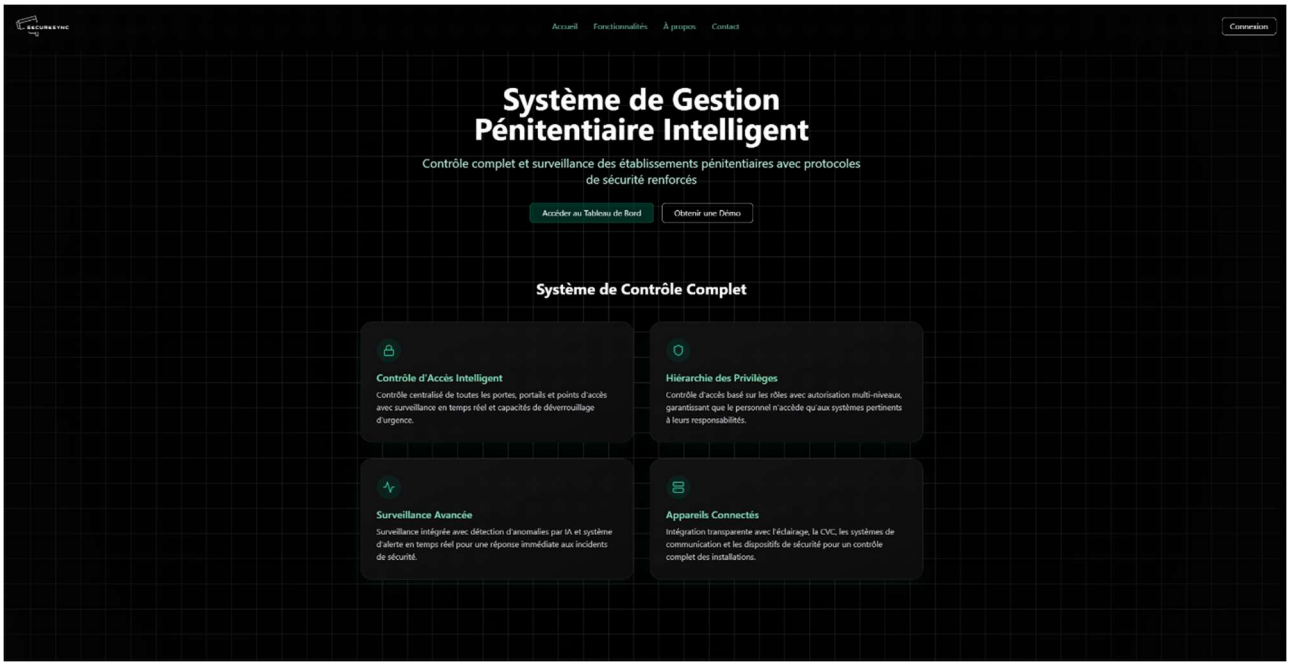
Rapport SecureSync

Rapport de Projet Développement Web

Plateforme intelligente de gestion de prison



Membres du groupe : Adam Terrak, Anthony Voisin, Firas Benmansour, Mehdi Sekkat

Sommaire

Introduction et Contexte

Problématique actuelle

SecureSync est une application web développée pour répondre à une problématique bien réelle dans les établissements pénitentiaires : la gestion fragmentée des appareils connectés. Aujourd'hui, les objets connectés font partie intégrante de notre quotidien, et leur présence au sein d'infrastructures sécurisées telles que les prisons s'impose naturellement. Portes automatiques, caméras, éclairages connectés : ces équipements, désormais indispensables au bon fonctionnement d'un établissement, continuent pourtant de fonctionner de manière indépendante, sans réelle interconnexion.

Cette situation engendre plusieurs problèmes majeurs :

- L'utilisation de **multiples plateformes** (par type ou par marque d'objet), complexifiant la gestion quotidienne ;
- Une **absence de synchronisation** entre les appareils, empêchant toute automatisation efficace ;
- Une **perte de temps** et **de données** due à ce cloisonnement ;
- Une **difficulté à optimiser la consommation énergétique**, alors qu'une coordination entre les équipements pourrait permettre des économies substantielles.

À cela s'ajoute une autre difficulté majeure : l'impossibilité de reconstituer un historique complet des actions effectuées sur chaque appareil, et de les associer précisément à un utilisateur donné. En effet, la gestion par compte unique, encore largement répandue dans les établissements pénitentiaires, empêche d'identifier les interventions individuelles de chaque agent. Ce mode de fonctionnement constitue un frein à l'amélioration de la **traçabilité**, de la **responsabilisation** du personnel et de la **sécurité interne**.

Pour répondre à ces enjeux de **traçabilité** et de **clarté d'usage**, l'interface utilisateur du site a été conçue par **Adam** et **Firas** selon une approche **pragmatique**. Après avoir recherché en ligne des **composants d'interface** pertinents tels que des **boutons**, des **cartes** ou encore des **menus** et les avoir **assemblés** de manière cohérente, ils ont pu

structurer les différentes pages du site. Cette méthode a permis de **gagner du temps** tout en assurant une interface **fonctionnelle** et **harmonieuse**. Après avoir défini une **charte de couleurs** spécifique, des tons **sobres** ont été attribués aux pages de présentation, une palette **lisible** et **épurée** aux interfaces d'utilisation, et des couleurs **distinctives** aux pages d'administration. Cette organisation visuelle contribue à rendre la **navigation plus intuitive** et adaptée aux différents **profils d'utilisateurs**.

Objectifs de l'application SecureSync

C'est pour répondre à ces enjeux que **SecureSync** a vu le jour. Développée avec les technologies suivantes :

- **Django** pour la partie backend,
- **Vite** comme outil de build du frontend,
- **Tailwind CSS** pour la mise en forme graphique,
- **React avec TypeScript** pour la création d'interfaces dynamiques et robustes,

Notre solution permet une **centralisation complète de la gestion des appareils**, enrichie par un **système d'authentification avancé** et une **gestion fine des droits d'accès**.

Au-delà de son utilité dans un établissement unique, **SecureSync** a été conçue pour **gérer plusieurs prisons simultanément**. La structure de sa base de données évite toute duplication inutile :

- Chaque utilisateur est stocké dans une unique table users .
- Les prisons sont identifiées dynamiquement sans nécessiter de tables distinctes.
- Les principales pages de l'application (comme la page d'accueil) sont **communes** à tous les établissements et adaptées automatiquement en fonction de la prison de l'utilisateur connecté.

En synthétisant les différents besoins spécifiques du milieu carcéral, notre équipe a abouti à une solution efficace et évolutive : **SecureSync**.

Répartition des tâches entre les membres du groupe

Stratégie GitHub et branches

Dès le lancement du projet, nous avons mis en place une stratégie GitHub claire. Deux branches principales ont été créées : `frontend` et `backend`. Cette séparation nous a permis de travailler en parallèle sans interférer. Des branches secondaires ont ensuite été utilisées pour le développement de fonctionnalités spécifiques. Plus légères, ces branches ont été fusionnées plus régulièrement, facilitant un cycle de développement agile.

À mesure que le projet progressait, l'intégration des deux branches principales s'est faite progressivement jusqu'à une consolidation finale sur la branche `main`, utilisée pour le rendu final.

Organisation de l'équipe

Afin de mener à bien notre projet, nous avons d'abord réfléchi à la meilleure façon de répartir le travail. Deux approches étaient possibles : soit adopter une structure classique en séparant clairement le développement frontend et backend, soit opter pour une répartition plus transversale, permettant à chaque membre de contribuer aux différentes couches du projet.

Finalement, nous avons choisi une organisation classique, mais équilibrée. Ainsi, **Firas** s'est consacré exclusivement au développement **frontend**, tandis qu'**Anthony** s'est concentré entièrement sur la partie **backend**. Cette spécialisation nous a permis de gagner en efficacité et d'éviter les conflits sur les fichiers. De leur côté, **Adam** et **Mehdi** ont joué un rôle de **polyvalents**, intervenant aussi bien sur le front que sur le back, selon les besoins du projet.

Sur le diagramme de Gantt ci-dessous, nous avons représenté cette organisation :

- Les **blocs rouges** correspondent aux tâches réalisées sur le **frontend**.
- Les **blocs verts** indiquent les tâches liées au **backend**.
- Les **blocs orangés** concernent les tâches annexes (choix des frameworks, nom du site, création du logo, etc.).

- Les **barres verticales bleues** représentent nos différentes réunions d'équipe, jalons essentiels dans l'organisation du projet.
- Les **blocs gris intitulés "MERGE"** marquent les moments clés d'intégration entre les branches.
- Enfin, le **bloc jaune** représente la **soutenance**, prévue juste après la deadline finale du projet

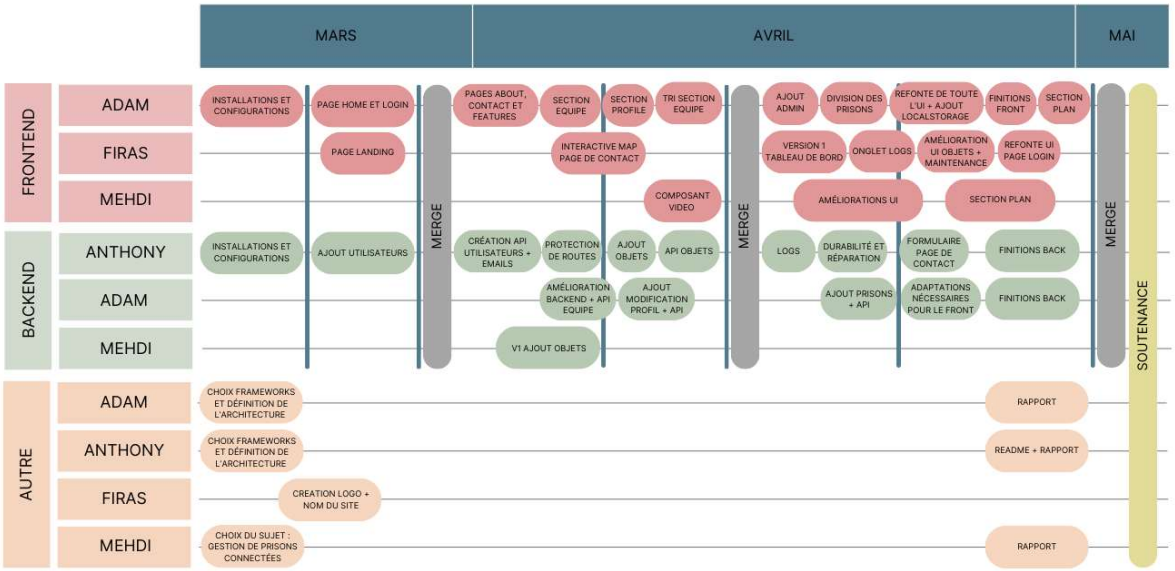


Diagramme de Gantt

Comme l'illustre le diagramme, nous avons globalement respecté la répartition initiale des rôles. Les membres spécialisés sont restés dans leur domaine respectif, tandis que les profils polyvalents sont intervenus là où cela s'avérait nécessaire, apportant ainsi une réelle flexibilité au sein de l'équipe. Cette dynamique de travail est également visible dans l'historique des commits sur GitHub, que nous avons utilisé comme base pour établir le diagramme de Gantt et garantir une traçabilité rigoureuse de nos contributions.

Dès le départ, nous étions à la recherche d'un thème innovant. C'est Mehdi qui a proposé l'idée de développer un site dédié à une prison connectée. Après validation par l'ensemble de l'équipe, nous avons adopté ce sujet original. Nous avons ensuite commencé à structurer notre projet et à réfléchir aux différentes fonctionnalités que nous souhaitions y intégrer.

Nous avons respecté l'organisation des différentes fonctionnalités en **modules distincts**, chacun correspondant à un type d'interaction spécifique avec le système : **Information**, **Visualisation**, **Gestion**, et **Administration**. Cette structuration nous a permis de clarifier les responsabilités de chaque partie de l'application et de mieux **contrôler les accès** aux différentes ressources.

Côté Frontend

Avant même de commencer le développement, nous avons pris le temps de bien réfléchir à la manière d'initialiser le projet frontend, en tenant compte des besoins en performances, en typage, et en structuration du code. Après avoir comparé plusieurs solutions (Create React App, Next.js, etc.), nous avons opté pour **Vite**, un bundler moderne et ultra-rapide, bien adapté aux projets React modernes. Le choix de **TypeScript** s'est imposé naturellement pour bénéficier d'un typage statique robuste, améliorant la fiabilité du code et l'expérience développeur.

Nous avons donc utilisé la commande suivante pour générer notre projet : `npm create vite@latest Securesync -- --template react-ts`
Ce template nous a fourni une base légère, optimisée, et parfaitement compatible avec notre stack (React, Tailwind CSS, ...), tout en nous laissant une grande flexibilité pour organiser notre architecture modulaire dès les premières étapes du développement.

Le développement de l'interface utilisateur a représenté un travail important, tant sur le plan fonctionnel qu'esthétique. Nous avons opté pour **React avec TypeScript**, afin de bénéficier à la fois de la souplesse de React et de la robustesse apportée par le typage statique. L'ensemble du site a été stylisé à l'aide de **Tailwind CSS**, assurant une interface à la fois cohérente visuellement et réactive.

L'application a été structurée autour de **composants modulaires et réutilisables**, ce qui facilite la maintenance, l'évolution du code et la collaboration. Chaque composant remplit un rôle bien défini, avec une séparation claire entre l'affichage, la logique métier et les appels à l'API.

Fonctionnalités clés développées

- **Visualisation et gestion des objets connectés** (portes, lumières, caméras, etc.) via des composants dynamiques affichant l'état en temps réel.
- **Plan interactif** de la prison, permettant une navigation intuitive grâce au positionnement visuel des objets.
- **Dashboards statistiques** présentant des indicateurs clés : consommation énergétique, état global, coût horaire, etc.
- **Consultation des logs** (utilisateurs et objets), avec des filtres par type et date.
- **Gestion des rôles et permissions**, avec affichage adaptatif en fonction des droits.
- **Mode sombre/clair**, activable manuellement ou automatiquement.
- **Interface responsive**, adaptée à tous les types d'écran, même si quelques ajustements restent à faire sur certains iPhones.

Points forts techniques

- Utilisation de **Radix UI** pour des composants accessibles et cohérents.
- Intégration de **Framer Motion** pour des animations fluides et modernes.
- Appels API centralisés avec **Axios**, incluant la gestion des tokens et des erreurs.
- **Routing sécurisé** via un composant `ProtectedRoute` qui redirige les utilisateurs non autorisés.
- **Gestion d'état centralisée** avec les contextes React, pour un accès rapide aux données de session et d'utilisateur.
- Support **multilingue** déjà intégré, avec une première version française.

Sécurité et expérience utilisateur

- **Vérification systématique des tokens JWT** à chaque appel API.
- **Déconnexion automatique** après une période d'inactivité.
- **Accès conditionnel** aux routes et composants sensibles en fonction des permissions.

Un soin particulier a été apporté à la **cohérence visuelle**, grâce à la création de composants personnalisés (boutons, cartes, modales, menus...) partageant un **style unifié**.

Division en modules

Le **module « Information »** est destiné aux visiteurs non connectés. Il représente la partie publique de notre plateforme. L'utilisateur peut uniquement naviguer sur un nombre limité de pages : **accueil**, **informations**, **à propos** et **contact**. Cela permet de présenter les fonctionnalités, de donner une vision d'ensemble des objectifs de la plateforme, et d'offrir une première expérience de navigation, sans compromettre la sécurité ou l'accès aux fonctionnalités sensibles.

Le **module « visualisation »** est destiné aux employés se connectant à la plateforme, il a accès à des pages dédiés tels que le tableau de bord, le plan et les paramètres. Ces pages lui permettent d'interagir avec les objets présents dans le système, comme configurer un thermostat, des éclairages intelligents ou des caméras de surveillance. Toutefois, l'employé junior ne dispose pas des droits nécessaires pour créer ou modifier les objets existants. Ainsi, il peut seulement contrôler l'état des objets sans pouvoir en ajouter ou en modifier les caractéristiques. Il faudra attendre l'obtention de 100 points afin d'y avoir accès.

Le **module « Gestion »** est destiné au **gestionnaire** et au **gérant**, qui disposent de **droits étendus** sur la plateforme. Contrairement aux employés, ce module donne également accès à des fonctionnalités avancées telles que la **gestion des utilisateurs**, la **supervision des activités** effectuées sur la plateforme et l'accès à des **rapports d'utilisation (pour les gérants)**. Le gestionnaire peut attribuer des rôles, suivre la progression des employés (notamment en ce qui concerne le système de points), et garantir le bon fonctionnement global de l'infrastructure. Ce niveau d'accès implique une **responsabilité accrue** et nécessite une **authentification renforcée** pour sécuriser l'accès à ces fonctions critiques. Le gérant peut également créer de nouveaux gestionnaires et gérer leurs profils.

Le **module « Administration »** est réservé aux **administrateurs système**. Il s'agit du niveau d'accès le plus élevé sur la plateforme. Ce module permet de gérer l'**infrastructure technique** dans son ensemble, y compris les **paramètres système globaux**, la **sécurité** (certificats, journaux d'audit, restrictions IP), et l'**architecture des bases de données**. L'administrateur possède également les droits concernant l'ajout ou la suppression d'établissements. Il est le seul utilisateur qui peut changer de prison quand il le souhaite. Les autres modules sont limités à une prison pour chaque utilisateur.

Côté Backend

Pour démarrer notre projet **Django**, nous avons initialisé le projet principal avec la commande **django-admin startproject backend**, ce qui a généré la **structure de base** contenant les **fichiers de configuration**. Nous avons créé plusieurs **applications internes** au projet qui ont ensuite été enregistrées dans le fichier **settings.py** du projet, pour qu'elles soient prises en compte par **Django**.

Avant de commencer réellement le développement du projet, nous avons pris le temps de réfléchir à une **structuration solide du backend**. Notre objectif était de concevoir une **architecture claire, évolutive et facilement maintenable**. Pour cela, nous avons adopté une **approche modulaire** en créant une **application distincte pour chaque fonctionnalité**. Cette séparation permet d'éviter de mélanger des **logiques différentes**, comme celles liées aux **objets** et celles liées aux **utilisateurs**. En suivant ce principe, nous avons également décidé de mettre en place deux **applications distinctes**, nommées **Account** et **Object**, qui avaient pour but de gérer respectivement les **utilisateurs** et les **objets**. Chacune d'elles contenait **deux tables** : une première dédiée à la **gestion des utilisateurs ou des objets**, et une seconde pour la **gestion des logs associés**. Ainsi, dans l'**architecture initiale**, le projet comportait **deux applications** et un total de **quatre tables**.

Cependant, au fur et à mesure de l'avancement du projet, de **nouveaux besoins** sont apparus côté **backend**. Nous avons donc **repensé notre architecture** afin de l'adapter à ces évolutions. Une **nouvelle application dédiée aux statistiques** a été ajoutée pour centraliser le **traitement et l'affichage des données analytiques**. De plus, nous avons **enrichi** l'application **Account** en y ajoutant de **nouvelles tables**, notamment pour implémenter la **gestion de la double authentification** et la **gestion des différentes prisons**.

Le travail backend s'est principalement concentré sur la **gestion de la base de données,** des **API** et l'envoie des e-mails.

1. Application `Account`

Elle gère tout ce qui concerne les utilisateurs et contient quatre tables principales :

- **User** : Table principale avec des champs tels que nom, prénom, username, mot de passe (haché via Django), points, etc.
- **UserLogs** : Enregistre les actions utilisateur (connexion, OTP, modifications, etc.) avec date et commentaire.
- **OTPCodes** : Permet la double authentification par code à 6 chiffres, avec heure d'expiration et indicateur d'utilisation.
- **Prison** : Elle permet à l'administrateur de gérer la **création et la suppression de prisons**.

Cette application comporte **18 API**, permettant notamment de :

- Se connecter
- Modifier son propre compte ou celui d'un autre (avec droits)
- Changer le nombre de points
- Afficher tous les utilisateurs
- Supprimer un utilisateur
- Ajouter des logs
- Afficher les logs
- Changer de Prison
- Changer de rôle

2. Application `Object`

Cette application s'occupe de la **gestion des objets connectés** avec deux tables :

- **Objects** : Stocke les objets avec nom, type, état, consommation, etc.
- **ObjectLogs** : Archive les actions réalisées sur les objets (ajout, suppression, modification d'état...).

Pour cette application, nous avons réussi à bien architecturer notre backend dès le début du projet, car elle contient exactement les tables prévues initialement.

3. Application Statistique

Contient une unique table :

- **Stat** : Enregistre les données statistiques envoyées périodiquement par le frontend (consommation, coûts, répartitions, etc.).

Pour la gestion des statistiques, aucun calcul n'est effectué côté backend. Nous nous contentons de stocker les données transmises par le frontend, qui se charge lui-même des traitements et des calculs nécessaires.

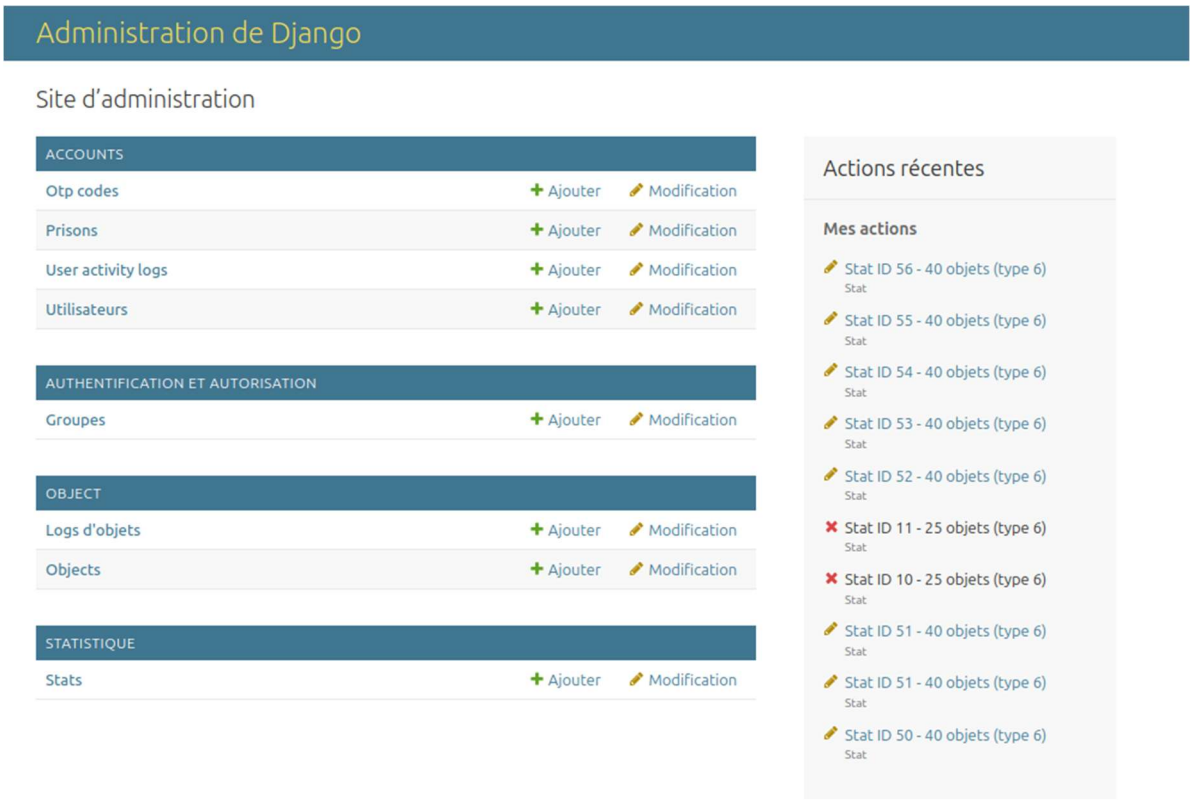
4.Gestion des Emails

La gestion des e-mails a été un aspect essentiel de notre backend, notamment pour renforcer la sécurité et améliorer la communication avec les utilisateurs. Pour cela, nous avons utilisé un **serveur SMTP gratuit fourni par Brevo (anciennement Sendinblue)**, qui permet l'envoi de **jusqu'à 300 e-mails par jour**. Tout d'abord, lors de la **connexion d'un utilisateur**, un **code OTP (One-Time Password)** lui est automatiquement envoyé par e-mail afin de valider la **double authentification**. Ensuite, lors de la **création d'un nouvel utilisateur**, un e-mail contenant ses **identifiants de connexion** (nom d'utilisateur et mot de passe temporaire) est généré et envoyé. Enfin, nous avons mis en place un **formulaire de contact** accessible depuis le frontend ; lorsqu'il est soumis, il **envoie automatiquement un e-mail à l'administrateur** du site contenant le message et les coordonnées de l'expéditeur.

5. Gestion des Modules (Information, Visualisation, Gestion, Administration)

Côté **backend**, des **vérifications strictes** ont été mises en place pour garantir que chaque utilisateur accède uniquement aux modules et données auxquels il est autorisé. Par exemple, l'accès au module **Administration** est réservé exclusivement aux utilisateurs ayant un **rôle administrateur**, tandis que les modules **Information** et **Visualisation** peuvent être accessibles à des profils plus généraux, mais toujours avec des droits limités. Chaque requête adressée au backend passe par une **vérification d'authentification** (token JWT), suivie d'une **vérification des permissions** liées au rôle de l'utilisateur.

Ces contrôles sont essentiels pour **protéger les données sensibles**, limiter les actions possibles selon les droits, et garantir une utilisation conforme du système. Ainsi, un utilisateur standard ne peut pas modifier des informations critiques ni accéder à des statistiques confidentielles, tandis qu'un gestionnaire peut, par exemple, avoir accès aux statistiques liées aux différents utilisateurs ou encore aux objets de sa prison.



Conclusion, Tests et Perspectives

Conclusion

En conclusion, ce projet s'est globalement très bien déroulé et représente une étape importante dans notre parcours de formation. Il nous a permis de découvrir et de nous familiariser avec des outils et technologies que nous n'avions encore jamais utilisés, tels que les frameworks web **Django**, **React** et **Vite**. Leur prise en main a constitué un véritable défi, mais elle nous a permis de renforcer considérablement nos compétences en développement web, tant sur le plan du **back-end** que du **front-end**. Cette première expérience concrète avec ces technologies nous a donné une meilleure compréhension des **architectures modernes d'applications web** et nous a permis de développer des bases solides pour de futurs projets.

Tout au long du projet, nous avons également été confrontés à plusieurs difficultés, notamment liées à la gestion de notre temps en raison de notre emploi du temps chargé en alternance. Il n'a pas toujours été évident de concilier les exigences de l'entreprise,

les cours et les impératifs du projet. Cependant, ces contraintes nous ont poussés à **revoir notre organisation**, à mieux anticiper les imprévus et à **travailler de manière plus rigoureuse**. Ces défis, bien que parfois exigeants, se sont révélés **formateurs** et nous ont permis de renforcer notre capacité à **travailler sous pression** et à **respecter les délais**.

Tests et Vérifications

Afin de garantir une expérience utilisateur fluide sur l'ensemble des supports, nous avons réalisé plusieurs **tests de réactivité** du site sur différents appareils. Nous avons utilisé les **outils de développement intégrés des navigateurs** (comme les DevTools de Chrome) pour simuler plusieurs résolutions d'écran (mobile, tablette, desktop). Ces tests nous ont permis de détecter certains problèmes d'affichage et de les corriger rapidement.

Nous avons également testé le site **en conditions réelles**, en le consultant directement depuis plusieurs smartphones : des appareils **Android** de différentes tailles, ainsi que des **iPhones** (notamment les modèles récents comme l'iPhone 16e et 15). Ces tests nous ont permis d'observer un **affichage globalement satisfaisant** sur la majorité des écrans, mais aussi de constater des **problèmes d'adaptation sur certains iPhones**, notamment au niveau du redimensionnement de certains composants et du menu de navigation.

Ces vérifications nous ont permis de **mettre en évidence les limites actuelles de notre mise en page** et soulignent l'importance de tester systématiquement l'interface sur une large variété de terminaux pour garantir une compatibilité maximale. Des ajustements CSS et l'utilisation de **media queries plus ciblées** seront nécessaires pour améliorer cette compatibilité dans les versions futures du site.

Perspectives

Malheureusement, nous avons identifié un point d'amélioration important concernant la **responsivité de l'interface sur certains appareils iOS** (iPhone). Alors que le site s'affiche correctement sur la majorité des appareils Android, certaines pages ne s'adaptent pas parfaitement aux dimensions d'écran des iPhone, ce qui peut affecter l'expérience utilisateur.

Ce projet a été un véritable **défi** et nous a permis d'améliorer nos compétences dans de nombreux domaines : **développement web**, **lecture de documentation** (notamment pour comprendre les frameworks et leur fonctionnement), **architecture du code**, ou encore **gestion de notre planning**. Nous sommes globalement **satisfaits du résultat**, tant d'un point de vue **esthétique** (front-end) que **fonctionnel** (back-end). Bien que le sujet soit très encadré, nous avons réussi à l'élargir en ajoutant des **fonctionnalités non demandées**, comme la **gestion de plusieurs prisons**. Ce projet nous a également donné l'opportunité de **collaborer avec des personnes avec lesquelles nous ne travaillerions pas instinctivement**. Cela a permis à chacun d'**ouvrir son esprit**, ce qui nous sera utile aussi bien dans notre **vie personnelle** que **professionnelle**.

Références

- Django Software Foundation. (2025). *Django (Version 5.x)* [Computer software]. https://www.djangoproject.com/
- Tailwind Labs. (2025). *Tailwind CSS (Version 3.x)* [Computer software]. https://tailwindcss.com/
- Meta. (2025). *React – A JavaScript library for building user interfaces (Version 18.x)* [Computer software]. https://reactjs.org/
- Microsoft. (2025). *TypeScript: JavaScript with syntax for types (Version 5.x)* [Computer software]. https://www.typescriptlang.org/
- Evan You. (2025). *Vite – Next Generation Frontend Tooling (Version 5.x)* [Computer software]. https://vitejs.dev/
- Irabor, J. (2021). *How To Build a To-Do application Using Django and React*. DigitalOcean. https://www.digitalocean.com/community/tutorials/build-a-to-do-application-using-django-and-react

Annexes

Page de connexion :

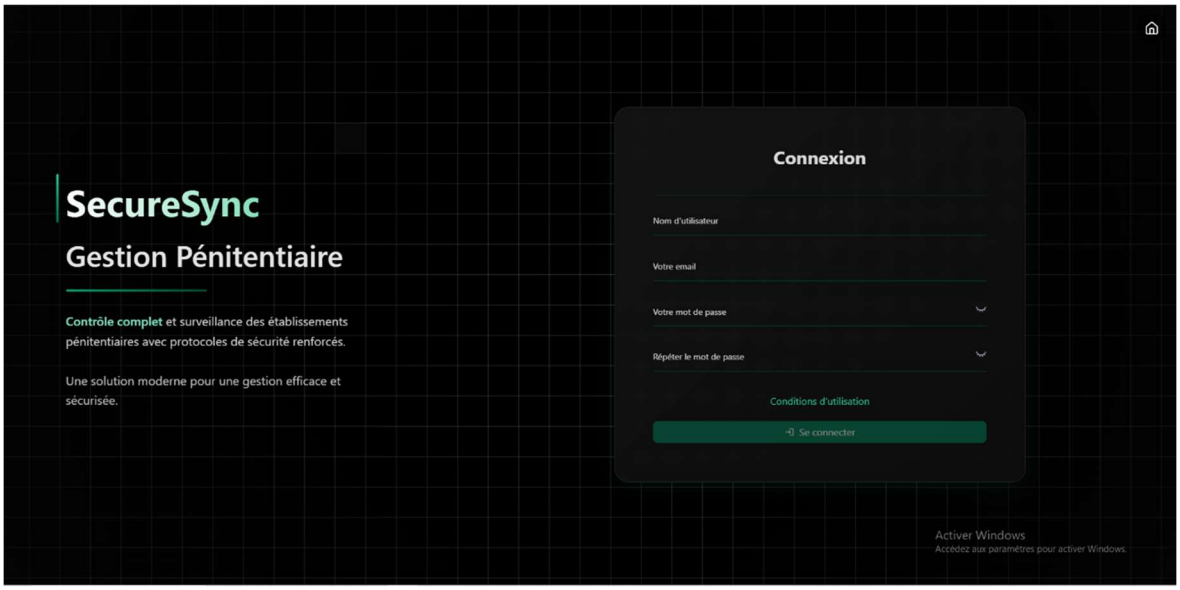
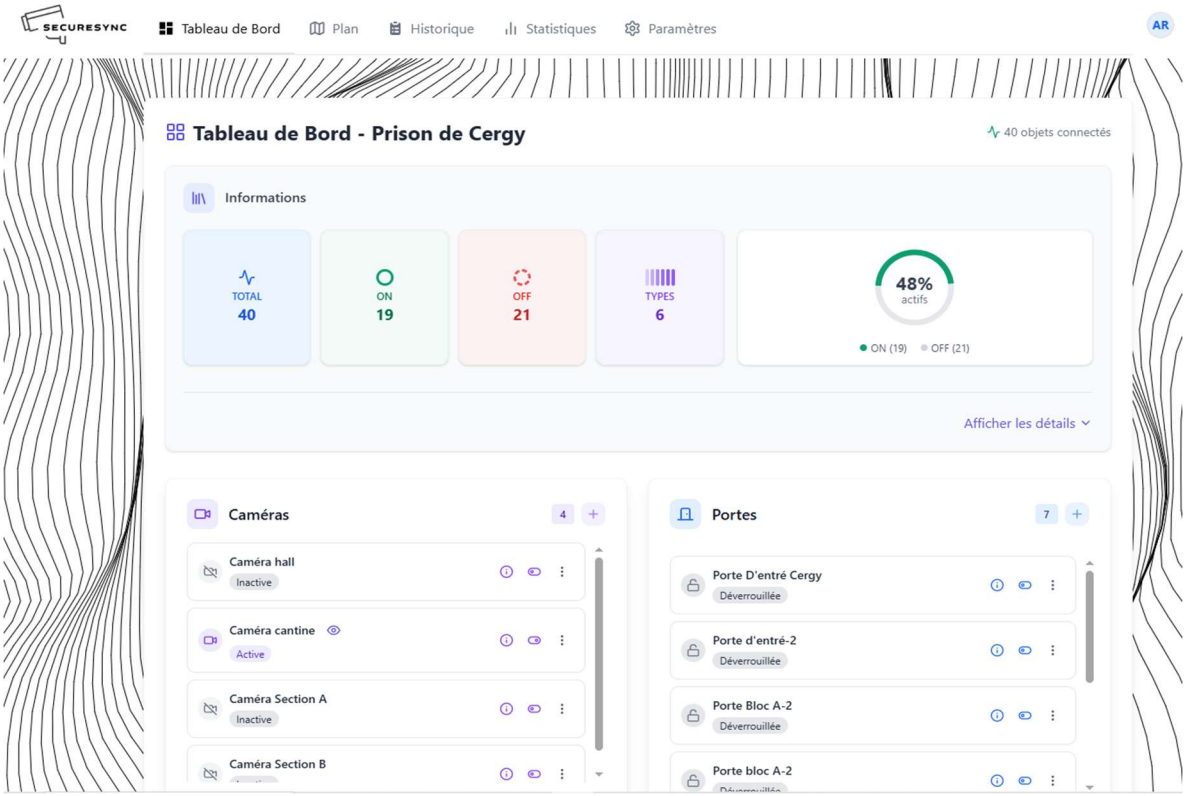
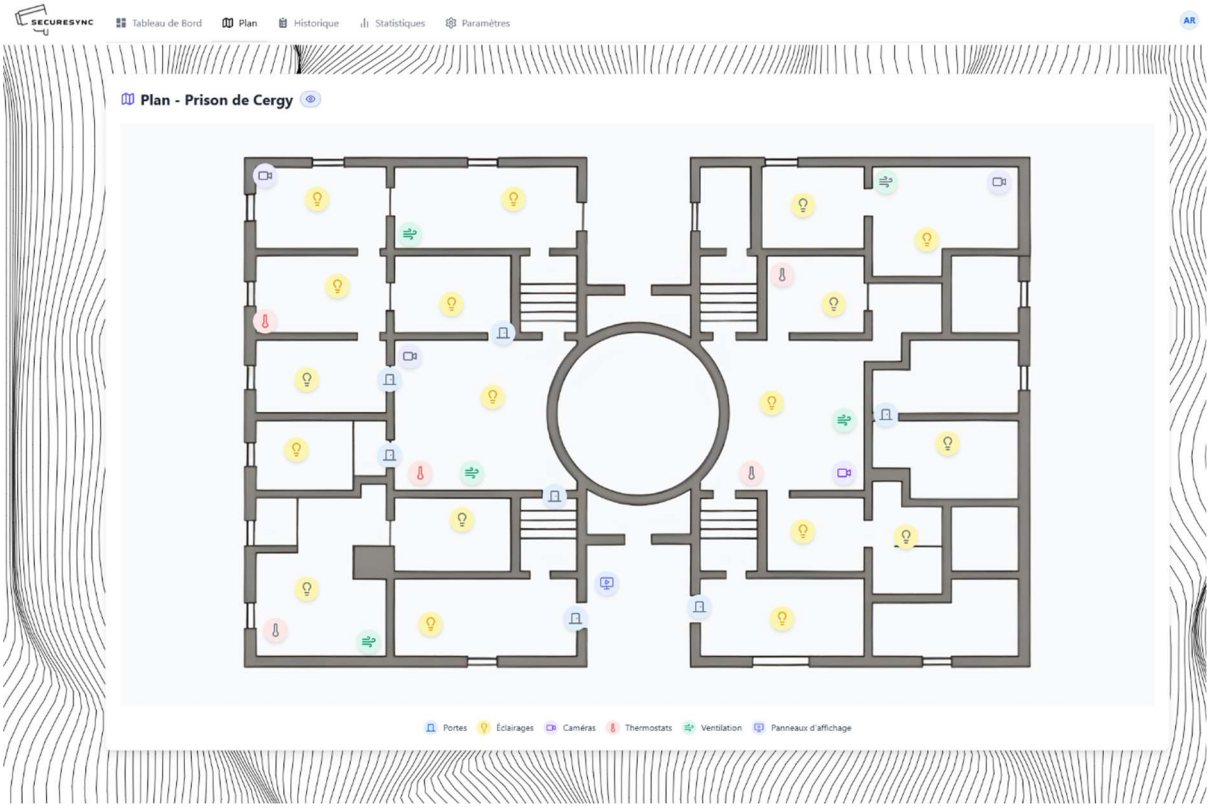


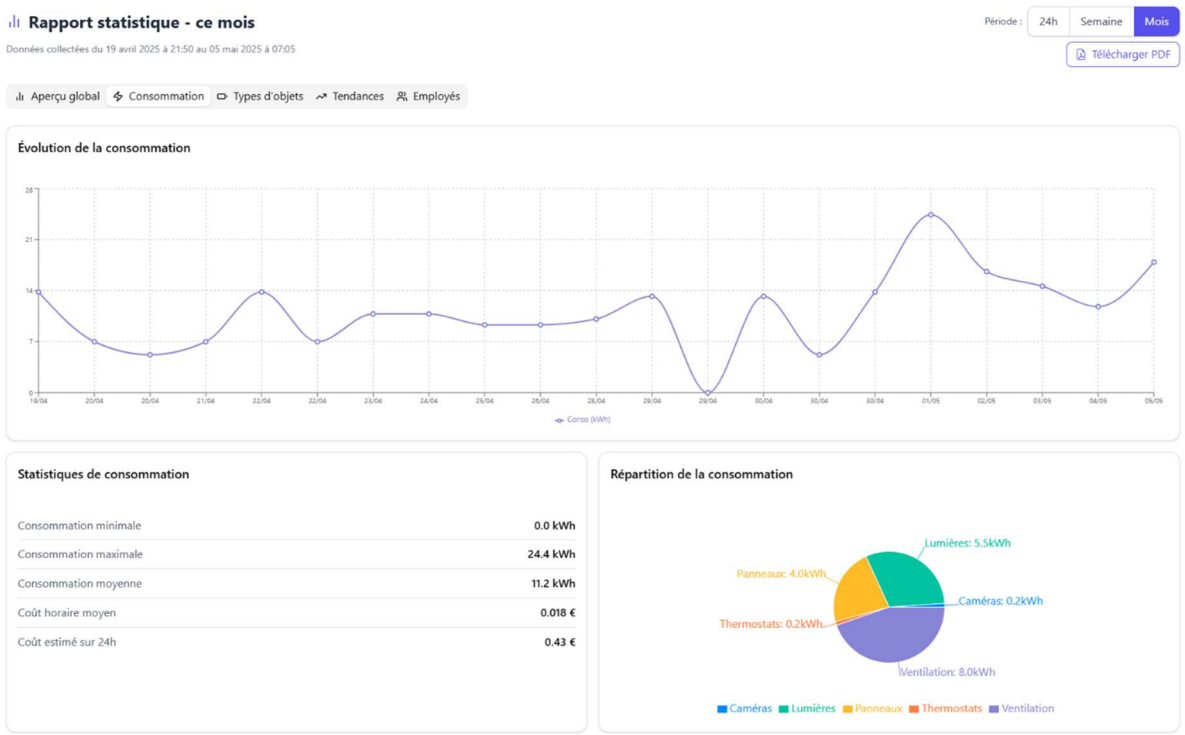
Tableau de bord:



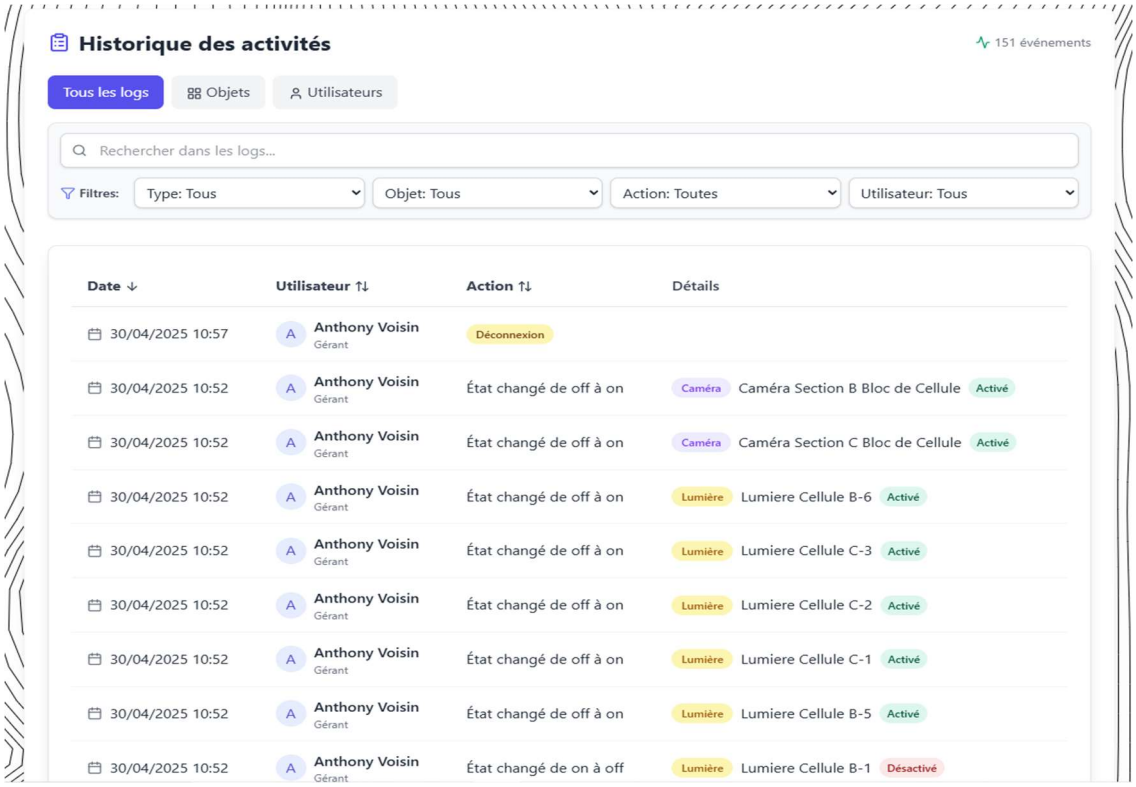
Plan :



Statistiques :



Historique :



Gestion des prisons :