_Toc105881239)

[Table des matières 5](#_Toc105881240)

[Introduction générale 7](#_Toc105881241)

[Chapitre 1 : Élaboration de cahier de charges 9](#_Toc105881242)

[1. Introduction : 9](#_Toc105881243)

[2. Problématique : 9](#_Toc105881244)

[3. Spécification générale : 10](#_Toc105881245)

[3.1. Objectif de projet : 10](#_Toc105881246)

[4. Spécifications fonctionnelles : 11](#_Toc105881247)

[5. Spécifications d’interface : 12](#_Toc105881248)

[6. Spécifications non fonctionnelles : 12](#_Toc105881249)

[7. Choix d’une application web : 13](#_Toc105881250)

[8. Conclusion : 14](#_Toc105881251)

[Chapitre 2 : Conceptualisation de l’application 15](#_Toc105881252)

[1. Introduction : 15](#_Toc105881253)

[2. Le choix d’UML : 15](#_Toc105881254)

[3. Diagrammes UML : 16](#_Toc105881255)

[3.1 Diagrammes des cas d’utilisation : 16](#_Toc105881256)

[3.2. Diagramme de classes : 19](#_Toc105881257)

[3.3. Diagrammes de séquence : 20](#_Toc105881258)

[Chapitre 3 : Réalisation de l’application 22](#_Toc105881259)

[1. Introduction : 22](#_Toc105881260)

[2. L’environnement de travail : 22](#_Toc105881261)

[2.1. Visual Studio Code : 22](#_Toc105881262)

[2.2. Git : 23](#_Toc105881263)

[2.3. GitHub : 23](#_Toc105881264)

[2.4. PlanetScale : 24](#_Toc105881265)

[3. Technologies utilisées : 24](#_Toc105881266)

[3.1. Front-end: 24](#_Toc105881267)

[3.1.1. HTML: 24](#_Toc105881268)

[3.1.2. CSS : 25](#_Toc105881269)

[3.1.3 JavaScript : 25](#_Toc105881270)

[3.1.4. ReactJS: 26](#_Toc105881271)

[3.1.5. NextUI: 26](#_Toc105881272)

[3.2. Back-end : 26](#_Toc105881273)

[3.2.1 NodeJS : 26](#_Toc105881274)

[3.2.2 Next.js : 27](#_Toc105881275)

[3.3. Base de données : 28](#_Toc105881276)

[3.3.1. MySQL: 28](#_Toc105881277)

[3.3.2. Prisma: 28](#_Toc105881278)

[3.3.3. Amazon S3: 29](#_Toc105881279)

[3.4. API : 29](#_Toc105881280)

[3.4.1. React Query : 29](#_Toc105881281)

[3.5. Authentification : 30](#_Toc105881282)

[3.5.1. Auth0: 30](#_Toc105881283)

[3.6. Hébergement : Heroku 30](#_Toc105881284)

[3.7. CDN : Cloudflare 31](#_Toc105881285)

[4. Autres outils : 31](#_Toc105881286)

[a. Freenom: 31](#_Toc105881287)

# 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 fin d’études en vue de l’obtention du diplôme 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

## Introduction :

Cette partie est consacrée pour la présentation de la problématique et le cahier des charges de notre l’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.

## 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** totaleet 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.

## 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.

## 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 tous les privilèges 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.
* Modifier une cotisation existante.
* Modifier une dépense existante.

**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).

## 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.

## 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 performant. 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, Safari Browser, Microsoft Edge, Mozilla Firefox et Opera Browser.

## 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.

## 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

## 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 base sur les diagrammes et offrant une flexibilité marquante.

## 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 référencé 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.

## 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 :

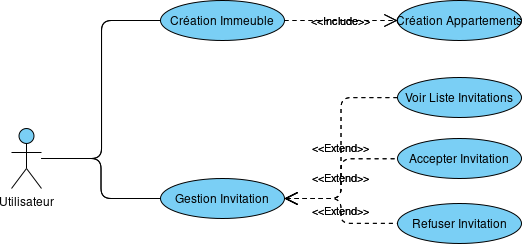


Figure 1 : Diagramme cas d'utilisation d'utilisateur

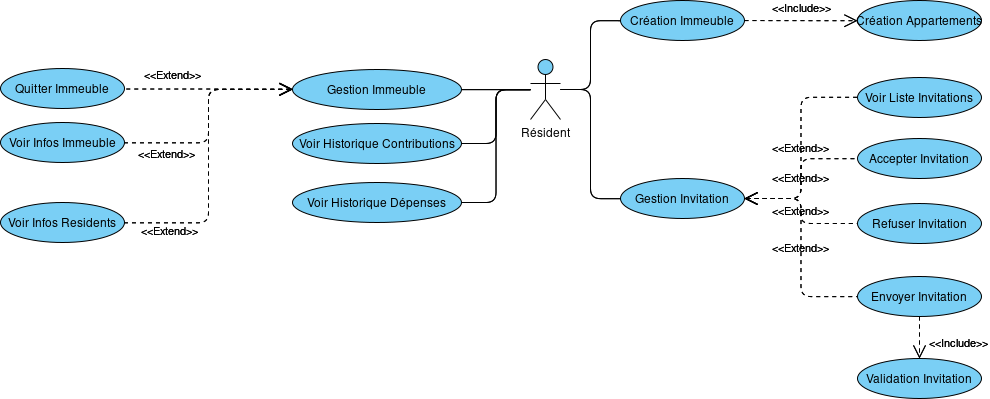


Figure 2 : Diagramme cas d'utilisation d'un résident

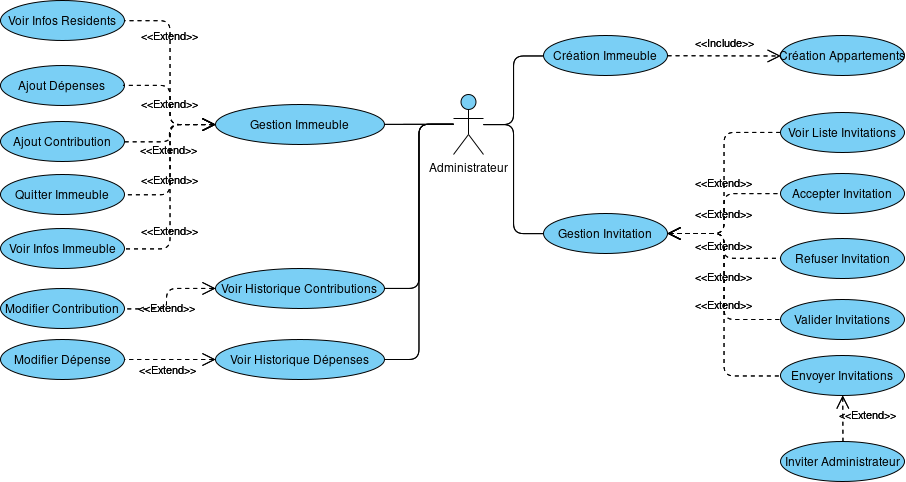


Figure 3 : Diagramme cas d'utilisation d'un administrateur

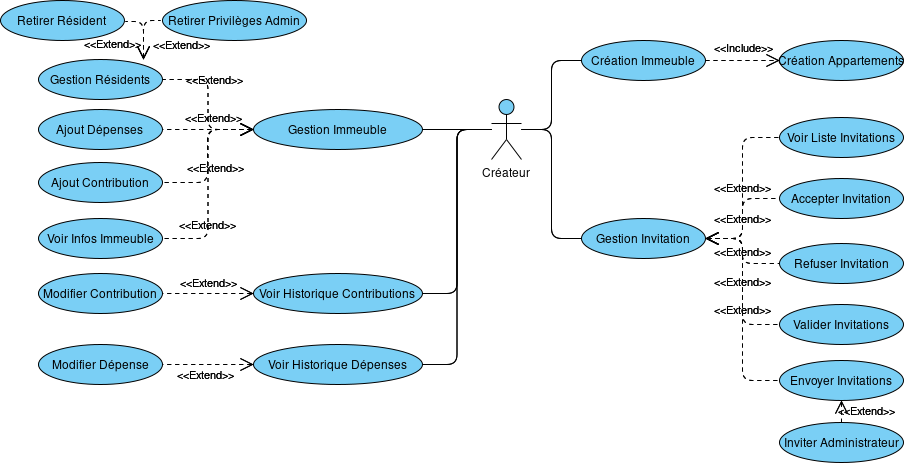


Figure 4 : Diagramme cas d'utilisation d'un créateur

### 3.2. Diagramme de classes :

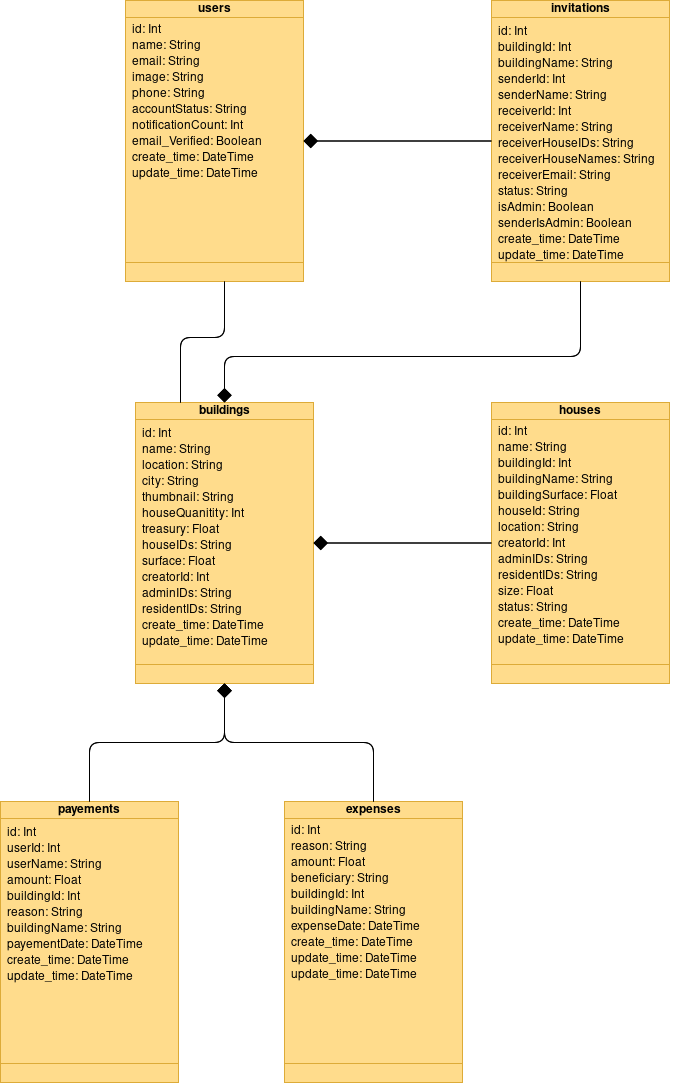
Le diagramme de classes permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d’utilisation. Et il nous permet de modéliser les classes du système et leurs relations indépendamment d’un langage de programmation particulier.

Figure 5 : Diagramme de classes

### 3.3. Diagrammes de séquence :

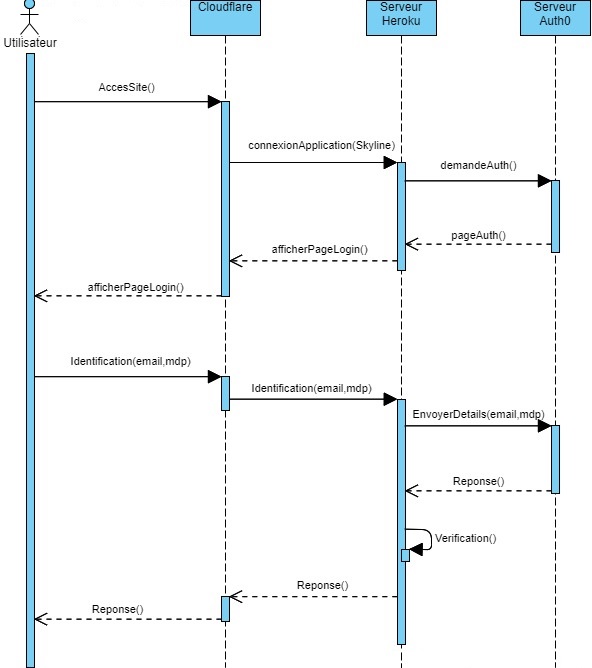
Les diagrammes de séquence sont une solution de modélisation dynamique en langage UML, ils se concentrent plus précisément sur les lignes de vie, les processus et les objets qui vivent simultanément, et les messages qu’ils échangent entre eux pour exercer une fonction avant la fin de la ligne de vie.

Figure 6 : Diagramme de sequence d’authentication

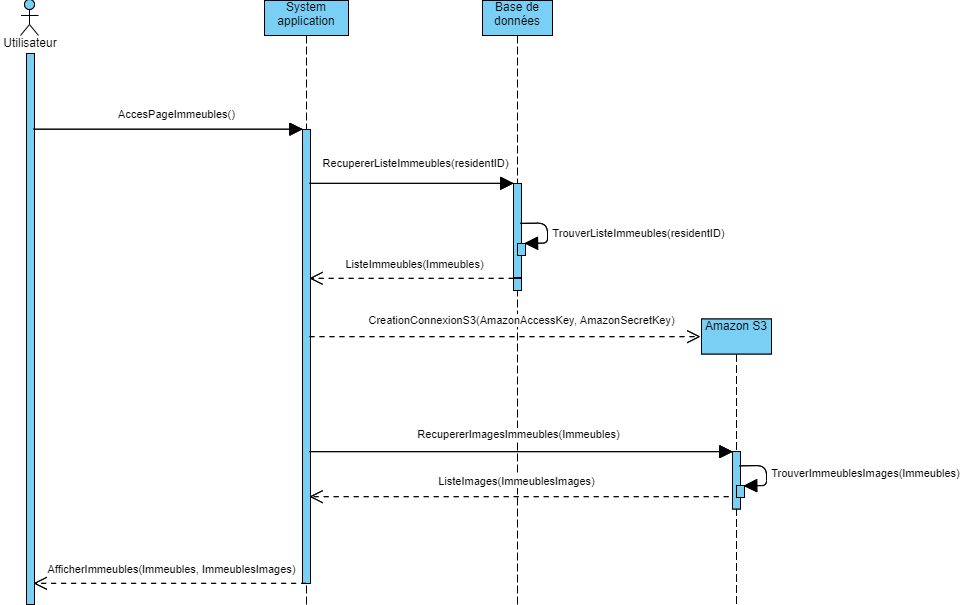


Figure 7 : Diagramme de séquence d'accès au page Immeubles

# Chapitre 3 : Réalisation de l’application

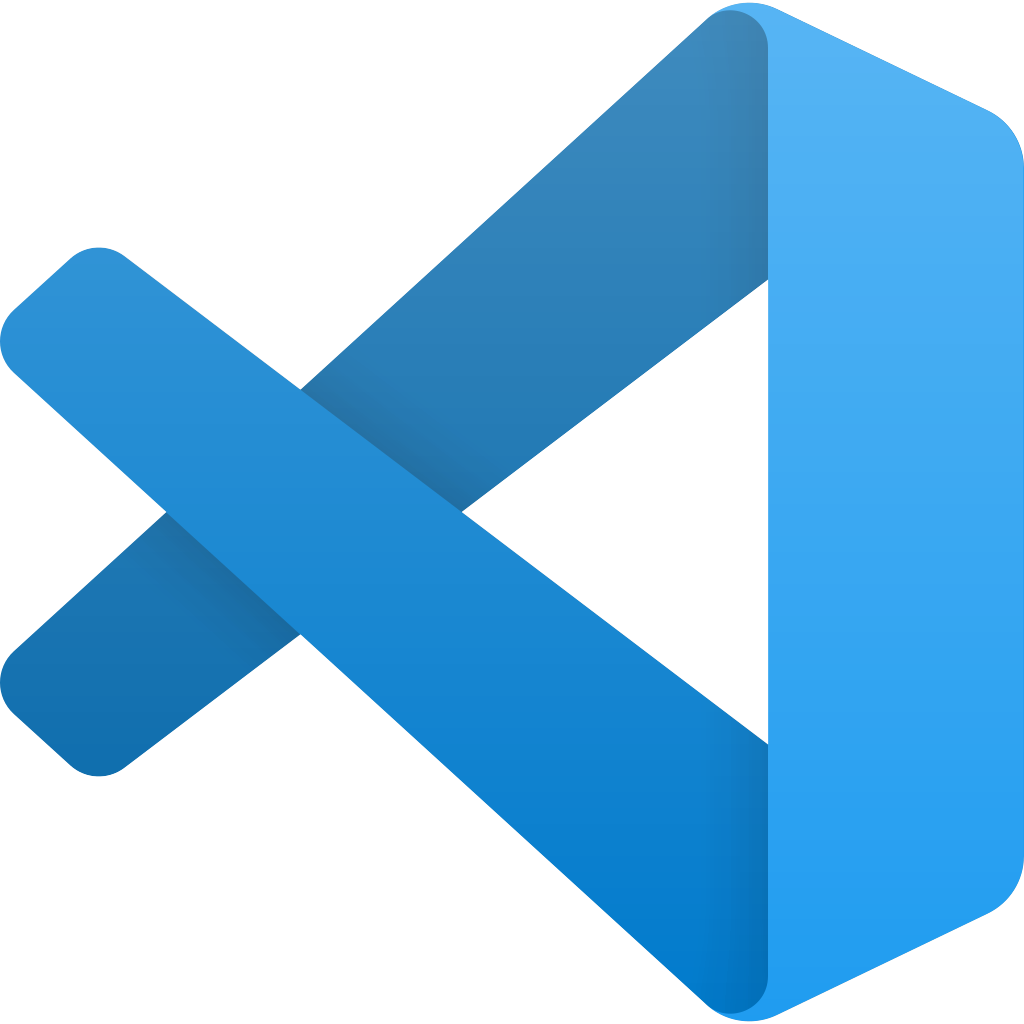
## Introduction :

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.

## 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 support 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.

### GitHub Logo, history, meaning, symbol, PNG2.3. 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.

### 2.4. 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 …).

## Technologies utilisées :

### 3.1. Front-end:

#### 3.1.1. HTML:



**HyperText Markup Langage** est un langage de balisage qui a été créé par Tim Berners-Lee en 1991. Il nous permettre 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, à 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 :

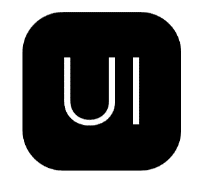
JavaScript désigne un langage de développement informatique, il a été créé par Brendan Eich en 1995. Ce qui rend Javascript spécial, c'est que tous les navigateurs modernes le supportent. Ce qui signifie que l'exécution des tâches est opérée par le navigateur lui-même, sur l'ordinateur de l'utilisateur,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

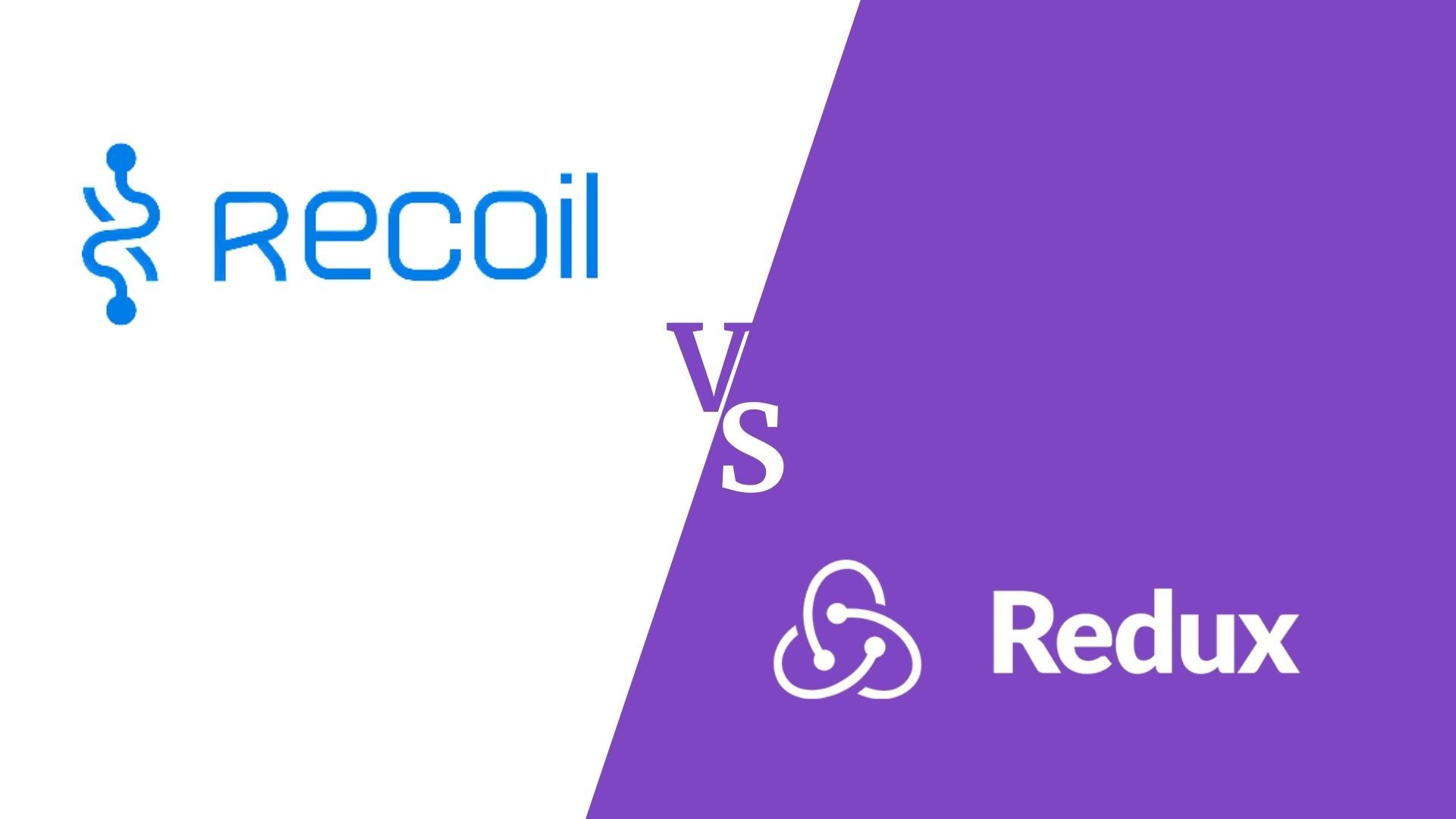
#### 3.1.5. NextUI:



**NextUI** est une bibliothèque de composants d'interface utilisateur, qui permet d'utiliser de composants magnifiques dans la réalisation d'applications.

Nous avons utilisé NextUI pour réaliser notre application.

#### 3.1.6. Recoil:



**Recoil** est une bibliothèque de gestion d'état 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 :

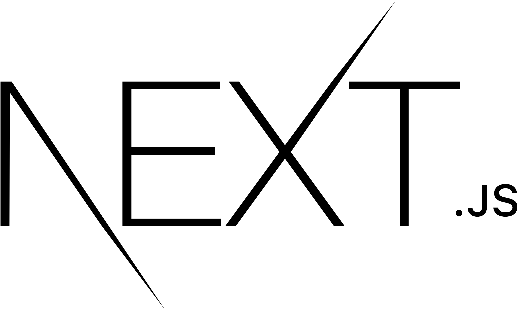


**Node.js** est un environnement d'exécution JavaScript open source, multiplateforme et dorsal qui fonctionne sur le moteur V8 et exécute le code JavaScript en dehors d'un navigateur Web.

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. Amazon 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. API :

#### 3.4.1. 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.

Nous discuterons de React Query plus tard dans ce chapitre.

### 3.5. Authentification :

#### 3.5.1. 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.

## 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](http://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.

## L’architecture de l’application :

Figure 8: L'architecture de l'application

### Cryptage de trafic :

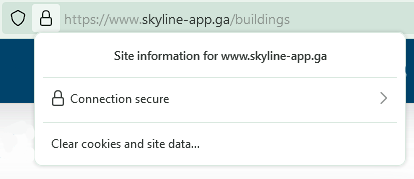


Figure 9: Connection sécurisé avec notre application

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

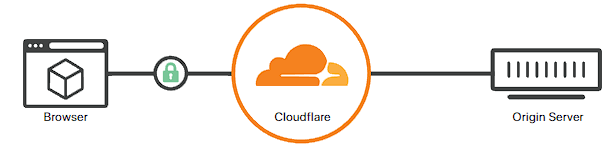
****

Figure 10: Cryptage de trafic depuis et vers Cloudflare

### 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.

### Communication avec la base de données :

Notre application se 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.

### Récupération et stockage des photos :

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 de la page web).

## Optimisations :

### 6.1. Optimisation 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.

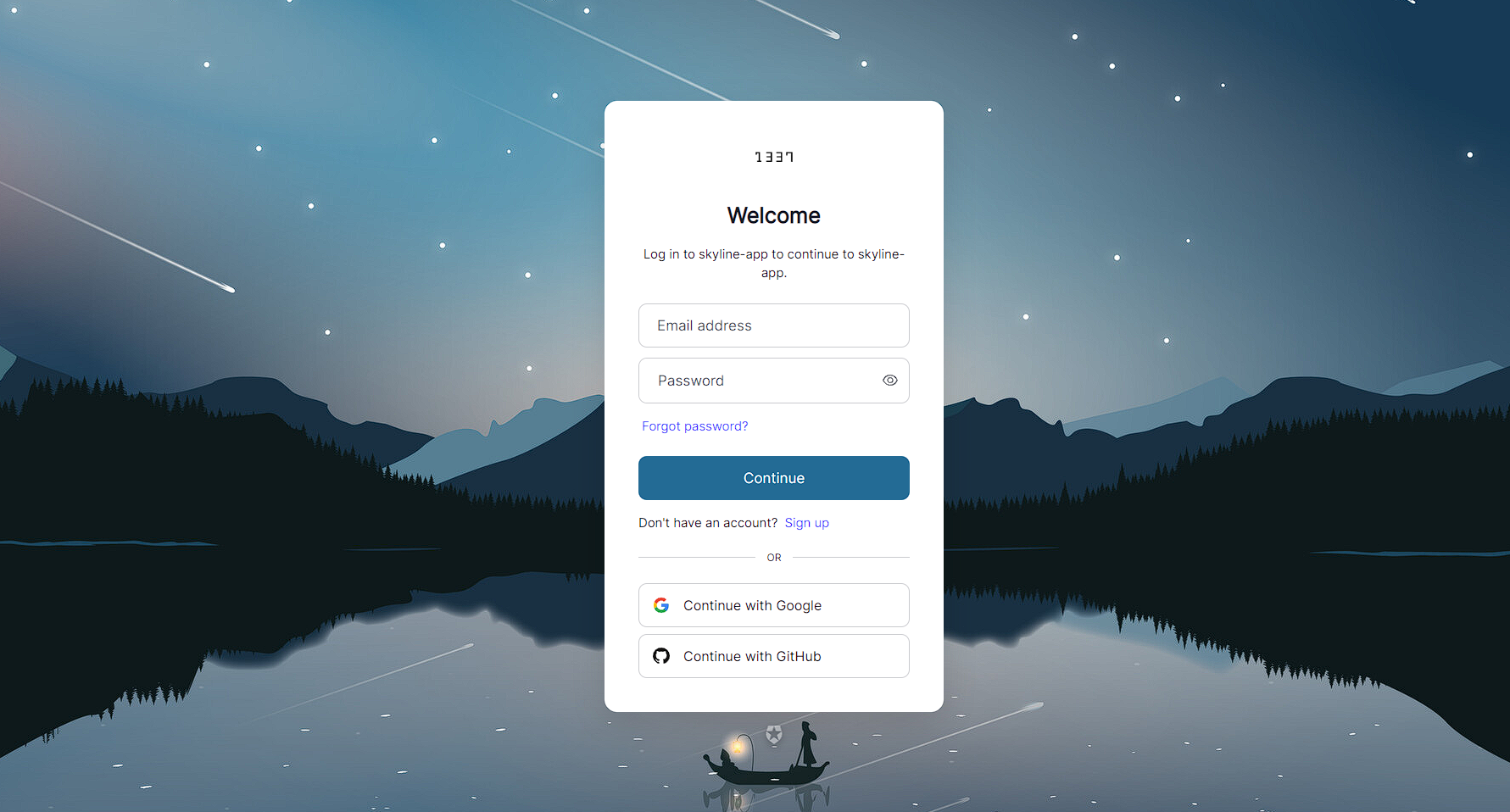


Figure 12: Page d'authentification fournie par Auth0

### 6.2. Optimisation de performance :

La principale raison pour laquelle nous avons opté pour Next.js est qu'il s'agit du meilleur framework 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. Le HTML généré, les données JSON et 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.

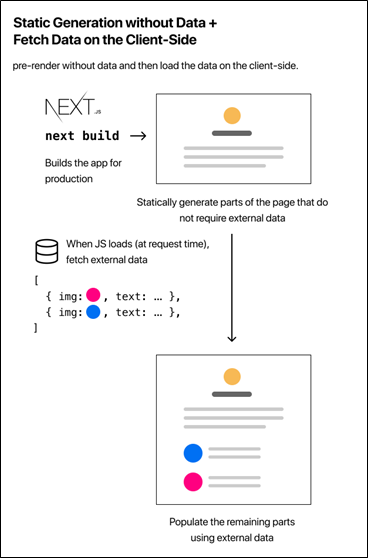
De plus, notre application est une SPA (Single Page Application), ce qui signifie que le fait de passer d'une page à l'autre ne rafraîchit pas complètement le site Web. Au lieu de cela, elle masque dynamiquement le contenu de la page actuelle et affiche l'autre tout en conservant les éléments persistants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

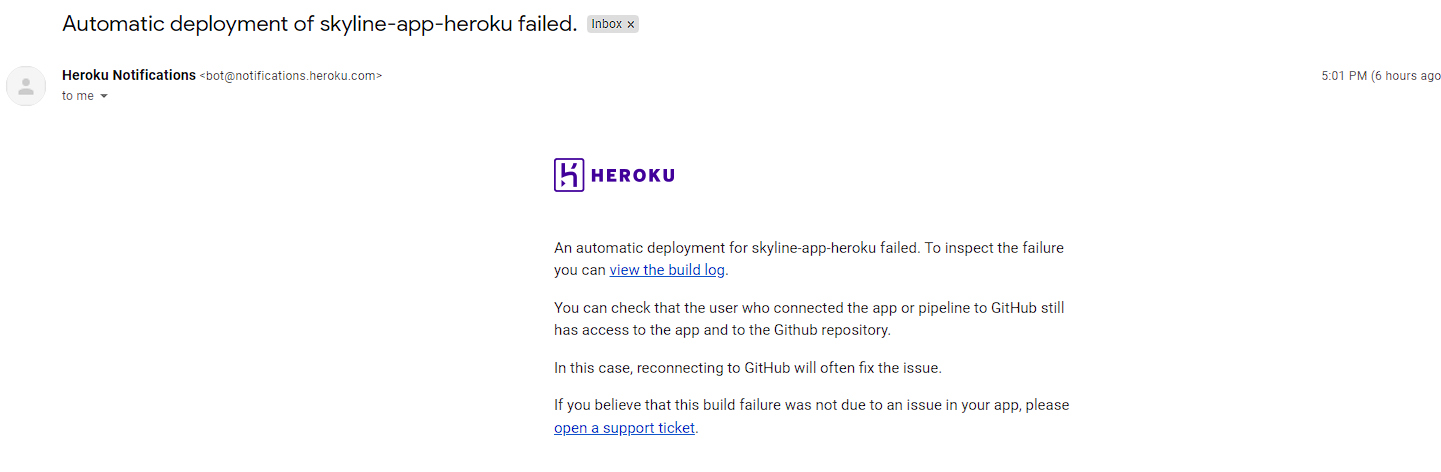
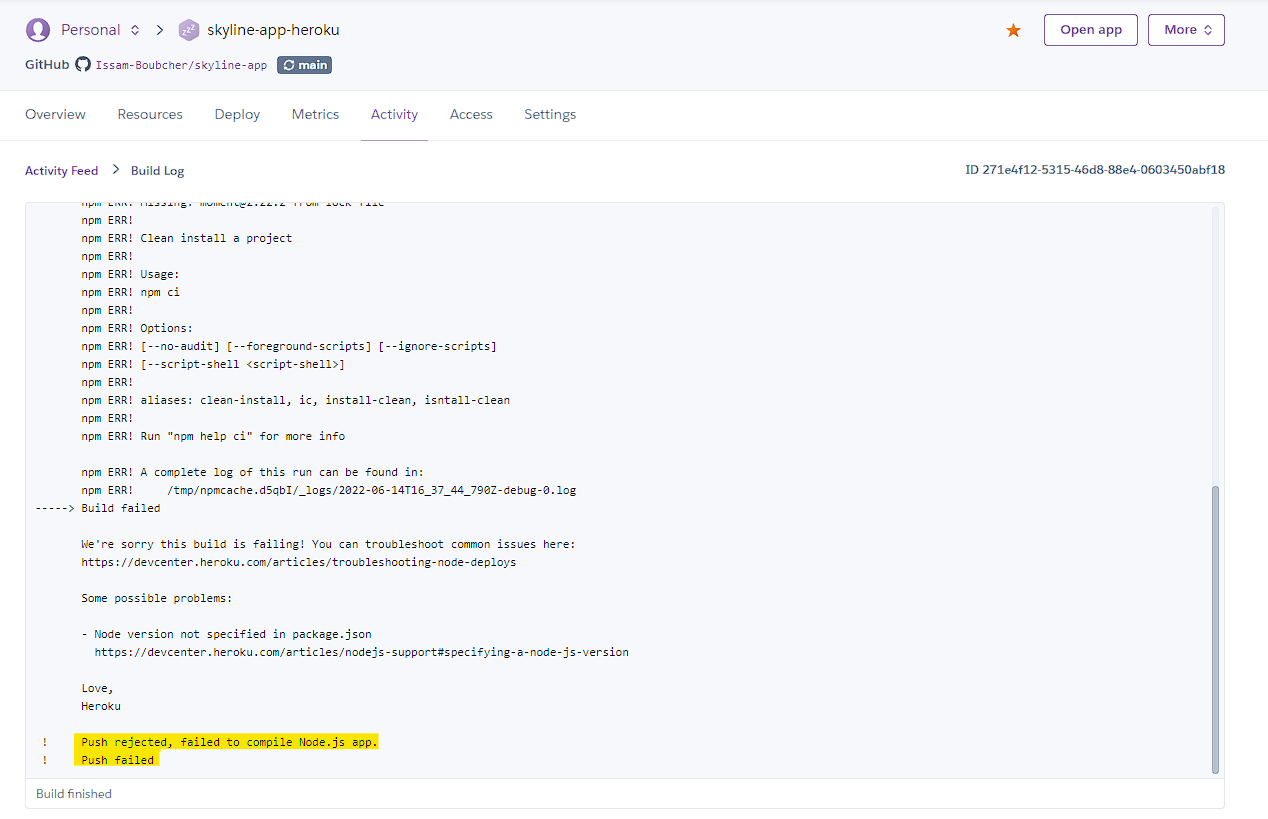
Nous avons configuré notre environnement de déploiement de telle sorte que dès que nous poussons un commit vers notre répertoire GitHub, Heroku détecte immédiatement que le répertoire a été mis à jour et télécharge automatiquement tous les fichiers modifiés, puis, il redéploie le répertoire vers notre 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.

## 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 mis à niveau vers 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 plus. Nous reviendrons plus tard en détail sur le concept des rôles et invitations.

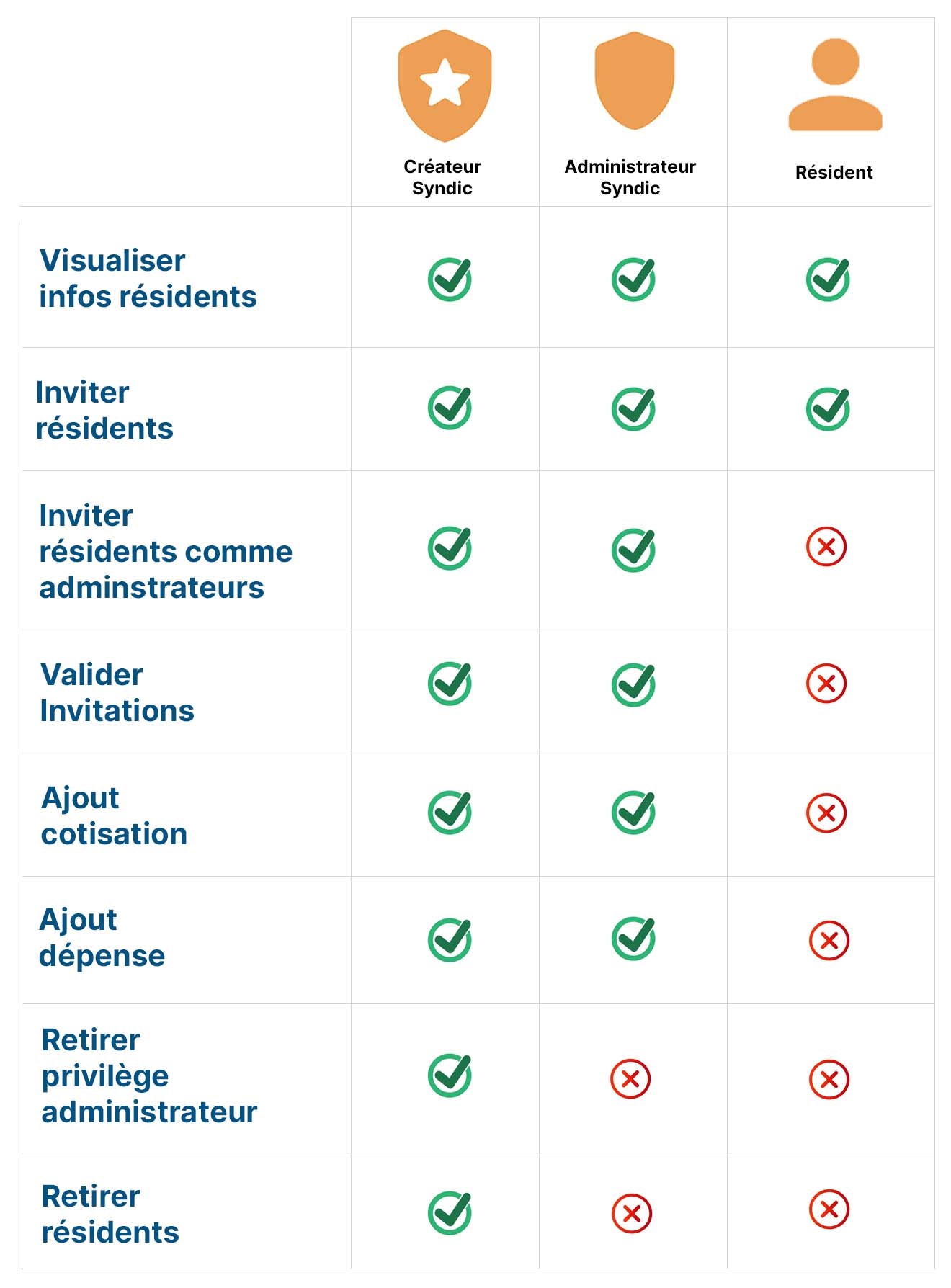


Figure 16: Privilèges des différents rôles des résidents