

PROJET ACCESS

Bastion : Avec nous, vous ne passerez pas..

SOMMAIRE

SOMMAIRE	2
INTRODUCTION	4
CHAPITRE 1 : INTRODUCTION DU PROJET	4
A) Contexte	4
B) Présentation du projet	4
C) Objectifs du projet	4
D) Description de l'Équipe et des Rôles	5
CHAPITRE 2: DÉVELOPPEMENT DU FRONT-END	5
A) Objectifs d'une interface attrayante et fonctionnelle	5
B) Intégration de l'accessibilité et de l'ergonomie	5
C) Conformité et respect avec les maquettes et la charte graphique	6
D) Technologies et Frameworks Utilisés : Justification de ces choix	6
E) Implémentation des mécanismes de sécurité standard	8
CHAPITRE 3 : DÉVELOPPEMENT BACK-END	8
A) Choix des technologies back-end et des bases de données	8
B) Stratégies de sécurité en profondeur : filtrage des données, authentification, journalisation, contrôle d'accès	10
C) Mise en place de l'authentification et l'autorisation solides	11
D) Chiffrement des données sensibles	12
E) Intégration des bonnes pratiques d'écoconception pour minimiser l'impact écologique	13
F) Compatibilité et Conformité avec les Standards Actuels	13
CHAPITRE 4 : DÉVELOPPEMENT ET CONSOMMATION D'API	14
A) Conception et développement de l'API	14
B) Sécurisation de l'accès à l'API et mécanismes d'authentification	15
C) Documentation de l'API	15
D) Processus de documentation et publication de l'API pour les développeurs	16
CHAPITRE 5 : STRATÉGIE DE TESTS, ASSURANCE QUALITÉ	17
A) Élaboration d'un plan de test exhaustif et structuré	17
B) Méthodologie de tests (unitaires, d'intégration, etc.) et couverture du code	18
C) Industrialisation du Développement	19
D) Automatisation des tests et gestion des dépendances	19
E) Optimisation de la chaîne de build et amélioration des performances	20
CHAPITRE 6 : DÉVELOPPEMENT HARDWARE ET HÉBERGEMENT	21
A) Développement Lecteur de badge	21
B) Déploiement et hébergement	21
CHAPITRE 7 : OPTIMISATION POUR LE SEO ET MARKETING DIGITAL	22
C) Stratégies de Référencement Naturel	22
D) Mise en œuvre des meilleures pratiques pour le SEO	22
E) Mesure de Performance Marketing	23
CHAPITRE 8 : FONCTIONNALITÉS IMPORTANTES	23
A) Calcul du temps de travail (journalier / hebdomadaire / mensuel).	23
B) Gérer les autorisations d'accès.	24

C) Le système de “badgeage” (badge ou puce RFID , application mobile).	25
D) Prévoir un système paramétrable d'export csv pour l'intégration dans un logiciel de RH.	27
CHAPITRE 9 : CONCLUSION	28
A) Conclusion	28
B) Remerciements	28
ANNEXES	29
Page d'accueil de l'application web	29
Page : Gestion des utilisateurs de l'application web :	29
Page de référencement SEO :	30
Organisation github :	30
Lecteur de badge :	31
Diagramme de la base de données :	32
Page génération du QR code sur l'application mobile :	33
Page calendrier professionnel sur l'application mobile :	34
Exemple de génération du CSV :	34

INTRODUCTION

CHAPITRE 1 : INTRODUCTION DU PROJET

A) Contexte

Ce projet est une simulation d'un projet qui pourrait être porté dans toutes organisations désirant suivre et évaluer les compétences de ses collaborateurs et de ses équipes.

Les membres du groupe seront désignés dans le présent document comme « le prestataire ». L'accent sera mis sur les qualités techniques

B) Présentation du projet

Le principal enjeu du projet “**Accès**” est de mettre en place un système de contrôle d'accès à une infrastructure. Ce système doit permettre de contrôler les entrées et les sorties dans une entreprise en intégrant un système de calcul du temps de présence (pointeuse).

Cet outil doit :

- Permettre une optimisation des ressources humaines sur le temps de présence des salariés.
- Calcul du temps de travail (journalier / hebdomadaire / mensuel).
- Identifier les personnes présentes dans l'entreprise.
- Gérer les autorisations d'accès (heures ouvrées, jours de travail, week-end, congés).
- Identifier les personnes ayant tenté d'entrer dans l'établissement à des heures non autorisées.
- Le système de “badgeage” pourra se faire selon plusieurs méthodes (badge ou puce RFID , application mobile).
- Pour les points d'accès des cartes Raspberry ou des cartes arduino peuvent être fournis.
- Prévoir un système paramétrable d'export (csv , xml , ...) pour l'intégration dans un logiciel de RH.

C) Objectifs du projet

L'objectif du projet est de développer intégralement une application mobile et web permettant de contrôler les accès d'une infrastructure :

- Gestion des profils autorisés
- Gestion du temps de travail
- Sécurité de l'infrastructure

D) Description de l'Équipe et des Rôles

Pour le développement de ce projet, nous avons composé une équipe de 3 stagiaires issus de la promotion des Développeurs Full-Stack : Romain GILOT, Loan KEOVILAY et Noé ZIADI.

- Romain GILOT travaillera principalement sur le développement front-end.
- Loan KEOVILAY travaillera principalement sur le développement back-end, communication et base de données et hardware.
- Noé ZIADI travaillera principalement sur le développement front-end et api, de plus il a été désigné comme chef de projet.

CHAPITRE 2: DÉVELOPPEMENT DU FRONT-END

A) Objectifs d'une interface attrayante et fonctionnelle

Tous les membres risquent d'utiliser souvent les applications. Il faut donc fournir une application fluide, compréhensible et soignée grâce à différentes actions :

- **L'amélioration de l'expérience utilisateur** permet aux utilisateurs de se déplacer facilement sur l'application grâce à une interface claire et simple.
- **L'utilisation simple** permet aux utilisateurs de trouver facilement et sans effort ce qu'ils recherchent grâce à une disposition intuitive des éléments.
- **Un visuel plaisant** incite les utilisateurs à utiliser les applications. Par ailleurs, c'est la première chose sur laquelle les usagers vont se baser pour déterminer la qualité des outils.

B) Intégration de l'accessibilité et de l'ergonomie

Le développement des applications a été fait de sorte à ce que tous les utilisateurs souffrant d'un handicap puissent utiliser les solutions logicielles de la même façon que les personnes bien portantes et de tous âges.

La conception a été rendue accessible en permettant à tous les types de téléphone et selon leurs performances d'accéder aux applications.

C) Conformité et respect avec les maquettes et la charte graphique

Afin de garantir le respect et la conformité des maquettes, l'équipe de développement a pris soin de préparer un **patron de conception** (design pattern) ainsi qu'une **palette de couleurs** pour créer une identité visuelle à l'application. L'application web et mobile se fonde sur les mêmes couleurs et éléments assurant aux utilisateurs une expérience agréable sur les deux appareils.

Exemple de l'application développé :

The screenshot displays the Bastion application interface. At the top, there is a dark header bar with the logo "BASTION" on the left and a user profile "John Doe" on the right. Below the header, a navigation bar includes links for "Accueil", "Absence", "Planning", and "Assistance". On the far right of the header are icons for notifications and help.

The main content area starts with a greeting "Bonjour John," followed by a message "Bienvenue sur votre espace de gestion de temps de travail." To the right of the greeting are two buttons: "Accéder à votre profil" and "Accéder au panel Admin".

Below the greeting, there is a section titled "Temps de travail" (Working time) featuring three colored boxes: blue (0h00 Journalier), red (0h00 Hebdomadaire), and orange (4h01 Mensuel).

Next is a section titled "Mes absences" (My absences) with a table:

Soumis	Raison	Status	Période
Soumis	Congé annuel	Soumis	Du : 2024-04-01 Au : 2024-04-03

Finally, there is a section titled "Mes permissions" (My permissions) with two buttons: "All access" and "Accès premier étage".

D) Technologies et Frameworks Utilisés : Justification de ces choix

Le front-end de l'application a été développé en utilisant deux technologies : **React.js** pour la version web et **React Native** pour la version mobile. Ces frameworks, maintenus par **Facebook**, sont largement reconnus pour leur efficacité dans le développement d'interfaces utilisateurs interactives et réactives.

Afin d'assurer une performance optimale de l'application, le prestataire a opté pour l'utilisation de ces frameworks car :

- **Composants réutilisables** : React utilise des composants pour assurer le développement, ce qui permet de créer des éléments réutilisables. Cette modularité facilite la maintenance du code et accélère le développement en permettant de réutiliser des éléments d'interface utilisateur.
- **Réactivité** : Grâce à la réactivité des composants, toute modification de l'état entraîne une mise à jour automatique de l'interface utilisateur. Cela rend le développement plus efficace et réduit les risques d'erreurs.
- **React Native pour le développement mobile** : React Native donne la possibilité de compiler le code pour les plateformes iOS (utilisant Swift) et Android (utilisant Java), ce qui permet d'économiser du temps et des ressources tout en étant maintenable.
- **Réutilisation du code** : En utilisant ReactJS pour le développement web et React Native pour le développement mobile, une grande partie de la logique métier, des composants et même des bibliothèques tierces peuvent être partagées entre les deux plates-formes. Cela réduit la duplication du travail, nous permettant de nous concentrer sur des aspects spécifiques à chaque plateforme lorsque cela est nécessaire.

Exemple de composant réutilisable :

```
<div>
  <div className="pb-5">
    <span className="text-md font-bold text-black">Temps de travail</span>
  </div>
  <div className="flex flex-row flex-wrap space-x-0 md:space-x-10 space-y-5 md:space-y-0">
    <Card rounded="rounded-lg" number={absenceData?.daily_hours} text="Journalier" color="bg-blue-400" />
    <Card rounded="rounded-lg" number={absenceData?.weekly_hours} text="Hebdomadaire" color="bg-red-400" />
    <Card rounded="rounded-lg" number={absenceData?.monthly_hours} text="Mensuel" color="bg-orange-400" />
  </div>
</div>
```

Dans ce cas, le composant **Card** est utilisé 3 fois pour pour l'affichage des statistiques du temps de travail. Il permet d'afficher des données différentes en se basant sur la même forme.

Temps de travail

0h00

Journalier

7h00

Hebdomadaire

15h02

Mensuel

Tailwind CSS :

Pour l'aspect visuel du projet, le prestataire a choisi Tailwind CSS. Ce framework CSS offre une approche unique en permettant la création d'interfaces utilisateur attrayantes et réactives en utilisant des classes utilitaires.

- **Personnalisation:** Tailwind est personnalisable, les développeurs peuvent l'ajuster facilement selon les besoins spécifiques.
- **Développement rapide :** Tailwind CSS permet de créer des interfaces rapidement en utilisant des classes permettant d'accélérer le développement.
- **Maintenabilité accrue :** Son approche modulaire et l'indépendance des classes rendent le code plus lisible et facile à entretenir.
- **Pas de surcharge inutile :** Tailwind génère uniquement les styles, évitant ainsi une surcharge inutile des fichiers de style.

Exemple d'utilisation :

```
<div className="bg-[#131B37] min-h-screen flex flex-col justify-center items-center">
```

E) Implémentation des mécanismes de sécurité standard

Le prestataire a favorisé la mise en place de mécanismes de sécurité standard pour assurer la protection des données et renforcer la fiabilité de l'application.

- **Certificat SSL/TLS :** Le certificat garantit la sécurité en chiffrant les communications entre la partie client et serveur. L'équipe de développement a utilisé les certificats proposés par Let's Encrypt. Il s'agit d'une autorité de certification gratuite, automatisée et ouverte. La base de données de Bastion utilise un certificat SSL paramétré dans le .env du projet.

```
DATABASE_URL="mysql://${DB_USER}:${DB_PASSWORD}@${DB_HOST}:${DB_PORT}/${DB_NAME} sslcert=DigiCertGlobalRootCA.crt.pem"
```

- **Gestion de l'authentification grâce à des jetons :** (JWT) Le jeton permet aux utilisateurs de rester connectés à une application en stockant le jeton dans le sessionStorage mis à disposition par les navigateurs.
- **Gestion des entrées utilisateurs :** via l'interface utilisateur pour éviter les injections de code **XSS** (Cross-site scripting) en encodant les données avant d'être affiché aux utilisateurs.

CHAPITRE 3 : DÉVELOPPEMENT BACK-END

A) Choix des technologies back-end et des bases de données

Le back-end de l'application a été développé avec **Node.js**, un environnement d'exécution JavaScript côté serveur qui permet de construire des applications robustes et évolutives.

Node.js est basé sur le moteur JavaScript V8 de Google Chrome et il offre un modèle de programmation asynchrone qui le rend particulièrement adapté aux applications nécessitant une manipulation efficace des entrées/sorties. Cela permet au serveur de gérer simultanément un grand nombre de connexions, assurant ainsi des performances élevées et une réactivité accrue.

Dans le développement de l'application, le prestataire choisit d'utiliser **Node.js** pour plusieurs raisons essentielles. Voici les principaux points qui expliquent la préférence du prestataire pour cette technologie :

- **Utilisation uniforme de JavaScript** : Un facteur clé de notre choix est la possibilité d'utiliser JavaScript de manière cohérente à la fois côté client et côté serveur.
- **Performances** : Node.js, basé sur le moteur V8 de Google, offre des performances avantageuses. Sa capacité à gérer efficacement de nombreuses connexions simultanées en fait une solution idéale pour les applications en temps réel, répondant à nos besoins opérationnels.
- **Écosystème npm dynamique** : Cette bibliothèque étendue de paquets, simplifie considérablement l'intégration de fonctionnalités tierces dans le projet.
- **Communauté active** : Node.js bénéficie d'une communauté active, offrant de nombreuses ressources en ligne, des forums de discussion et des bibliothèques open source. Cette assistance communautaire est importante dans notre développement, dans le débogage et dans les résolutions de problèmes.
- **Polyvalence** : Node.js se démarque par sa polyvalence, il est adapté à une variété d'applications.

MySQL a été choisi comme Système de Gestion de Base de Données (SGBD) car :

- **Fiabilité et durabilité** : Il fait partie des SGBD les plus utilisés au monde ainsi sa stabilité est reconnue pour n'importe quelle ampleur de projet.
- **Support** : Une grande communauté de développeurs utilise cet outil, Ainsi quand les développeurs rencontrent des problèmes, ils peuvent accéder à des forums pour chercher un moyen de résoudre celui-ci. Également, l'écosystème de MySQL propose une grande variété de plugins et de documentations.

- **Intégration facile** : il s'intègre facilement à tous les types de projet quelque soit la technologie utilisée. Il facilite le développement et le déploiement de ce dernier.
- **Sécurité** : MySQL inclut une gestion robuste des utilisateurs, le chiffrement des données, et des outils de contrôle d'accès et d'audit, garantissant ainsi la confidentialité et l'intégrité des données stockées.

Prisma a été utilisé comme **ORM** (Object-Relational Mapping) pour simplifier l'interaction avec la base de données.

Un **ORM** est un type de programme informatique qui permet de simplifier l'interaction entre une application et une base de données relationnelle en utilisant des objets de programmation pour représenter les données, empêchant d'écrire des requêtes SQL directes.

De plus, **Prisma** contribue à l'uniformisation du langage en permettant l'utilisation d'une syntaxe cohérente dans le langage de programmation (TypeScript ou JavaScript)

B) Stratégies de sécurité en profondeur : filtrage des données, authentification, journalisation, contrôle d'accès

La conception et la mise en œuvre de la couche de persistance des données a une importance cruciale pour pouvoir tracer les informations de l'application. Également, elle a une importance pour la sécurité des données qui y sont intégrées.

Filtrage des données entrantes et sortantes :

- Filtrage des données entrantes : L'outil Express Validator permet de vérifier la validité de certaines données en fonction du paramétrage, il permet aussi de "trimmer" et échapper les caractères problématiques pour faire face au XSS
- Filtrage des données sortantes : l'ORM Prisma assure que les données qui sont extraites sont vérifiées et formatées avant de l'afficher aux utilisateurs. Cette solution permet de valider que les données répondent aux spécifications de sécurité.

Le système de filtrage est utilisé dans un middleware avant que la requête atteigne la route. Ce middleware se charge de vérifier les paramètres de la requête (comme des filtres, des autorisations, etc.), et d'appliquer un filtrage si nécessaire.

Exemple du code de Express validator pour la modification d'une absence :

```
const checkUpdateAbsenceData = [
    body('start_date').optional().trim().escape().isDate().toDate().withMessage('La date n\'est pas au bon format'),
    body('end_date').optional().trim().escape().isDate().toDate().withMessage('La date n\'est pas au bon format'),
    body('user_id').optional().trim().escape().isNumeric().withMessage('L\'id utilisateur doit être un nombre entier').toInt(),
    body('reason').optional().trim(),
    body('statut').optional().trim(),
];

```

Journalisation des actions en base de données :

- Suivi des modifications : Toutes les tables de la base de données contiennent les colonnes **created_at**, **modified_at** et **deleted_at** qui enregistrent la date et l'heure de la l'action. Il y a également une colonne **is_active** qui permet de voir si l'enregistrement est actif ou non, permettant de suivre l'état des données sans la supprimer définitivement.
- Suivi des actions utilisateur : Afin de suivre les actions des utilisateurs, les colonnes **created_by**, **modified_by** et **deleted_by** permettent d'identifier la personne qui réalise une action, assurant sécurité et responsabilité au sein du système.
- Table logs : Afin de suivre toutes les actions de l'utilisateur, une table logs a été créée dans la base de données de Bastion

Exemple des journalisations :

created_at	modified_at	deleted_at	created_by	modified_by	deleted_by
2024-07-15 11:43:21	2024-07-15 13:43:20.798	NULL	10 johndoe	NULL	NULL

Exemple d'une log de la table logs :

id	content	fk_user	created_at	modified_at
31	Un nouvel utilisateur jdoe a été créé par un admin	10	2024-08-26 16:50:09	2024-08-26 1

C) Mise en place de l'authentification et l'autorisation solides

Pour sécuriser l'accès aux applications, le prestataire a développé un espace de connexion sécurisé permettant d'accéder aux données. De plus, l'authentification permet à la personne connectée de consulter toutes ses informations. Le **RGPD** oblige qu'un espace de connexion soit mis en place afin d'accéder à ses données personnelles.

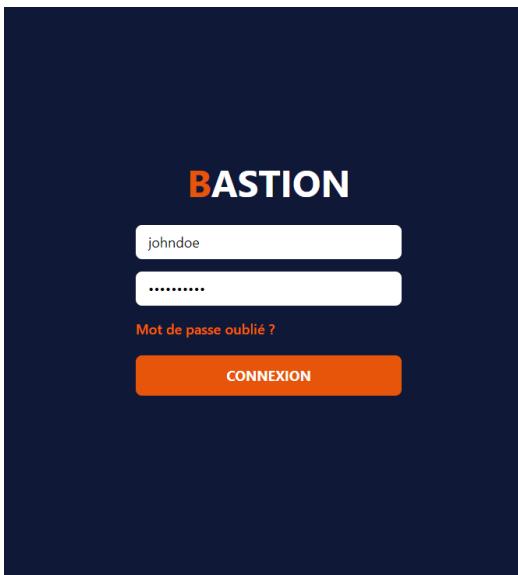
Pour se faire, le prestataire a implémenté deux mécanismes :

- Authentification :

- Génération des tokens : Dès qu'un utilisateur se connecte aux applications, il génère un token signé avec une clé secrète. Le token inclut toutes les informations essentielles d'une personne comme un identifiant, une adresse email, etc...
- Stockage des tokens : Une fois le token généré, il est stocké dans le sessionStorage pour le web et le AsyncStorage dans le mobile pour l'utiliser dans toutes les requêtes futures afin de vérifier l'authenticité de l'utilisateur.
- Validation des tokens : À chaque requête, le serveur vérifie le token en le décodant pour s'assurer que le token n'a pas expiré ou été dégradé. Cela permet d'afficher les ressources seulement aux personnes authentifiées.

- **Autorisation :**

- Middleware : Création de middleware pour vérifier l'authentification et l'autorisation d'accès selon les rôles de la personne.
- Rôle utilisateur : Création de rôle pour déterminer le niveau d'accès d'une personne (Utilisateur, Administrateur...).



Pour générer un **JWT token**, l'utilisateur doit rentrer ses identifiants de connexion dans l'interface web/mobile, si la connexion est un succès, il stocke le token dans le **sessionStorage** du navigateur et redirige l'utilisateur vers la page d'accueil.

Exemple de stockage du token dans le Session storage :

Key	Value
token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjEwLCJ1c2VyTG9naW4iOiJqb2huZG9lIiwidXNlcIjvbGUiOiJBZG...

D) Chiffrement des données sensibles et personnelles

Pour renforcer le niveau de sécurité des données, dès que le système insère une donnée sensible et/ou personnelle dans la base de données, il chiffre celle-ci pour éviter toute tentative d'usurpation de la donnée.

Chiffrement avec bcrypt :

- Hachage du mot de passe : Cela permet de ne pas stocker les mots de passe en clair dans la base mais uniquement de stocker une empreinte de ces derniers. Bcrypt ajoute également un "**sel**" unique à chaque mot de passe avant de le hacher pour éviter que plusieurs mots de passes identiques correspondent au même salage ainsi éviter les attaques par "Table Arc-En-Ciel" (Rainbow Table Attack).

password

\$2b\$10\$4silUR46XE7RmQqqgTfAme8436guAom5yh1demLMvcJ...

- Chiffrement des données : Lors de l'insertion d'une donnée, le système chiffre cette donnée pour maintenir la confidentialité en la rendant illisible pour toute personne non autorisée. Si le token est valide, alors les données auxquelles l'utilisateur a accès peuvent être déchiffrées. Le RGPD oblige que les données personnelles soient protégées contre tout accès et divulgation non autorisé.

lastname

firstname

0e9f3a6bff6086f7b1c473bb02979c30:c8316a2271d8b28f6... 9f16c1e99eac368d8d8d280fb8e74296:830bde0b1fb92e89c...

E) Intégration des bonnes pratiques d'écoconception pour minimiser l'impact écologique

Pour minimiser l'impact écologique dans le cadre du développement back-end, les développeurs ont adoptés quelques pratiques écoresponsable :

- **ORM Prisma** : Afin de faciliter l'utilisation des requêtes SQL, Prisma permet d'optimiser les requêtes et de gérer les transactions à la base de données.
- **Sélection des données nécessaires** : Afin de réduire le poids des résultats en sélectionnant seulement les colonnes nécessaires.

F) Compatibilité et Conformité avec les Standards Actuels

L'équipe de développement a veillé à ce que le back-end respecte les standards actuels.

Pour se faire, ils ont vérifiés :

- **Encodage UTF-8** : Pour assurer la bonne prise en charge des caractères spéciaux.
- **Sécurisation du protocole HTTP(S)** : Cela permet de sécuriser les échanges entre le serveurs et les clients et d'assurer la compatibilité avec les différents navigateurs.

- **Utilisation d'une API RESTful** : Les développeurs ont procédé au développement d'une API RESTful se basant sur les méthodes HTTP standard (GET, POST, PUT, DELETE).
- **Standard de sécurité** : Données personnelles chiffrées, système d'authentification et d'autorisation, protection contre les attaques XSS et CSRF, filtrage et validation des entrées, journalisation des actions...

CHAPITRE 4 : DÉVELOPPEMENT ET CONSOMMATION D'API

A) Conception et développement de l'API

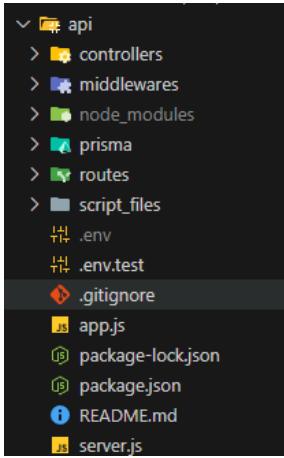
Dans le contexte du projet, le prestataire a développé une **API RESTful** se basant sur les principes de conception **REST** permettant d'obtenir une structure claire entre le front-end et le back-end.

Avec une **API RESTful**, l'interface uniforme sépare le client et le serveur, permettant ainsi aux clients de communiquer simultanément avec différents serveurs. Grâce à cet aspect, les développeurs peuvent développer presque tout ce qu'ils veulent et peuvent profiter de la flexibilité et de l'évolutivité de l'API.

Pour se faire l'API a été réalisé avec le framework **Express.js** pour Node.js afin de fournir un accès efficace et sécurisé aux données des applications.

- **Utilisation d'endpoints** : Plusieurs routes ont été mises en place pour accéder aux données spécifiques en utilisant des méthodes HTTP standard comme le GET, POST, PUT et DELETE.
- **Gestion des erreurs** : Un système de gestion des erreurs a été implémenté permettant de retourner des erreurs cohérente en cas d'échecs ou d'entrées invalide. Cette gestion permet de ne pas afficher le détail interne des erreurs.
- **Structuration de l'API** : L'architecture de l'API a été pensée de manière à séparer clairement toutes les fonctions, tous les contrôleurs, et toutes les routes pour chaque entité ou service. C'est une approche plus simple qui permet de s'organiser et d'assurer facilement la maintenabilité de l'API.

Structuration de l'API de Bastion :



B) Sécurisation de l'accès à l'API et mécanismes d'authentification

Afin de protéger les données personnelles et pour vérifier que seules les personnes autorisées peuvent accéder aux ressources, des mesures de sécurité ont été mises en place :

- **Authentification JWT** : Tous les utilisateurs doivent s'authentifier pour obtenir un token JWT. Une vérification de l'utilisateur est donc réalisée pour chaque interaction qu'il fera sur l'API.
- **Chiffrement des communications** : Toutes les communications sont chiffrées entre les clients et le serveur grâce au protocole HTTPS qui utilise les protocoles SSL/TLS pour protéger les communications.
- **Système de permission** : Un système de rôle et de groupe de rôle a été mis en place pour limiter l'accès à certaines routes.

```
router.get('/logs', authToken, authRoleUser(ALL_ACCESS_GROUP), logController.allLogs);
router.get('/log/:id', authToken, authRoleUser(ALL_ACCESS_GROUP), logController.logById);
router.post('/log', authToken, authRoleUser(ADMIN_GROUP), logController.addLog);
router.put('/log/:id', authToken, authRoleUser(ADMIN_GROUP), logController.updateLog);
router.delete('/log/:id', authToken, authRoleUser(ADMIN_GROUP), logController.deleteLog);
```

```
# Paramétrage
READ_ONLY_HIS_DATA_GROUP=Utilisateur # Groupe de type user dont on ne veux pas
MODIFY_ONLY_HIS_DATA_GROUP=Utilisateur,Chef équipe # Groupe de type user dont
les siennes

RESTRICTED_GROUP=Utilisateur,Chef équipe,Assistant-RH,RH,DRH,Administrateur
MANAGER_GROUP=Chef équipe,Assistant-RH,RH,DRH,Administrateur
RESTRICTED_RH_GROUP=Assistant-RH,RH,DRH,Administrateur
RH_GROUP=RH,DRH,Administrateur
ALL_ACCESS_GROUP=DRH,Administrateur
ADMIN_GROUP=Administrateur
SUPERADMIN_GROUP=
```

C) Documentation de l'API

Le prestataire a utilisé Swagger car il permet directement de documenter les routes, et de produire des ressources indépendamment du langage de programmation utilisé.

- **Endpoints clairs** : Chaque route de l'API est documentée avec une description des actions qu'elle permet : les lectures, écritures, suppressions...
- **Paramètres d'entrée et de sortie** : La documentation détaille les paramètres attendus, qu'ils soient dans le corps de la requête, l'URL ou les en-têtes.
- **Codes de retour** : Les différents codes de statut HTTP renvoyés comme 200, 301, 400, 401, 404, 500.

Exemple

d'endpoint:

DELETE /users/{id} Supprime un utilisateur par ID 🔒 ↴

Exemple de paramètre :

Parameters		Try it out
Name	Description	
id * required integer (path)	ID de l'utilisateur <input type="text" value="id"/>	

Exemple de réponse :

Responses		
Code	Description	Links
200	Utilisateur supprimé	No links
404	Aucun utilisateur trouvé	No links
500	Erreur interne sur le serveur API	No links

D) Processus de documentation et publication de l'API pour les développeurs

Pour faciliter la charge de travail, le prestataire a mis en place un processus et une publication clair pour la documentation :

- **Publication** : La documentation est accessible via une interface web pour que les développeurs puissent visualiser et interagir avec les endpoints de l'API. Ils peuvent directement tester les requêtes depuis cette page.
- **Exemples** : La documentation fournit des exemples de requêtes et de réponses pour les principaux cas d'utilisation de l'API, facilitant ainsi l'intégration et le développement.

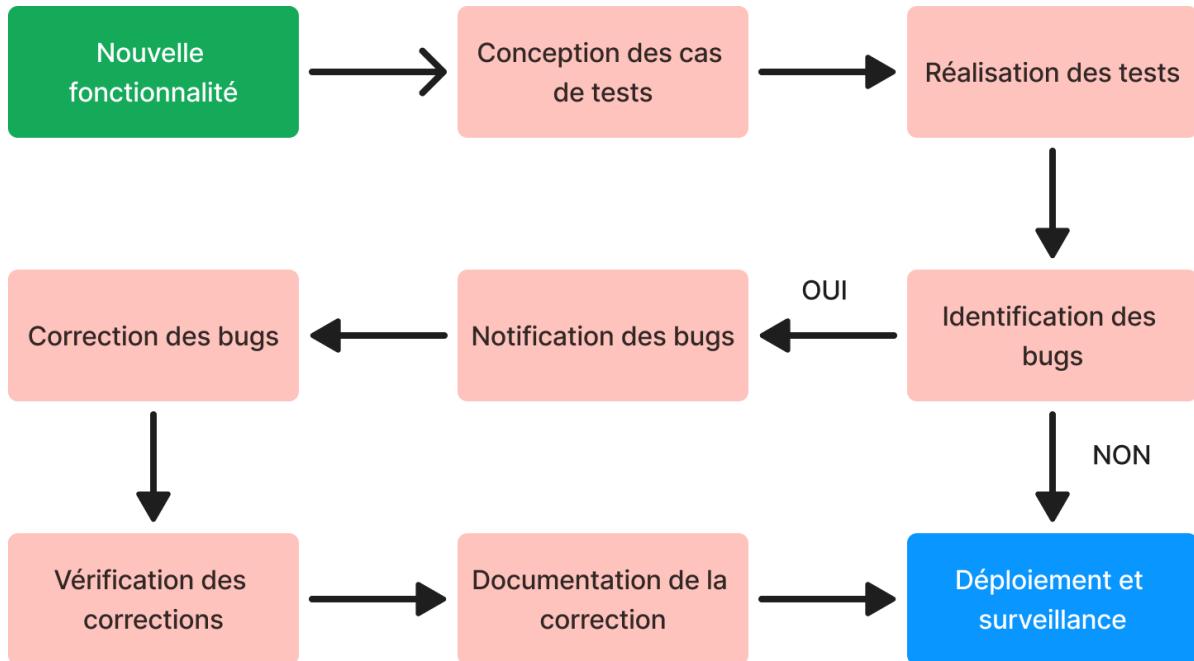
Exemple de l'interface l'API de Bastion :

The screenshot shows the Swagger API Documentation interface for a Bastion API. At the top, it says "API Documentation 1.0.0 OAS 3.0". Below that, there's a "Servers" dropdown set to "http://localhost:3001". The main area shows the "Users" section under "Gestion des utilisateurs". It includes sections for "Auth", "User", and "Schemas". Under "User", there are several API endpoints listed with their methods, URLs, and descriptions:

- Auth**: POST /login - Authentifie un utilisateur et retourne un token JWT
- User**: GET /users - Récupère tous les utilisateurs
- User**: POST /users - Ajoute un nouvel utilisateur
- User**: GET /users/{id} - Récupère un utilisateur par ID
- User**: PUT /users/{id} - Met à jour un utilisateur existant
- User**: DELETE /users/{id} - Supprime un utilisateur par ID

CHAPITRE 5 : STRATÉGIE DE TESTS, ASSURANCE QUALITÉ

A) Élaboration d'un plan de test exhaustif et structuré



Processus de tests :

- **Nouvelle fonctionnalité** : À partir du moment où les développeurs terminent une nouvelle fonctionnalité, ils démarrent le processus de test.
- **Conception des cas de tests** : Les développeurs préparent plusieurs scénarios sur toutes les actions qu'un utilisateur peut effectuer.
- **Réalisation des tests** : À partir des scénarios élaborés lors de l'étape précédente, les développeurs effectuent les différents tests puis effectue les tests sur un échantillon de personnes n'ayant pas travaillé sur l'application (Beta Testing et Pentesting).
- **Identification des bugs** : Lors de la réalisation des tests, des anomalies ont-elles été identifiées ?
 - **OUI** : Prochaine étape : Notification des bugs.
 - **NON** : Prochaine étape : Déploiement et surveillance.
- **Notification des bugs** : Rédaction d'un document qui précise l'existence d'un problème.
- **Correction des bugs** : L'équipe de développeurs corrige les bugs.
- **Vérification des corrections** : Validation des tests passés avec succès sans que cela ne cause d'effets de bords.

- **Documentation de la correction** : Rédaction d'un rapport sur le problème notifié précédemment et la façon dont il a été corrigé.
- **Déploiement et surveillance** : Mise à jour du code avec la nouvelle fonctionnalité et surveillance pour détecter tout autre problème.

B) Méthodologie de tests (unitaires, d'intégration, etc.) et couverture du code

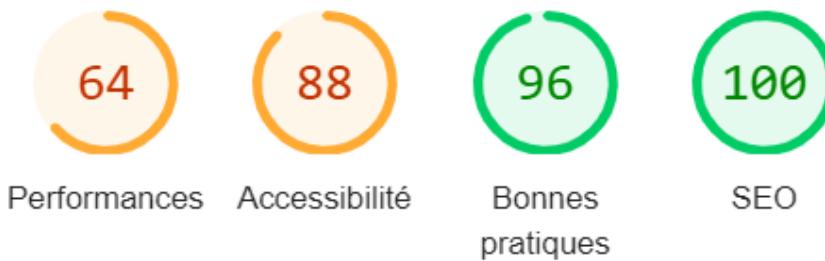
Méthodologie de tests :

Pour justifier la compatibilité des applications et leur bon fonctionnement, le prestataire a réalisé plusieurs tests pour fournir une application de bonne qualité.

- **Test de compatibilité multi-navigateurs** : afin de vérifier que les applications soient fonctionnelles sur web suivant les navigateurs (Firefox, Google...) et mobile.
- **Test de responsivité et réactivité** : afin de s'assurer que l'affichage des applications s'adapte à la taille des écrans tout en gardant une bonne disposition des éléments.
- **Test fonctionnel suivant les scénarios utilisateurs** : se placer du point de vue des utilisateurs permet de s'assurer que les outils sont utilisés de manière intuitive et que les fonctionnalités répondent à leurs besoins réels.
- **Test d'accessibilité** : afin de s'assurer que les applications sont convenables aux personnes ayant des besoins particuliers.
- **Test de régression** : afin de vérifier que toutes les modifications ou les ajouts n'ont pas engendré d'autres problèmes sur les applications.
- **Test de conformité au RGPD** : afin de garantir que toutes les fonctionnalités respectent la charte informatique relative au RGPD.

Couverture du code Industrialisation du Développement :

- **Couverture des tests unitaires** : le prestataire a utilisé la librairie Jest. Elle permet d'écrire et d'exécuter des tests unitaires et d'intégration.
- **Analyse statique** : des extensions comme **ESLint** et **SNYK** ont été utilisées pour réaliser des analyses statiques du code afin de détecter les potentielles erreurs ou surveiller les performances de l'application et de pouvoir les corriger rapidement et facilement. Le prestataire a utilisé **Lighthouse** pour analyser les performances de la page web. Il a généré un rapport détaillé avec des indicateurs sur la performance, l'accessibilité, les bonnes pratiques et le SEO

Moyenne de toutes les pages de Bastion :

C) Intégration continue et Déploiement continu

L'intégration continue et le déploiement continu permettent d'optimiser et d'automatiser le déploiement des nouvelles fonctionnalités efficacement.

Pour permettre l'optimisation et l'automatisation, le prestataire a utilisé plusieurs outils :

- **Gestion des versions** : Afin de suivre toutes les versions qui ont été déployées sur le projet, l'équipe de développement a décidé d'utiliser Github pour gérer les versions du projet, les branches, et de pouvoir revenir à des versions antérieures. (cf Organisation GitHub)
- **Intégration Continue (CI)** : Création d'une pipeline GitHub Actions automatisant le processus de tests, déclenchée à chaque commit d'un développeur afin de vérifier la présence d'éventuelles erreurs.
- **Déploiement Continu (CD)** : Mise en place de l'intégration continue qui, après validation des tests, déclenche automatiquement le déploiement en production de la nouvelle fonctionnalité, pris en charge par notre hébergeur.

D) Automatisation des tests et gestion des dépendances

Pour assurer une stabilité de l'application tout au long du projet, le prestataire a automatisé les tests et a géré efficacement toutes les dépendances nécessaires au bon fonctionnement du projet.

- **Gestion des dépendances** : Utilisation du gestionnaire de paquet NPM qui a permis de suivre toutes les versions des bibliothèques importées dans le projet.
- **Automatisation des tests** : Un script a été mis en place pour automatiser les tests Jest directement dans Github Action, assurant ainsi que les tests unitaires soient exécutés automatiquement à chaque commit et pull request afin de vérifier qu'il n'y ait aucune régression.

E) Optimisation de la chaîne de build et amélioration des performances

Un workflow d'intégration Github a été mis en place. Lorsque qu'un push sur la branche désignée est effectué, les tests unitaires sont lancés.

Pendant l'exécution du workflow, un environnement de test est créé avec une base de données temporaire en plus d'une instance NodeJs. Le workflow gère les dépendances à installer avec NPM. Tout cela permet de simuler un environnement d'exécution propre afin que les tests unitaires puissent s'exécuter.

All workflows

Showing runs from all workflows

Filter workflow runs

Help us improve GitHub Actions

Tell us how to make GitHub Actions work better for you with three quick questions.

Give feedback

34 workflow runs

Event Status Branch Actor

Workflow	Status	Time	...
resetPassword+notifyUser	preprod-unit-tests	16 hours ago	...
fix-auth-token-41209	preprod-unit-tests	yesterday	...

CI #29: Commit b329e5c pushed by Oozak-ACY

CI #28: Commit 486e3a7 pushed by Oozak-ACY

CHAPITRE 6 : DÉVELOPPEMENT HARDWARE ET HÉBERGEMENT

A) Développement Lecteur de badge

Le matériel utilisé pour construire le lecteur de badge est l'écosystème Raspberry et Arduino. (*cf. Annexe Lecteur de Badge*)

Ainsi, dans un lecteur de badge Bastion, une raspberry pi 4 sous l'OS Raspbian Bookworm est utilisée, en complément, une caméra 4k est directement branchée sur cette raspberry afin de détecter les qrcodes qui lui sont présentés. Afin de détecter les badges RFID, une carte Arduino UNO avec un capteur RFID, est branchée en série à la raspberry, grâce à cela, le module arduino pourra communiquer le unique ID du badge capté, à la raspberry.

Le lecteur de badge utilise 2 scripts distincts pour les différentes méthodes d'accès, tous deux utilisant python. Ces scripts feront des appels à l'api sous conditions, celle-ci déterminera si l'utilisateur peut rentrer ou non. Dans les 2 cas, le script python créera une log locale pour retracer le déroulement des évènements depuis le lecteur, et créera une log en base de donnée via l'api bastion.

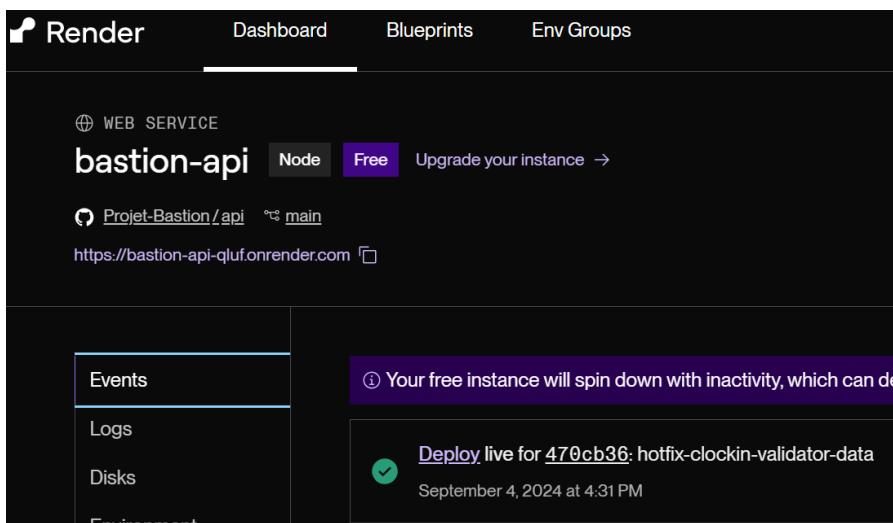
Ainsi un système de journalisation local et base de données sera disponible.

B) Déploiement et hébergement

La base de données est déployée et hébergée par Azure de Microsoft qui est doté d'un gros service support, et qui permet une bonne scalabilité de la bdd. Cette scalabilité permet une bonne gestion du coût de l'hébergement en fonction de la demande pour l'application Bastion.

L'api quant à elle est déployée sur l'hébergeur **Render** qui permet une bonne scalabilité avec des packs d'abonnements qui augmentent donc la capacité de l'api.

Render intègre un système de déploiement automatique paramétrable par l'utilisateur. Pour Bastion un déploiement est lancé à chaque modifications effectuées sur la branche main du github.



The screenshot shows the Render.com interface for a 'bastion-api' project. It's a 'WEB SERVICE' type, running on a 'Node' environment and is 'Free'. The instance is currently active on the 'main' branch, with a URL provided: <https://bastion-api-qluf.onrender.com>. A deployment log entry from September 4, 2024, at 4:31 PM, indicates a successful deployment for commit [470cb36](#): hotfix-clockin-validator-data.

CHAPITRE 7 : OPTIMISATION POUR LE SEO ET MARKETING DIGITAL

Le référencement ne faisait pas partie de nos prérogatives pour développer le projet. Néanmoins, si le client nous demande de mettre en ligne l'application, le prestataire a prévu une façon d'optimiser le SEO et comment il a référencé le produit. (*cf. Page référencement SEO*)

C) Stratégies de Référencement Naturel

Dans l'objectif d'améliorer la visibilité des applications, le prestataire a adopté des stratégies de référencement naturel qui indiquent un bon positionnement dans les résultats de recherche et améliore la notoriété.

D) Mise en œuvre des meilleures pratiques pour le SEO

Différentes pratiques peuvent être mises en œuvre pour améliorer le référencement de l'outil :

- **Utilisation et optimisation des balises HTML** : Toutes les pages de l'application web contiennent des balises titres **<title>**, des balises de description **<meta>** ainsi que des balises d'en tête **<h1>**, **<h2>**... Cela permet au navigateur de comprendre le contenu des pages pour le classer de la meilleure façon.
- **Amélioration des URLs** : Pour rendre lisibles le système de navigation, l'équipe de développeur a rendu les URLs lisibles et optimisés (ex: bastion.fr/admin/company)

E) Mesure de Performance Marketing

Pour pouvoir mesurer l'efficacité du référencement, le prestataire pourrait utiliser des outils d'analyse et de suivi de performance pour favoriser l'optimisation et la compréhension du comportement des utilisateurs.

Google Analytics est un moyen pour suivre le trafic, analyser les actions utilisateurs, savoir quelle page est à optimiser, connaître les sources utilisées permettant de savoir de quel endroit proviennent les utilisateurs comme :

- **Organique** : Résultat des moteurs de recherche.
- **Social** : Des liens partagés en provenance des réseaux sociaux.
- **Direct** : Les utilisateurs écrivent directement l'URL du site web.
- **Payant** : Les visiteurs cliquent sur des publicités payantes pour accéder au site.

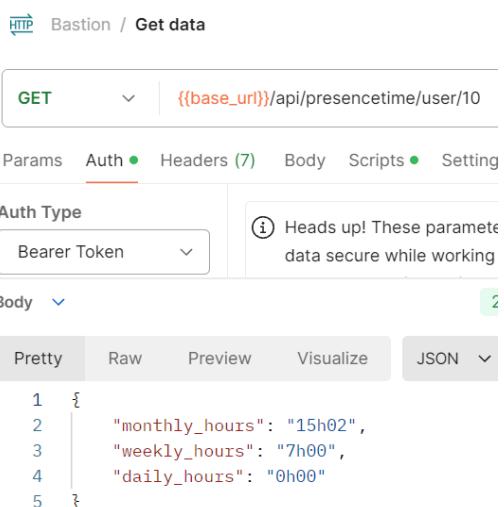
CHAPITRE 8 : FONCTIONNALITÉS IMPORTANTES

A) Calcul du temps de travail (journalier / hebdomadaire / mensuel).

Afin de fournir aux utilisateurs et aux ressources humaines le calcul du temps de travail du personnel, le prestataire a procédé à plusieurs étapes.

Dans L'API, les développeurs ont créé une route spéciale pour calculer le temps de travail des utilisateurs. A partir de l'identifiant de l'utilisateur connecté, l'API récupère cet ID permettant de savoir sur quel utilisateur le script de calcul doit s'exécuter. Le calcul retourne le résultat.

Exemple du résultat de la route API :



The screenshot shows the Bastion API testing interface. The URL is set to `GET {{base_url}}/api/presencetime/user/10`. The Headers tab is selected, showing an `Auth Type` of `Bearer Token`. The response is displayed in JSON format:

```
1 {  
2   "monthly_hours": "15h02",  
3   "weekly_hours": "7h00",  
4   "daily_hours": "0h00"  
5 }
```

Pour centraliser le temps de travail dans l'application, la page permettant de visualiser les données est la page d'accueil car c'est la première page auquel les utilisateurs accèdent.
(*cf Annexe : Page d'accueil de l'application web*)

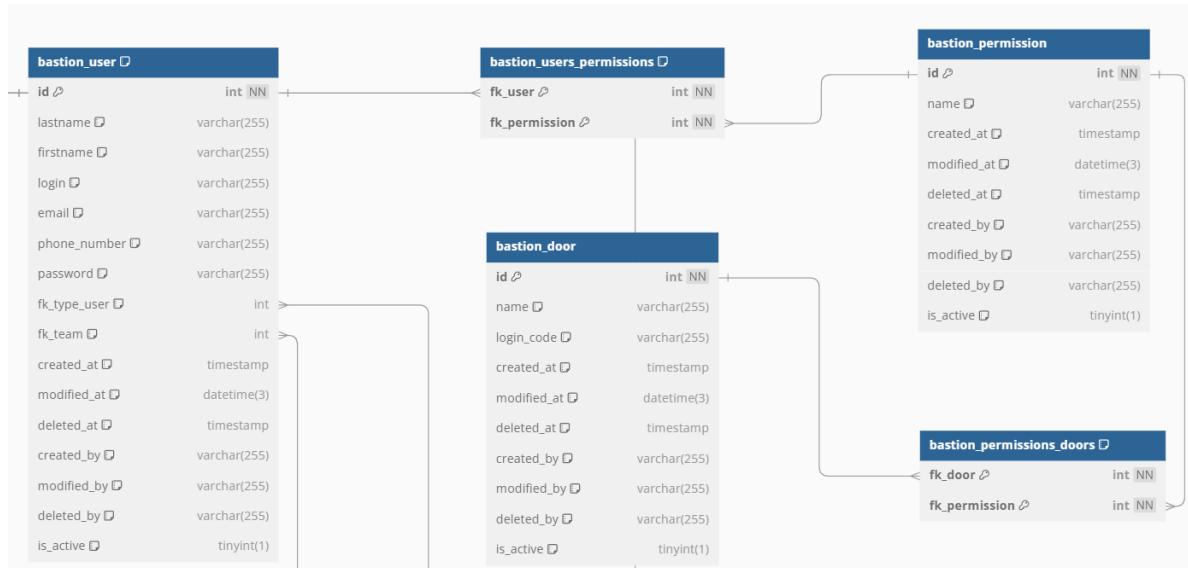
Temps de travail



B) Gérer les autorisations d'accès.

Un système de gestion des permissions a été implémenté permettant de déterminer quel rôle a l'utilisateur ainsi que les portes auxquelles celui-ci a accès. Suivant le rôle, l'utilisateur obtient différents d'accès dans les applications qui déterminent les ressources et données auxquelles il aura accès. Et suivant les permissions acquis, l'utilisateur pourra accéder à certaines portes.

Les développeurs ont fait en sorte que les portes (matérialisées par des lecteurs de badges) soient rattachées à des permissions, et les utilisateurs sont rattachées à certaines permissions.



Les types utilisateur ont comme paramètres les jours et heures auxquels ils sont autorisés à être dans l'entreprise peu importe la situation.

```
"id": 13,
"name": "Assistant-RH",
"start_hour": 6,
"end_hour": 20,
"presence_days": "lundi,mardi,mercredi,jeudi,vendredi",
"active": true
```

Ici, les utilisateurs de type “Assistant-RH” sont autorisés à entrer dans le bâtiment à partir de 6h du matin et quitter celui-ci à 20h. Et ce, du Lundi au Vendredi. En dehors de ces jours et heures l'accès leur sera refusé pour des raisons de sécurité.

Attention ceci n'est pas lié aux emplois du temps des utilisateurs, cela représente les heures “tolérées” d'accès.

The screenshot shows the BASTION application interface. On the left, there's a sidebar with various menu items: Dashboard, Entreprises, Utilisateurs, Permissions & Portes, Badges, Qrcodes, Assurances, Absences, and Logs. At the bottom of the sidebar is a 'Déconnexion' button. The main area has a header 'BASTION' and a user dropdown 'john doe'. Below the header, there are three sections: 'Édition de Ikeovilay' (with fields for Prénom: Loan, Nom de Famille: Keovilay, Email: loan.keovilay@outlook.com, Téléphone: 0666554432, Rôle: Entreprise, and Entreprise: Simple Company), 'Configuration des badges' (with a dropdown 'Sélectionner un badge' and an 'Enregistrer' button), and 'Configuration des permissions' (with checkboxes for Accès porte 1, Accès deuxième étage, and Accès premier étage). A modal window titled 'Édition de Ikeovilay' is open over the configuration sections. It contains fields for Prénom (Loan), Nom de Famille (Keovilay), Email (loan.keovilay@outlook.com), Téléphone (0666554432), and Rôle (Entreprise). Below these fields is a dropdown menu titled 'Sélectionner un rôle' with the current role 'Administrateur' highlighted. A red box surrounds this dropdown menu. At the bottom of the modal are 'Enregistrer' and 'Annuler' buttons.

C) Le système de “badgeage” (badge RFID, application mobile).

Pour sécuriser les accès, un lecteur de badge a été positionné au niveau de chaque porte.

Ce lecteur de badge peut être utilisé avec 3 outils :

- **Badge** : Un badge physique peut être affecté à un utilisateur disposant des droits d'entrée dans le bâtiment.

The screenshot shows the Bastion application's badge management page. The left sidebar is titled "PANEL GESTION" and includes links for Dashboard, Entreprises, Utilisateurs, Permissions & Portes, Badges, Qrcodes, Assurances, Absences, and Logs. The main content area is titled "Gestion des badges". It features a search bar "Rechercher par code de b...", a button "Nouveau Badge", and an orange button "Ajouter un badge". A table lists three badges:

ID	CODE	EMPLOYÉS	ACTIONS
13	KIRIKOU	X	<button>Modifier</button> <button>Supprimer</button>
12	FNZAIONGOZIA	fabrice tesson	<button>Modifier</button> <button>Supprimer</button>
11	ABCD1234	john doe	<button>Modifier</button> <button>Supprimer</button>

The screenshot shows the Bastion application's user management page. The left sidebar is titled "PANEL GESTION" and includes links for Dashboard, Entreprises, Utilisateurs, Permissions & Portes, Badges, Qrcodes, Assurances, Absences, and Logs. The main content area is titled "Gestion des utilisateurs". It features three sections: "Édition de solo" (with fields for Prénom, Nom de Famille, Email, Téléphone, Rôle, Entreprise, and Équipe), "Configuration des badges" (with a dropdown "Sélectionner un badge" and a blue button "Enregistrer"), and "Configuration des permissions" (with checkboxes for Accès porte 1, Accès deuxième étage, and Accès premier étage). At the bottom are "Enregistrer" and "Annuler" buttons.

- **Application Mobile :** Le téléphone est l'outil le plus utilisé, l'application mobile permet aussi d'être utilisé comme badge d'accès. Pour se faire, un système de génération des QR Codes a été réalisé. Ce QR code a une durée de vie de 1 minute à partir du moment où le QR Code a été généré. Dès que le qr code est utilisé, il est automatiquement désactivé (dans la base de données, la colonne **is_active** passe de 1 à 0). Puis, il en re génère un nouveau. (cf. Annexe : *Page de génération du QR Code sur l'application mobile*)

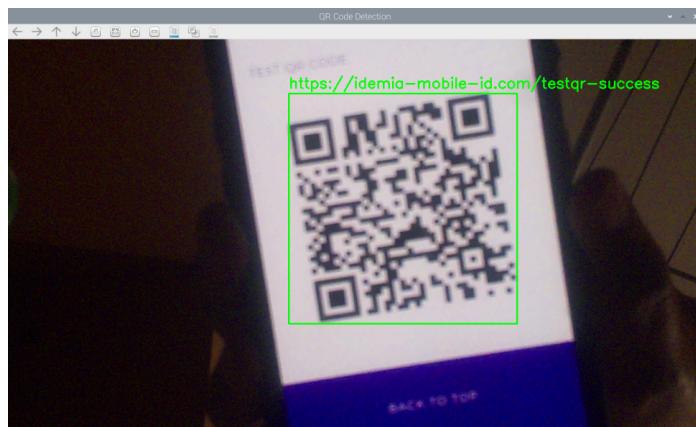
Lorsqu'un agent essaye de badger sa carte ou son QR code pour pouvoir entrer dans un bâtiment hors heures ouvrables, l'API va vérifier les heures et les jours de travail de la personne, ses absences (si elle est en congé ou non) et ses permissions (si elle a accès à une porte ou non).

Un log est ainsi créé en bdd si l'utilisateur est autorisé ou non à entrer

13	johndoe a tenté d'accéder au bâtiment sans autorisations....	john doe	2024-09-06 02:14:08	Supprimer
----	--	----------	---------------------	---------------------------

Deux programmes python tournent en permanence sur les lecteurs d'accès. Un fichier d'environnement permet de paramétrier le code de la porte pour pouvoir l'identifier.

- Pour détecter les qrcodes, un programme utilise les librairies picamera2 et opencv pour le traitement d'image en temps réel, ainsi que pyzbar qui est un OCR qui permet la reconnaissance et la traduction des qrcodes.



- Pour détecter les badges, un programme en C++ compilé sur la carte arduino connectée en série à la raspberry, permet de détecter le passage d'un badge rfid et le transmettre au programme python, conçu pour le badge rfid.

Une fois le code du badge ou qrcode récupéré et stocké, les deux programmes ont le même fonctionnement. Un appel api est effectué avec dans son corps : le code, la date de badgeage et le code de la porte. La route appelée, vérifie tout d'abord si ce code n'est pas expiré et est rattaché à un utilisateur. Ensuite, la route va vérifier si l'utilisateur a le droit d'accéder au bâtiment et à la porte au moment T, et, si l'utilisateur est absent à ce moment.

Si l'api autorise l'accès, une log locale est créée dans le lecteur d'accès et une log d'accès et créée dans la bdd. Une ligne clockin est aussi créée afin de retracer les entrées/sorties. Enfin si c'est un qrcode il est désactivé en base de données afin qu'il ne soit plus utilisable.

Si l'api n'autorise pas l'accès, une log locale est créée dans le lecteur d'accès et une log d'accès et créée dans la bdd.

D) Prévoir un système paramétrable d'export csv pour l'intégration dans un logiciel de RH.

Pour travailler de manière complémentaire avec les logiciels des Ressources Humaines comme Antibia, Eurecia... la fonctionnalité d'export des données a été réalisée pour pouvoir importer les données du logiciel de gestion du temps de travail dans les logiciels des Ressources Humaines.

Pour pouvoir créer cette fonctionnalité, les développeurs ont utilisé une librairie **react-csv**.
(cf. Annexe : Exemple de génération du CSV)

Lors de la consultation de la section “Gestion des utilisateur”, les RH peuvent exporter tous les utilisateurs affichés dans le tableau.

Gestion des utilisateurs



The screenshot shows a user management interface titled "Gestion des utilisateurs". At the top right, there are two buttons: "Créer un utilisateur" (Create user) in orange and "Tout Exporter" (Export all) in green with a CSV icon. A red arrow points to the "Tout Exporter" button. Below the buttons is a table with columns: EMPLOYÉS, RÔLE, EMAIL, TÉLÉPHONE, ENTREPRISE, and ACTIONS. The table contains three rows of user data with "Modifier" and "Supprimer" buttons in the ACTIONS column.

EMPLOYÉS	RÔLE	EMAIL	TÉLÉPHONE	ENTREPRISE	ACTIONS
solo solo	Assistant-RH	solo@bastion.fr	+33 0123456789	Non spécifié	<button>Modifier</button> <button>Supprimer</button>
fabrice tesson	DRH	tesson@bastion.fr	+33 0329171957	Example Company	<button>Modifier</button> <button>Supprimer</button>
john doe	Administrateur	johndoe@bastion.fr	+33 0123456789	Example Company	<button>Modifier</button> <button>Supprimer</button>

Voici le csv généré :

ID	Prénom	Nom	Email	Téléphone	Login	Entreprise	Équipe	Présence (Mois)	Présence (Heure)	Présence (Jours)
10	john	DOE	johndoe@bastion.fr	123456789	johndoe	Example Compa	Development	Te: 15h02	7h00	0h00
11	fabrice	TESSON	tesson@bastion.fr	3291719572	tesson	Example Compa	Development	Te: 4h00	4h00	0h00
12	solo	SOLO	solo@bastion.fr	123456789	solo	Example Compa	Development	Te: 0h00	0h00	0h00
13	Noé	ZIADI	noeziadi@outlook.com	777921473	nziadi	Example Compa	Development	Te: 0h00	0h00	0h00
14	Loan	KEOVILAY	loan keovilay@orange.fr	666554432	lkeovilay	Example Compa	Development	Te: 0h00	0h00	0h00

Lors de la modification d'un utilisateur, il est aussi possible d'exporter les informations de celui-ci :

Édition de Ikeovilay

Exporter

Prénom	Nom de Famille
Loan	Keovilay
Email	Téléphone
loan.keovilay@gmail.com	0666554432
Rôle	Entreprise
Sélectionner un rôle	Sélectionner une entrep ^e
Équipe	
Sélectionner une équipe	

Le csv généré :

ID	Prénom	Nom	Email	Téléphone	Login	Entreprise	Équipe	Rôle	Présence (Mois)	Présence (Heur	Présence (Jours /
10	john	DOE	noeziadi@outlook.com	777921473	johndoe	Example Compte	Development Team	Administrateur	15h02	7h00	0h00

E) Couverture du code source par des tests unitaires.

Des tests ont été mis en place avec la librairie JEST dans l'api afin de couvrir les routes du CRUD utilisateur qui est l'objet le plus complet, complexe et le point central de l'application Bastion.

Les tests sont exécutables en local lors du développement et sont aussi inclus dans le fichier de build de github action afin que ceux-ci soient lancés à chaque nouveau push et pull de la branche “main” et “preprod”.

```

PASS  tests/usersRoutes.test.js
User Routes
  POST /api/user
    ✓ Création d'un utilisateur valide (1066 ms)
    ✓ Échec de la création d'un utilisateur sans login (55 ms)
  PUT /api/user/:id
    ✓ Modification d'un utilisateur existant (21 ms)
    ✓ Échec de la modification d'un utilisateur avec des données invalides ()
    ✓ Échec de la modification d'un utilisateur avec un id inexistant (28 ms)
  DELETE /api/user/:id
    ✓ Suppression d'un utilisateur existant (12 ms)
    ✓ Échec de la suppression d'un utilisateur avec un id inexistant (13 ms)
  GET /api/user/:id
    ✓ Récupération d'un utilisateur existant (13 ms)
    ✓ Échec de la récupération d'un utilisateur avec un id inexistant (7 ms)
Routes security
  ✓ Accès d'un utilisateur drh autorisé pour une route (11 ms)
  ✓ Accès d'un utilisateur drh non autorisé pour une route (7 ms)
  ✓ Accès d'un utilisateur simple non autorisé pour une route (5 ms)
  ✓ Accès d'un utilisateur non autorisé sans token (29 ms)

Test Suites: 1 passed, 1 total
Tests:       13 passed, 13 total
Snapshots:  0 total
Time:        2.646 s, estimated 4 s
Ran all test suites.

```

CHAPITRE 9 : BUDGET RÉEL

Lors de l'initialisation du projet Bastion, le prestataire a pris en compte plusieurs critères pour estimer le budget global :

- **Taux Journalier Moyen (TJM)** : Fixé à 270 €/jour pour chaque développeur junior full-stack.
- **Coût de l'infrastructure** : Matériels informatiques et hardware.
- **Temps de production** : Plus la durée de développement est longue, plus le coût final est élevé.
- **Hébergement** : Les frais liés à la mise en ligne et au maintien de l'application.

Répartition du Projet en Phases

- **Phase de Préparation** (4 mois) : Planification, conception et configuration.
- **Phase de Développement** (6 mois) : Travail de développement intensif.

Estimation du Nombre de Jours de Travail

En supposant que chaque développeur travaille à temps plein (22 jours ouvrés par mois) :

- **Phase de Développement** (6 mois) : $6 \text{ mois} \times 22 \text{ jours/mois} = 132 \text{ jours}$
- **Phase de Préparation** (4 mois) : $4 \text{ mois} \times 22 \text{ jours/mois} = 88 \text{ jours}$

Répartition du Temps de Travail

- **Application Web et Backend** : 50 % du temps total, soit 66 jours.
- **API Sécurisée** : 30 % du temps, soit 40 jours.
- **Application Mobile** : 30 % du temps, soit 40 jours.

Calcul des Coûts (TJM à 270 €/jour)

- **Développeur Web/Backend** (66 jours) : $66 \text{ jours} \times 270 \text{ €/jour} = 17\,820 \text{ euros}$
- **Développeur API Sécurisée** (40 jours) : $40 \text{ jours} \times 270 \text{ €/jour} = 10\,800 \text{ euros}$
- **Développeur Mobile** (40 jours) : $40 \text{ jours} \times 270 \text{ €/jour} = 10\,800 \text{ euros}$

Ajout des Coûts Matériels et Hébergement

- **Coût du hardware** : 150 €
- **Coût par porte ajoutée** : 150 € × nombre de portes
- **Coûts d'hébergement** (par mois) :
 - Hébergement API : 25 €/mois
 - Hébergement Web : 25 €/mois
 - Hébergement Mobile : 25 €/mois
 - Hébergement de la base de données : 15,5 €/mois
 - **Total d'hébergement** : 90,5 €/mois
 - **Hébergement total pour 10 mois** : 90,5 €/mois × 10 mois = **905 euros**

Coût Total du Projet

Le coût total du projet Bastion est donc composé de :

- **Développeur Web/Backend** : 17 820 euros
- **Développeur API Sécurisée** : 10 800 euros
- **Développeur Mobile** : 10 800 euros
- **Coût du hardware** : 150 euros
- **Coût des portes ajoutées** : 150 € × nombre de portes
- **Coût d'hébergement** : 905 euros

Coût du Support

- **Coût du support** (prestation unique) : **270 euros**

Coût Total avec Support

- **Total avec support** : 17 820 € + 10 800 € + 10 800 € + 150 € + (150 € × nombre de portes) + 905 € + 270 € = **39 745 euros + coût des portes**

Conclusion

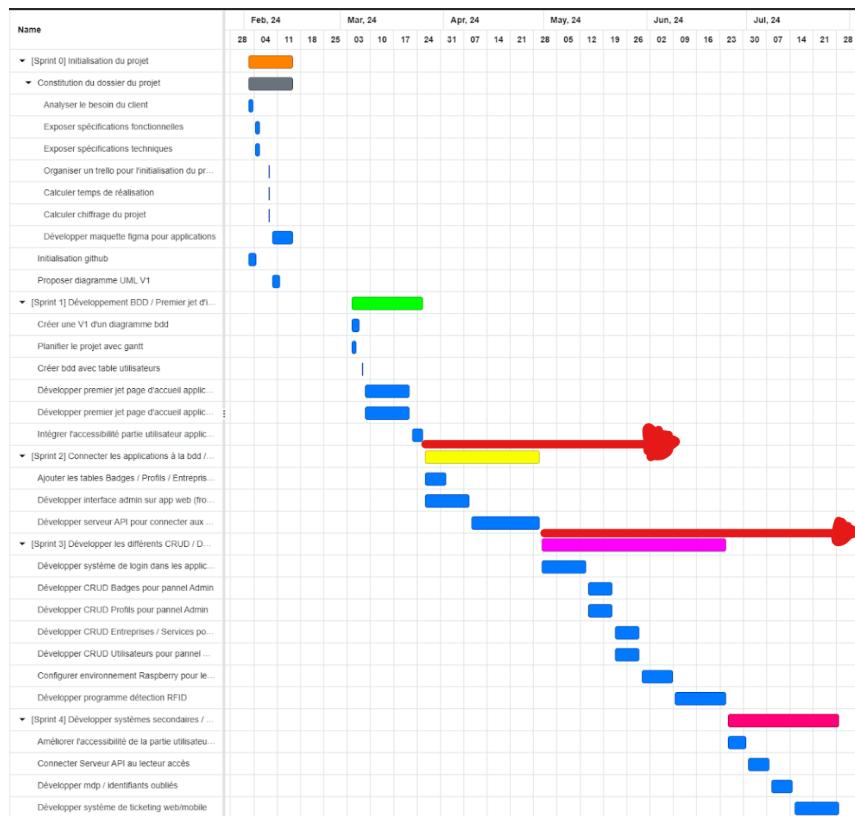
Le coût total du projet Bastion, en incluant les développeurs, le matériel, les portes ajoutées, l'hébergement, et le support, est de : **39 745 euros (+ coût des portes)**.

CHAPITRE 10 : CONCLUSION

A) Respect du planning de développement

En se basant sur le trello et le diagramme gantt établi, le prestataire a suivi la méthode agile pour développer le projet Bastion (*cf Annexes : Gant Notion et Trello*).

Le projet était initialement prévu en 4 sprints, 1 sprint a été ajouté en cours de développement dû à l'ajout de demande et détails de la part du client (Frank Quirin).



Le développement a pris du retard sur les sprints 2 et 3 notamment sur les créations des CRUD de toutes les tables de l'application Bastion (comme le montre les flèches rouges).

Ce qui en conclusion de projet a accusé un retard de 1 mois par rapport à la fin de sprint 4 initiale prévue.

B) Conclusion

Le client avait exprimé un besoin précis : disposer d'une solution logicielle capable de gérer les accès à son infrastructure tout en surveillant le temps de travail des employés, dans le but d'optimiser la gestion des ressources humaines. Cette demande visait non seulement à assurer un contrôle efficace des entrées et sorties, mais aussi à faciliter le calcul du temps de présence des collaborateurs (journalier, hebdomadaire, mensuel) et à identifier les personnes présentes ou tentant d'accéder à l'entreprise en dehors des heures autorisées.

Pour répondre à ces exigences, nous avons développé **Bastion**, une solution mobile et web complète. En intégrant des technologies modernes comme **React**, **Node.js** et une **API RESTful**, Bastion permet de gérer les accès et le temps de travail de manière fluide et sécurisée. L'application s'appuie sur un système de badgeage via **puces RFID** ou **application mobile** et prend en charge l'intégration de dispositifs matériels comme des cartes **Raspberry Pi** ou **Arduino** pour gérer les points d'accès physiques.

Bastion se distingue par son approche sécurisée, en respectant les bonnes pratiques de **sécurité des données** et en étant conçu avec des principes d'**écoconception**. Le développement du projet a été mené par une équipe de trois développeurs Full-Stack, répartis sur différents aspects techniques : **Romain GILOT** sur le développement front-end, **Loan KEOVILAY** sur le back-end, les bases de données et le hardware, et **Noé ZIADI**, en tant que chef de projet, sur le développement front-end et API.

En résumé, **Bastion** constitue une réponse sur mesure aux besoins du client, apportant une solution technologique robuste et évolutive pour la gestion des accès et du temps de travail dans les entreprises, tout en facilitant les processus RH grâce à une intégration simple et une gestion centralisée.

C) Remerciements

Le prestataire exprime sa gratitude envers toutes les personnes qui ont participé de près et de loin à la réalisation de ce projet. Le prestataire et l'équipe de développement remercient

Cédric BRASSEUR, Théo GAMORY, Franck QUIRIN, Stéphanie BONNE et Alexandre LEROUX pour leurs conseils et leurs expertises qui ont permis de mener à bien ce projet.

ANNEXES

Page d'accueil de l'application web

BASTION

john doe ▾

Accueil Absence Planning Assistance

Bonjour john,
Bienvenue sur votre espace de gestion de temps de travail.

Temps de travail

- 0h00 Journalier
- 7h00 Hebdomadaire
- 15h02 Mensuel

Mes absences

Sousmis	Raison	Status	Période
Aucune absence trouvée pour le statut sélectionné			

Mes permissions

Page : Gestion des utilisateurs de l'application web :

localhost:3000/admin

BASTION

john doe ▾

Accueil Absence Planning Assistance

PANEL GESTION

- Dashboard
- Entreprises
- Utilisateurs
- Permissions & Portes
- Badges
- Qrcodes
- Assistances
- Absences
- Logs

Gestion des utilisateurs

EMPLOYÉS	RÔLE	EMAIL	TÉLÉPHONE	ENTREPRISE	ACTIONS
solo solo	Assistant-RH	solo@bastion.fr	+33 0123456789	Non spécifié	Modifier Supprimer
fabrice tesson	DRH	tesson@bastion.fr	+33 0329171957	Example Company	Modifier Supprimer
john doe	Administrateur	johndoe@bastion.fr	+33 0123456789	Example Company	Modifier Supprimer

Page de référencement SEO :



Organisation github :

The screenshot shows the GitHub profile page for the 'Bastion' repository. It includes a profile picture, the repository name 'Bastion', a description 'Projet Access Bastion From Metz Numeric School made by students', and a 'Unfollow' button. The page displays several repositories: 'api' (Private, updated 1 hour ago), 'web' (Private, updated 3 hours ago), 'access-reader' (Private, updated yesterday), 'mobile' (Private, updated last week), and 'mobileArchived' (Private archive, updated on May 30). On the right side, there are sections for 'View as: Public', 'People' (showing four user icons), and 'Top languages' (JavaScript and Python).

Lecteur de badge :

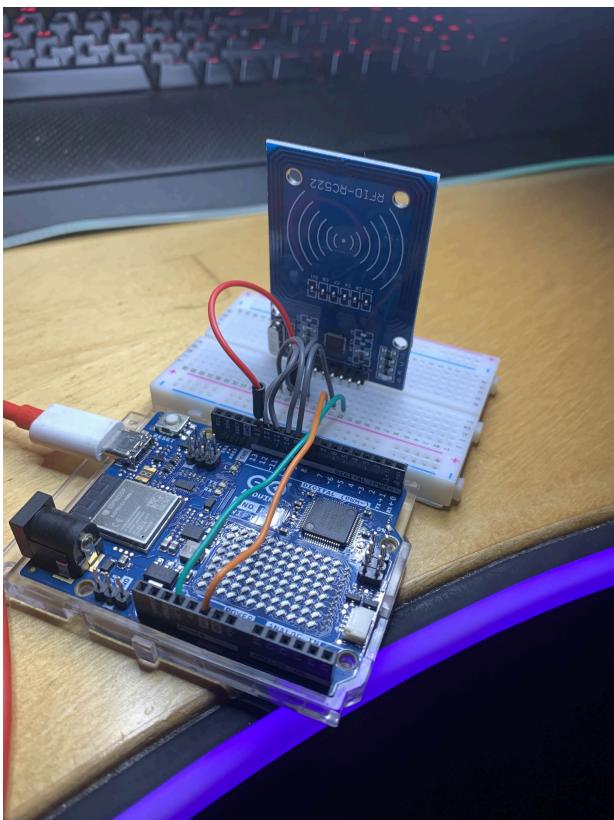
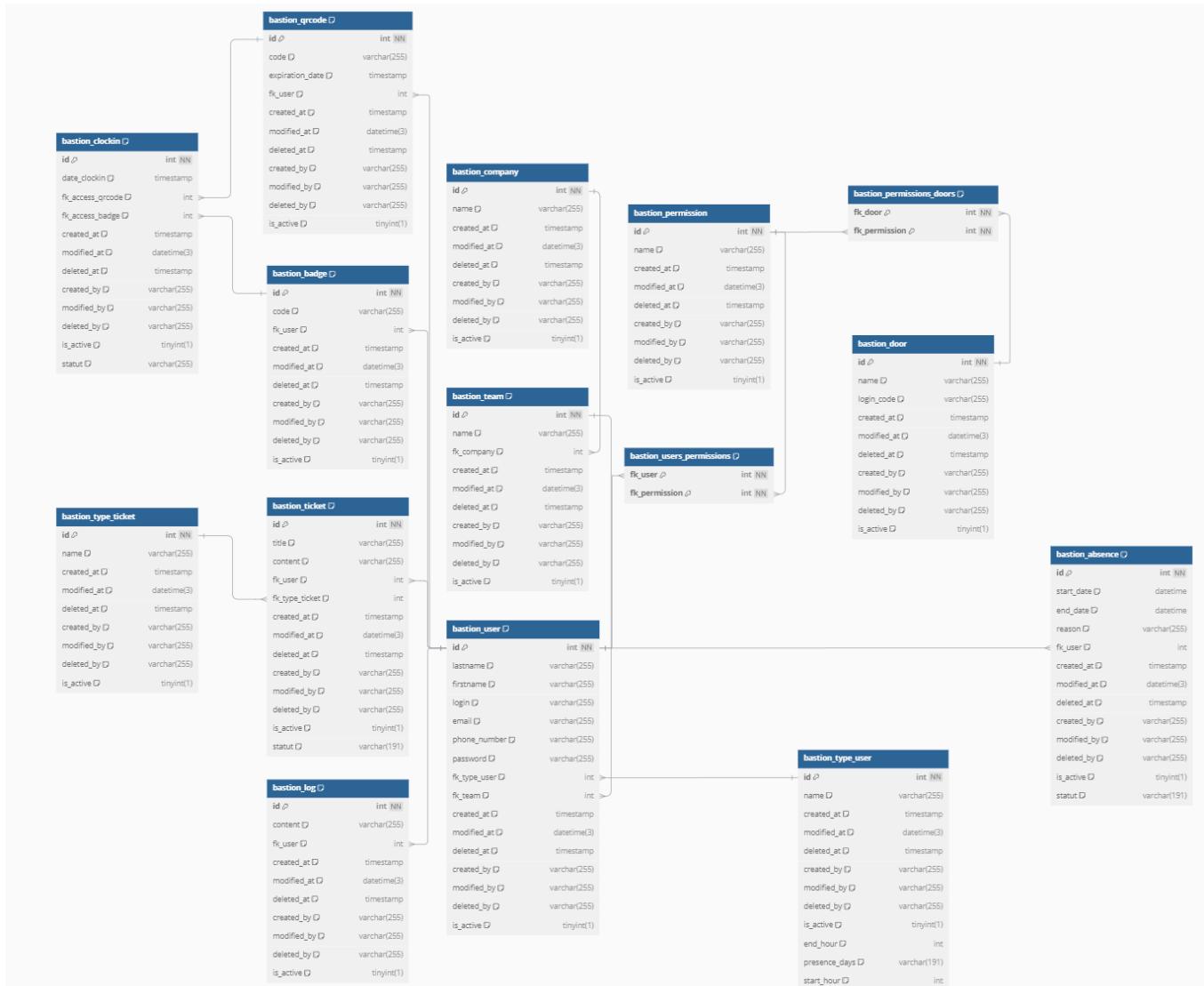


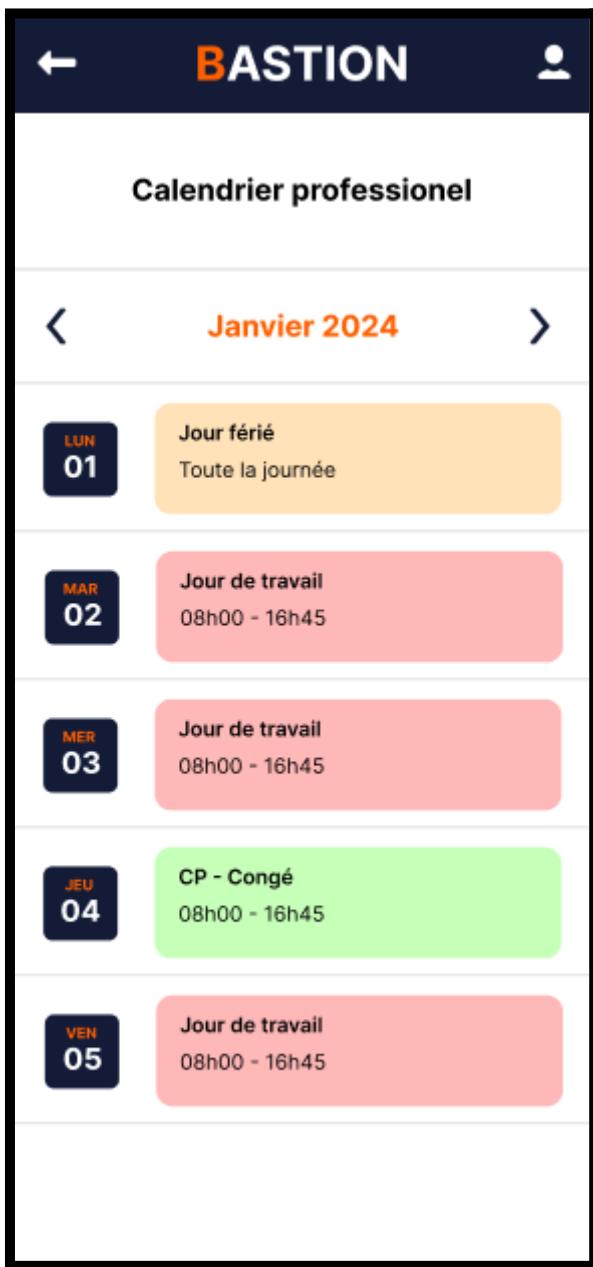
Diagramme de la base de données :



Page génération du QR code sur l'application mobile :



Page calendrier professionnel sur l'application mobile :



Exemple de génération du CSV :

ID	Prénom	Nom	Email	Téléphone	Login	Entreprise	Équipe	Présence (Mois)	Présence (Heure)	Présence (Jours)	Actif
10	john	DOE	johndoe@bastion	123456789	johndoe	Example Compa	Development	Te: 15h02	7h00	0h00	No
11	fabrice	TESSON	tesson@bastion	3291719572	tesson	Example Compa	Development	Te: 4h00	4h00	0h00	No
12	solo	SOLO	solo@bastion.fr	123456789	solo	Example Compa	Development	Te: 0h00	0h00	0h00	No
13	Noé	ZIADI	noeziadi@outlook	777921473	nziadi	Example Compa	Development	Te: 0h00	0h00	0h00	No
14	Loan	KEOVILAY	loan.keovilay@o	666554432	lkoovilay	Example Compa	Development	Te: 0h00	0h00	0h00	No

Gantt Notion et Trello :





Trello : <https://trello.com/b/02b2QNhP/bastion>