Cahier des Charges Technique :

DragN'Receive

CLEMENT Pierre, ADOUX Gabin, CELLOT Guéric, DUPORT Pauline de DragN'Receive

Le 04/03/2023

Pour l'ITII Picardie de Beauvais - Session FA25

1 - Analyse fonctionnelle

1.1 - Présentation générale

La société DragN'Receive souhaite créer un système de Click & Collect à l'échelle de toute l'agglomération de Beauvais pour le commencement, en ayant pour partenaires des marchés et commerces centrés principalement sur la vente de denrées alimentaires et de produits de grande consommation 100% français.

Un site web doit donc être réalisé, permettant la gestion des boutiques partenaires, permettant ainsi à ces dernières de proposer leurs produits, d' ajouter de nouveaux produits, gérant stocks et commandes clientes.

DragN'Receive envisage désormais la conception d'une application mobile permettant aux clients de réaliser des commandes dans les différentes boutiques affiliées situées dans l'agglomération de Compiègne. La méthode de distribution de cette application serait par l'affichage ou la distribution d'un QR code dans les boutiques partenaires permettant l'installation de l'application pour les clients.

Une API devra donc être réalisée et sera donc à faire évoluer entre les deux différentes applications.

1.2 - Analyse du besoin

Pour la réalisation de l'application mobile, il sera nécessaire d'utiliser un serveur, permettant l'hébergement du client mais aussi une éventuelle base de données. L'utilisation d'un langage de programmation compatible au développement mobile sera également nécessaire, il faudra également mettre en place une interface graphique, se basant sur la charte graphique définie.

Le but serait donc de réaliser une application mobile disponible sur Android et IOS, focalisée sur la vente de produits des boutiques partenaires pour les clients, qui géreront et suivront leurs commandes.

1.3 - Solutions envisagées

Les solutions envisagées permettant la réponse des divers besoins du projet sont :

- Utiliser le serveur dédié de l'entreprise ou bien utiliser un serveur OVH loué par cette dernière.
- L'utilisation d'un de ces langages de programmation : JavaScript, Java, Kotlin, C++ étant les langages les plus utilisés pour la réalisation d'applications mobiles.
- La réalisation d'une interface de programmation (API) communiquant avec (et entre) l'application mobile, la base de données. Les langages les plus répandus pour sa réalisation sont le Python, le JavaScript ainsi que le Java.
- L'utilisation d'un outil permettant la conversion / compatibilité du développement vers une solution mobile tels que : Ionic, NativeScript, Flutter.

1.4 - Solution retenue

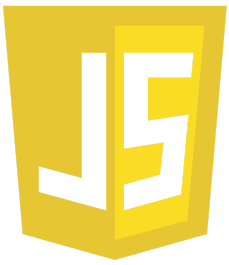
1.4.1 - Présentation des choix

La solution optimale permettant de répondre à chacunes des problématiques serait donc, selon notre expertise :

- D'utiliser le serveur dédié, permettant la mise en place de l'application mobile, de plus la base de données et l'API créé pour le site web y sont stockées et seront utilisées pour ce projet.
- Nous avons retenu comme langage de programmation le JavaScript avec l'utilisation du framework Vue, également utilisé dans le projet précédent.
- Pour ce qui est de l'API, nous allons utiliser l'API réalisée durant le dernier projet permettant de la faire évoluer durant le développement mobile, fait en JavaScript avec l'utilisation de l'environnement d'exécution NodeJS accompagné du framework ExpressJS.
- Pour ce qui est du framework mobile, nous avons sélectionné Ionic, permettant d'utiliser le framework Vue, simplifiant donc grandement la vitesse de développement du projet et permettant une ressemblance avec le site web étant tout deux réalisés en Vue.

1.4.2 - Benchmark

1.4.2.1 - Langage de programmation mobile



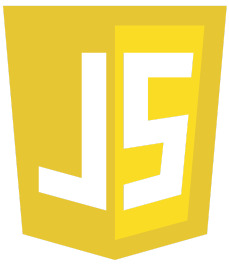

	JavaScript	Java	Kotlin	C++
Utilisation	Généraliste	Généraliste	Généraliste	Client Lourd, Mobile
Typage	Fort	Fort	Fort	Fort
Avis	Langage très simple de compréhension, orienté objet, en plein essor ces dernières années dans de multiples domaines par ses performances de dynamisation et de polyvalence.	Langage de programmation orienté objet, demande un code rigoureux et long, compilation rapide, permet de créer divers types de clients. 100% compatible Android.	Langage de programmation orienté objet, compilé pour JVM et JavaScript. Concurrent de Java sur le marché du développement mobile Android.	Langage de programmation orienté objet compilé, langage très pointilleux permettant l'optimisation des performances et la précision pour des applications variées.
Date de lancement	1996	1995	2011	1985
Outils Mobile	Ionic, NativeScript, React Native	Java Natif	Kotlin Multiplatform Mobile	C++ Natif

1.4.2.2 - Framework mobile



	Ionic	NativeScript	Flutter
UI	HTML / CSS / Composants	XML / CSS / Composants	Dart / Widgets
Apprentissage	Apprentissage facile et rapide, documentation complète, l'utilisation d'un framework web permet de réaliser rapidement une application mobile. Très intéressant pour sa rapidité si le balisage HTML est maîtrisé. Très similaire au développement web.	Le plus performant des 3, cependant c'est également celui qui nécessite le plus d'apprentissage. Offrant beaucoup de libertés, cela à un coût. Il fonctionne également avec du HTML et donc similaire au développement web. La documentation est très riche.	Apprentissage rapide et simple par l'utilisation des widgets à l'intérieur des scripts, il ne requiert pas la connaissance de l'HTML. La documentation ainsi que la chaîne YouTube officiel permettent d'avoir une compréhension accrue.
Écosystème	Gros écosystème, regroupant ceux de Vue Angular et React.	Communauté impliquée, moins importante mais encore en plein essor.	Croissance importante, moins importante que Ionic mais reste conséquente.
Langage	JavaScript, TypeScript	JavaScript, TypeScript	Dart
Compatibilités	Android et IOS	Android et IOS	Android et IOS
Performances	La moins performante, mais reste puissante.	Bonne performance, au coude à coude avec Flutter.	Bonne performance, au coude à coude avec NativeScript.
Taille	Léger	Plus lourd que Ionic et Flutter	Moins léger que Ionic
Date de lancement	2013	2015	2017
Utilisé par	Electronic Arts, Airbus Helicopters	Raiffeisen Bank, Sennheiser, PUMA	BMW, Alibaba, eBay, Toyota, Google Pay

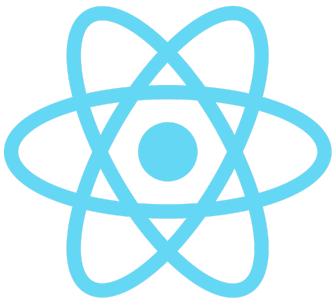
1.4.2.3 - Langage de programmation web




	JavaScript	Ruby	Php	Python
Utilisation	Généraliste	Généraliste	Web (Omniprésent)	Généraliste
Typage	Fort	Fort	Fort (depuis la 7.1)	Fort
Avis	Langage très simple de compréhension, en plein essor ces dernières années dans de multiples domaines par ses performances de dynamisation et de polyvalence.	Lisible, léger, rapide et simple à écrire. Permet de gagner du temps dans la programmation par ses performances. Langage sous-estimé en pleine résurgence.	Simple à comprendre, grosse marge d'erreur dans le code et nécessite un apprentissage supplémentaire pour empêcher les failles de sécurité.	Langage possédant des code plus verbeux (plus explicite et plus long), la compréhension est très rapide. Sa polyvalence permet de réaliser de petits comme de gros projets.
Date de lancement	1996	1995	1994	1991
Frameworks	Vue, Angular, React, ExpressJS	Ruby on rails	Symfony, Laravel	Django

1.4.2.4 - Framework web





	Vue	Angular	React
Réactivité	Très réactif, une seule page permet le changement d'affichage sans rechargement	Bien réactif, permet de changer le contenu sans rafraîchissement, possibilité de faire des formulaires réactifs	Si un composant modifié est rendu, tous ses sous-composants également même ceux non modifiés le seront (à configurer manuellement)
UI	HTML / CSS / Composants	HTML / CSS	Gestion basé en JSX (même le CSS) nécessite l'apprentissage
Routage	Router Vue officiel	Router Angular officiel	Pas de router officiel
Ecosystème	Important mais moin que React	Structure très encadré	Énorme, gros avantage
Apprentissage	Documentation irréprochable, apprentissage facile et rapide.	Conséquent pour réaliser des applications complexe	Conséquent pour réaliser des applications complexe
Mobile	Vue Native / Ionic	Angular 2 et NativeScript	React Native, testé et stable
Performances	Pas de différences significative	Pas de différences significative	Pas de différences significative
Taille	Très léger	2x plus lourd que Vue	Un petit peu plus lourd que Vue
Date de lancement	2014	2010	2013
Utilisé par	Alibaba, GitLab	Google, Wix	Facebook, Uber

1.4.3 - Explication des choix

Notre équipe à donc choisi d'utiliser le langage JavaScript qui, pour la réalisation du projet, est le plus approprié, utilisable avec de multiples frameworks, notre choix se justifiant par ses performances mais également car il va nous permettre de réaliser aussi bien l'application que le côté permettant les interactions avec la base de données. De plus, il a déjà fait ses preuves pour la réalisation du site web de DragN'Receive.

Nous avons donc choisi trois frameworks, l'un pour la réalisation du **Back-End***, l'autre pour le **Front-End**** et le dernier pour rendre compatible le logiciel sous forme d'une application mobile**.**

Nous avons choisi en Back-End le framework ExpressJS basé sur du NodeJS.

Pour ce qui est du Front-End nous avons décidé d'utiliser Vue. Le choix de ce dernier, n'aura aucun impact sur ses performances mais aura un impact positif sur la rapidité, qualité et compréhension du code de notre équipe.

Pour l'outil mobile, nous avons choisi le framework Ionic car ce dernier est compatible avec Vue et plus globalement le JavaScript HTML et CSS, possédant des environnements de configuration qui permettent donc des libertés visuelles. Ionic permet également d'utiliser divers composants permettant un développement rapide et une interface soignée et homogène. Il permet aussi d'utiliser l'application sur Android ainsi que IOS permettant de cibler la majorité des utilisateurs mobile.

***Back-End :** Partie caché de l'utilisateur, permettant la gestion de la base de données et les interactions de cette dernière permettant la communication avec le Front-End

****Front-End :** Partie d'interfaces graphiques visible par l'utilisateur qui vient interagir avec le Back-End pour retourner/envoyer des éléments à l'utilisateur/la base de données.

2 - Conception fonctionnelle

2.1 - Dictionnaire de données

2.1.1 - Utilisateur

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
lastname	varchar	50	Obligatoire	nom de famille
firstname	varchar	50	Obligatoire	prénom
email	varchar	100	Obligatoire, format d'email	adresse mail
password	varchar	60	Obligatoire, comporte une majuscule, une minuscule, un nombre et un caractère spécial au minimum	mot de passe
path	varchar	255	Non-Obligatoire	chemin vers la photo
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification
roleId	varchar	1	FK: Rôle, Obligatoire	permet de séparer, admins, clients et partenaires
shopId	varchar	36	FK: Boutique, Non-Obligatoire	relie la boutique du partenaire

2.1.2 - Rôle

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	1	PK, Obligatoire	Chaîne de caractère entre 1 et 4
label	varchar	20	Obligatoire	nom du rôle
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification

2.1.3 - Boutique

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
name	varchar	50	Obligatoire	nom de la boutique
email	varchar	100	Obligatoire	adresse mail
phone	varchar	10	Non-Obligatoire	numéro de téléphone
city	varchar	60	Non-Obligatoire	ville
street	varchar	100	Non-Obligatoire	rue
postal	int	5	Non-Obligatoire	code postal
path	varchar	255	Non-Obligatoire	chemin vers le logo
deleted	tinyint (bool)	1	Obligatoire	Booléen pour savoir si la boutique est supprimée
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification

2.1.4 - Produit

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
name	varchar	50	Obligatoire	nom du rôle
description	varchar	255	Obligatoire	description du produit
price	float	9,2	Obligatoire	prix
stock	int	11	Obligatoire	stock restant
path	varchar	255	Obligatoire	chemin vers l'image du produit
deleted	tinyint (bool)	1	Obligatoire	entier 1 ou 0 pour savoir si le produit est supprimé
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification
shopId	varchar	36	FK: Boutique, Obligatoire	relie le produit à sa boutique

2.1.5 - Commande

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
quantities	int	11	Obligatoire	quantité de produit
price	float	9,2	Obligatoire	prix total de la commande
number	varchar	100	Obligatoire	numéro de commande
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification
userId	varchar	36	Obligatoire	relie l'utilisateur à la commande
productId	varchar	36	Obligatoire	relie le produit à la commande
shopId	varchar	36	Obligatoire	relie la boutique à la commande
orderStatusId	varchar	1	FK: Statut Commande, Obligatoire	relie un statut à la commande

2.1.6 - Statut Commande

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	1	PK, Obligatoire	Chaîne de caractère entre 1 et 5
label	varchar	20	Obligatoire	nom du statut
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification

2.1.7 - Ticket

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
title	varchar	60	Obligatoire	titre du ticket
content	varchar	255	Obligatoire	contenu
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification
userId	varchar	36	Obligatoire	relie l'utilisateur au ticket
ticketStatusId	varchar	1	FK: Statut Ticket, Obligatoire	relie un statut au ticket

2.1.8 - Statut Ticket

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	1	PK, Obligatoire	Chaîne de caractère entre 1 et 2
label	varchar	20	Obligatoire	nom du statut
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification

2.1.9 - Message

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
content	varchar	255	Obligatoire	contenu
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification
userId	varchar	36	Obligatoire	relie l'utilisateur au message
ticketId	varchar	36	FK: Statut Ticket, Obligatoire	relie un message au ticket

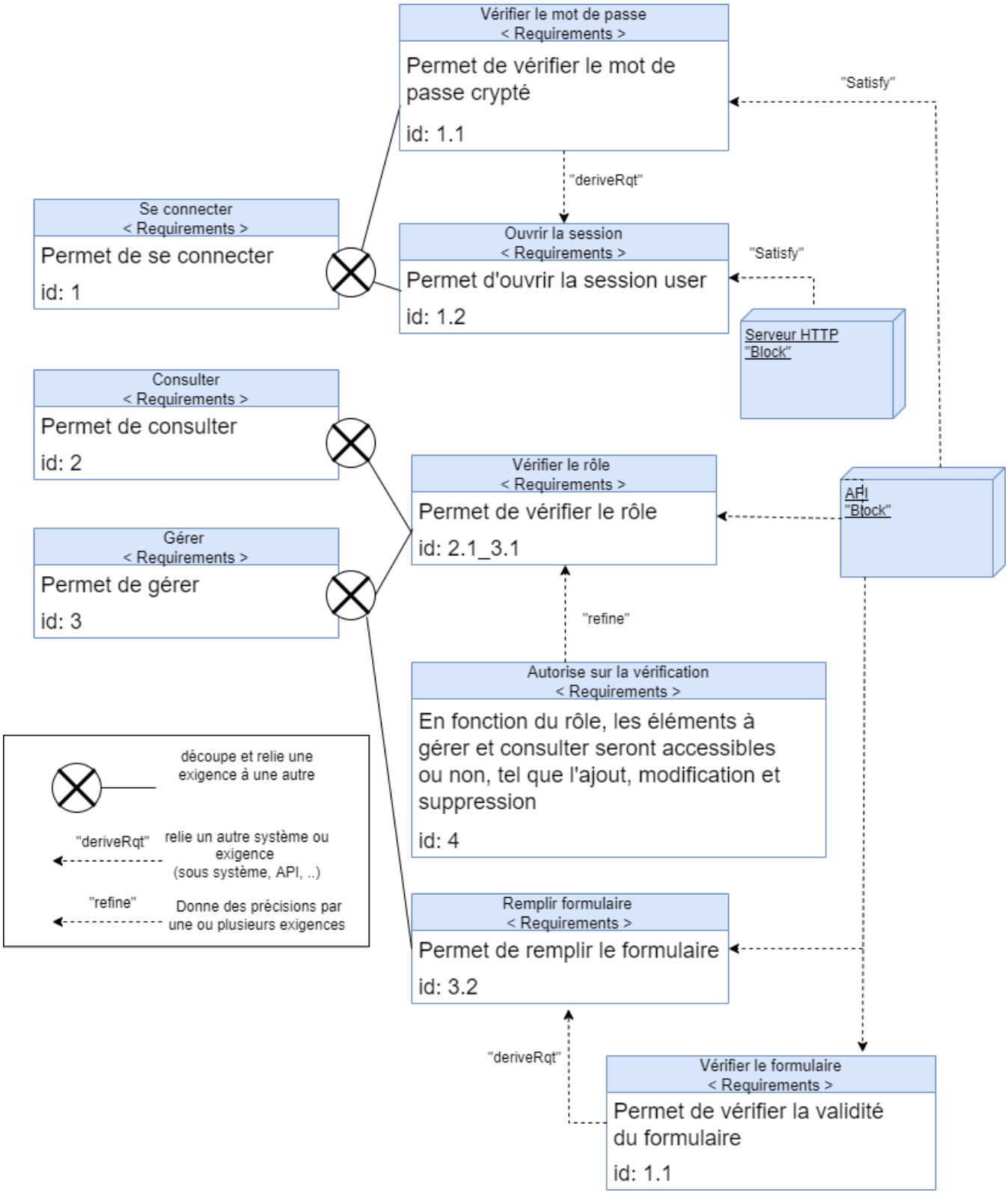
2.1.10 - Web Token

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
token	varchar	255	Obligatoire	token généré aléatoirement
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de la dernière modification
userId	varchar	36	Obligatoire	relie l'utilisateur au ticket

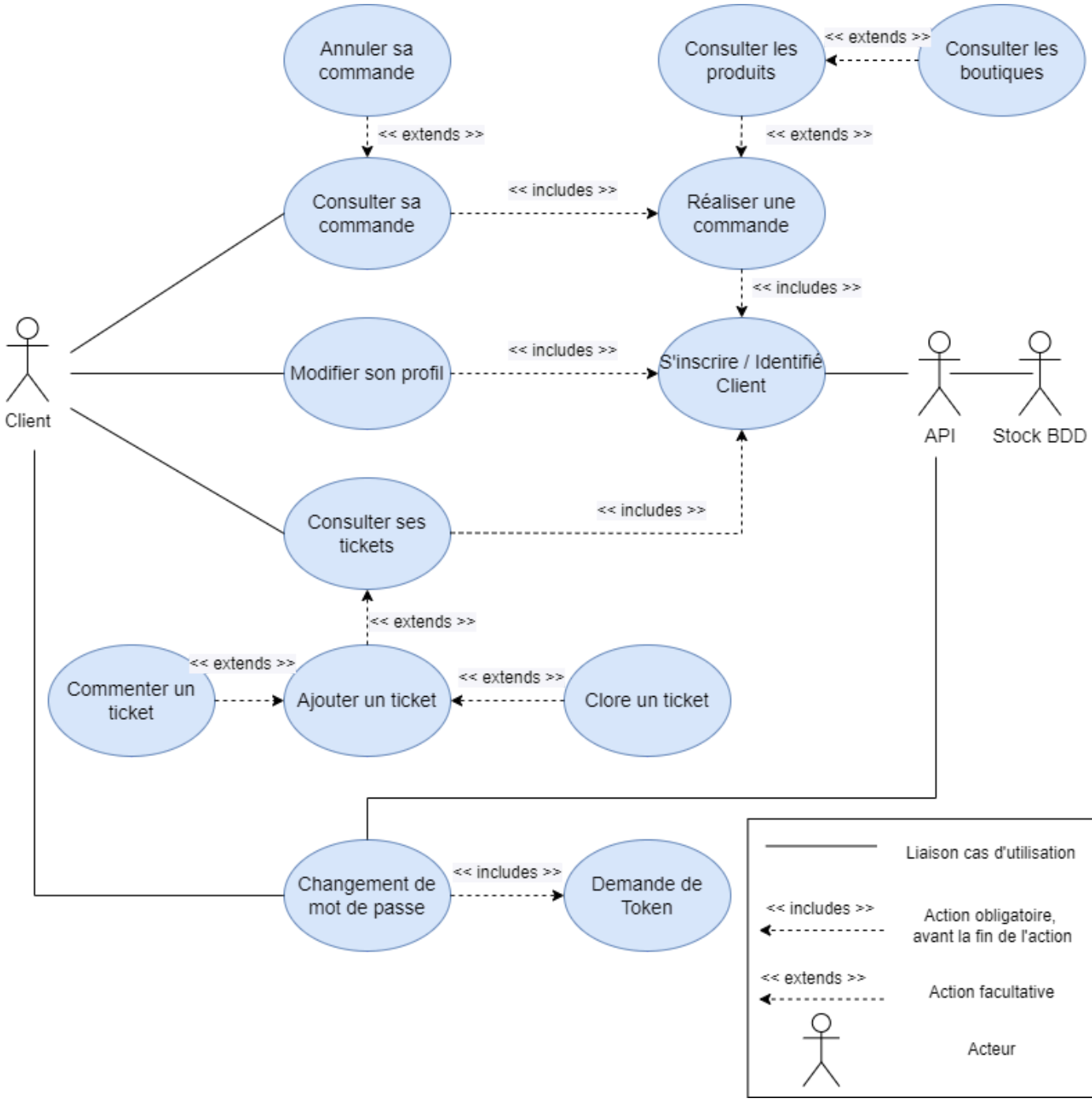
2.1.11 - Mobile Code

Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
code	varchar	4	Obligatoire	code généré aléatoirement (nombre)
createdAt	datetime	yyyy-mm-ddThh:mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh:mm:ss.000Z	Obligatoire	date de la dernière modification
userId	varchar	36	Obligatoire	relie l'utilisateur au ticket

2.2 - Exigences



2.3 - Cas d'utilisations

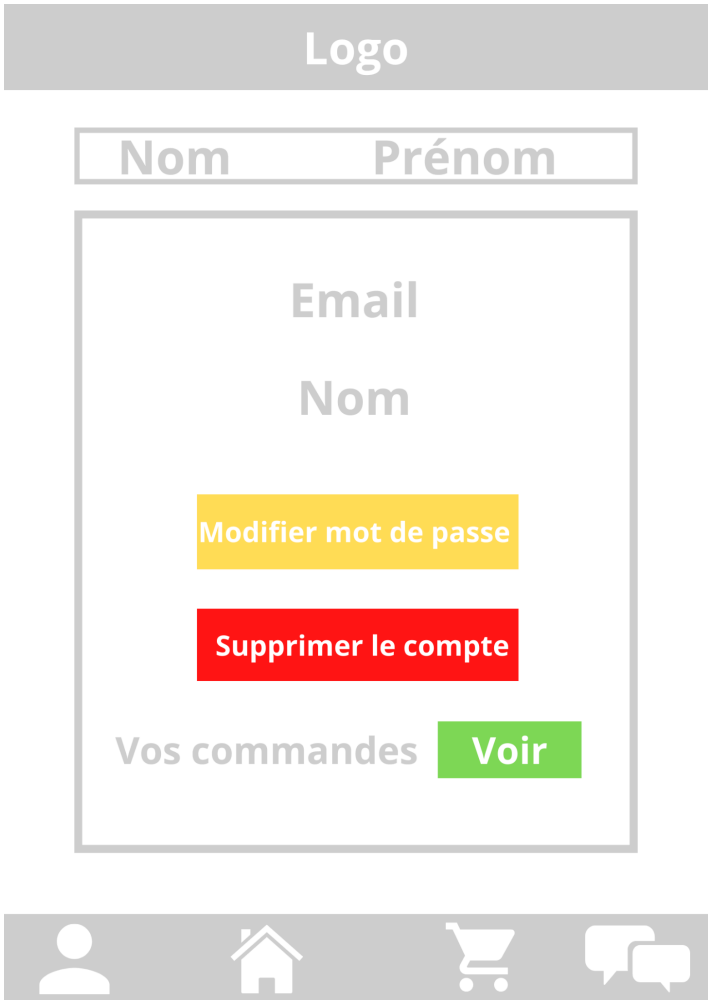


2.4 - Maquettes Mobile

⚠ Les maquettes suivantes sont des aperçus de la basse fidélité de l'application mobile. Elles sont donc volontairement dénuées de toutes formes de graphisme. Ces aperçus servent donc à lister et placer les éléments fonctionnels sur les différentes pages du site plus qu'à donner une idée de son esthétique, qui seront voués à changer en fonction des besoins exprimés.

2.4.1 - Connexion

2.4.2 - Profil

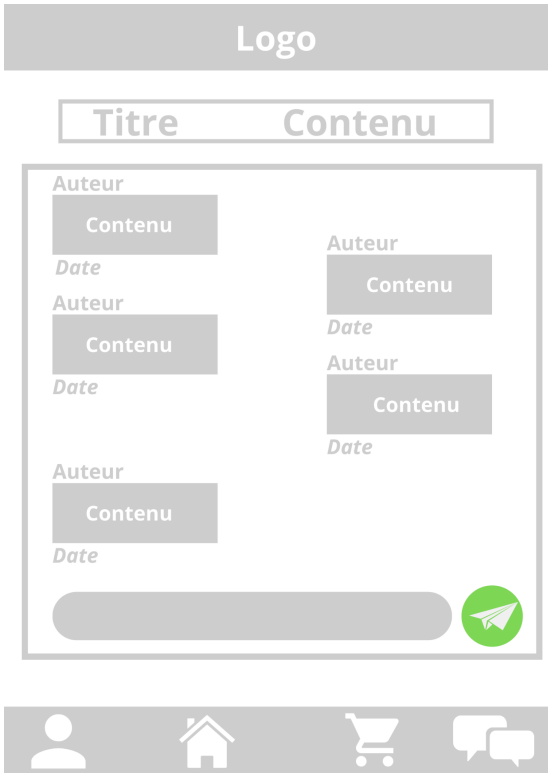

Logo
Nom
Prénom
Email
Nom
Modifier mot de passe
Supprimer le compte
Vos commandes
Voir

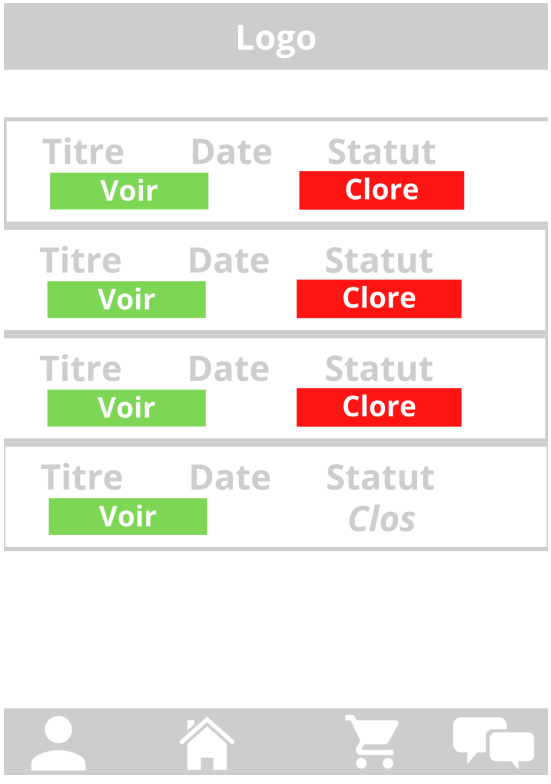
2.4.2 - Boutique

2.4.3 - Commande

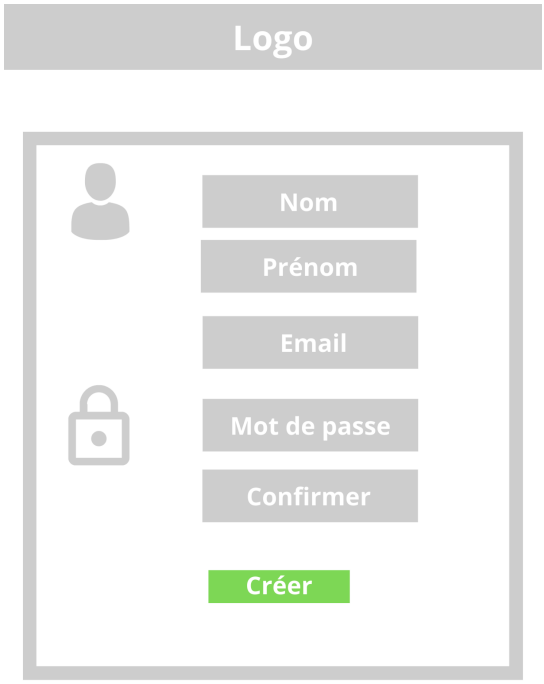
2.4.4 - Produit

2.4.5 - Ticket

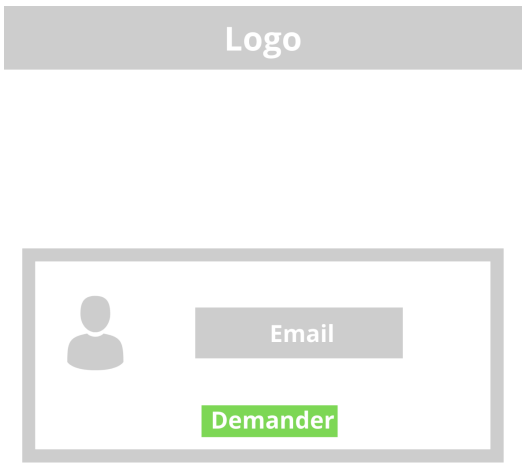

Logo
Titre
Contenu
Auteur
Contenu
Date
Auteur
Contenu
Date
Auteur
Contenu
Date
Auteur
Contenu
Date
Auteur
Contenu
Date


Logo
Titre
Date
Statut
Voir
Clore
Titre
Date
Statut
Voir
Clore
Titre
Date
Statut
Voir
Clore
Titre
Date
Statut
Voir
Clos

2.4.6 - Inscription



2.4.7 - Récupération de mot de passe

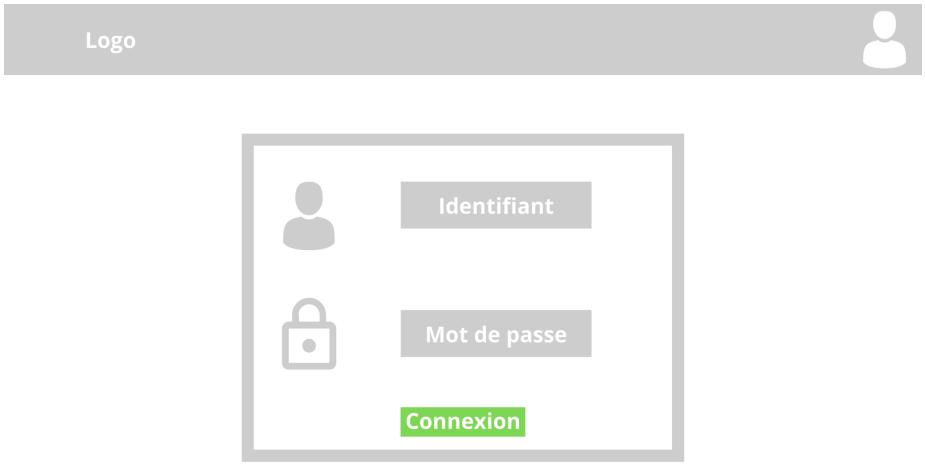


2.4.8 - Changement de mot de passe

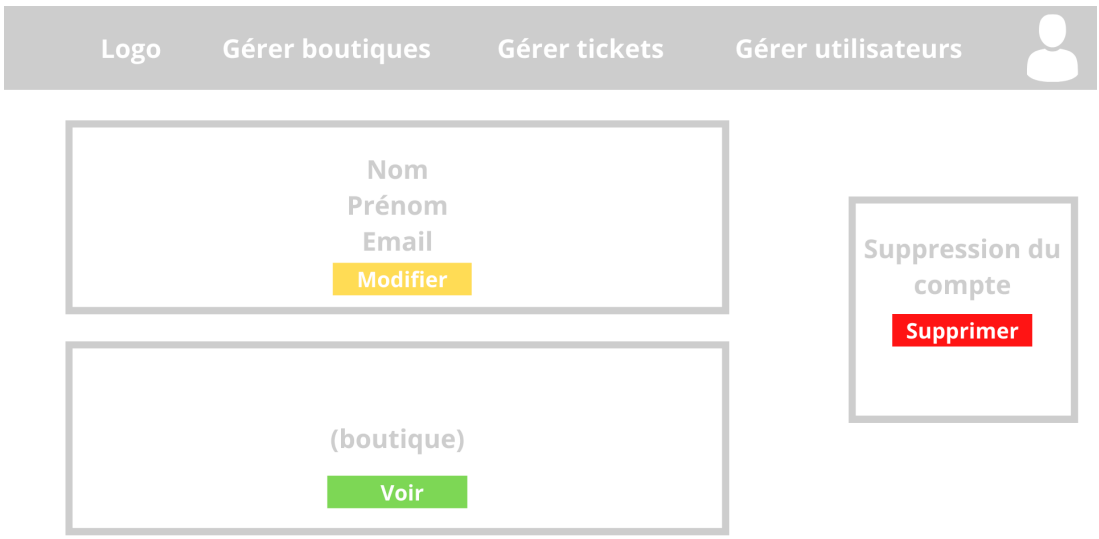
2.5 - Maquettes Web

⚠ Les maquettes suivantes sont des aperçus de la basse fidélité du site web. Elles sont donc volontairement dénuées de toute forme de graphisme.
Ces aperçus servent donc à lister et placer les éléments fonctionnels sur les différentes pages du site plus qu'à donner une idée de son esthétique, qui seront voués à changer en fonction des besoins exprimés.

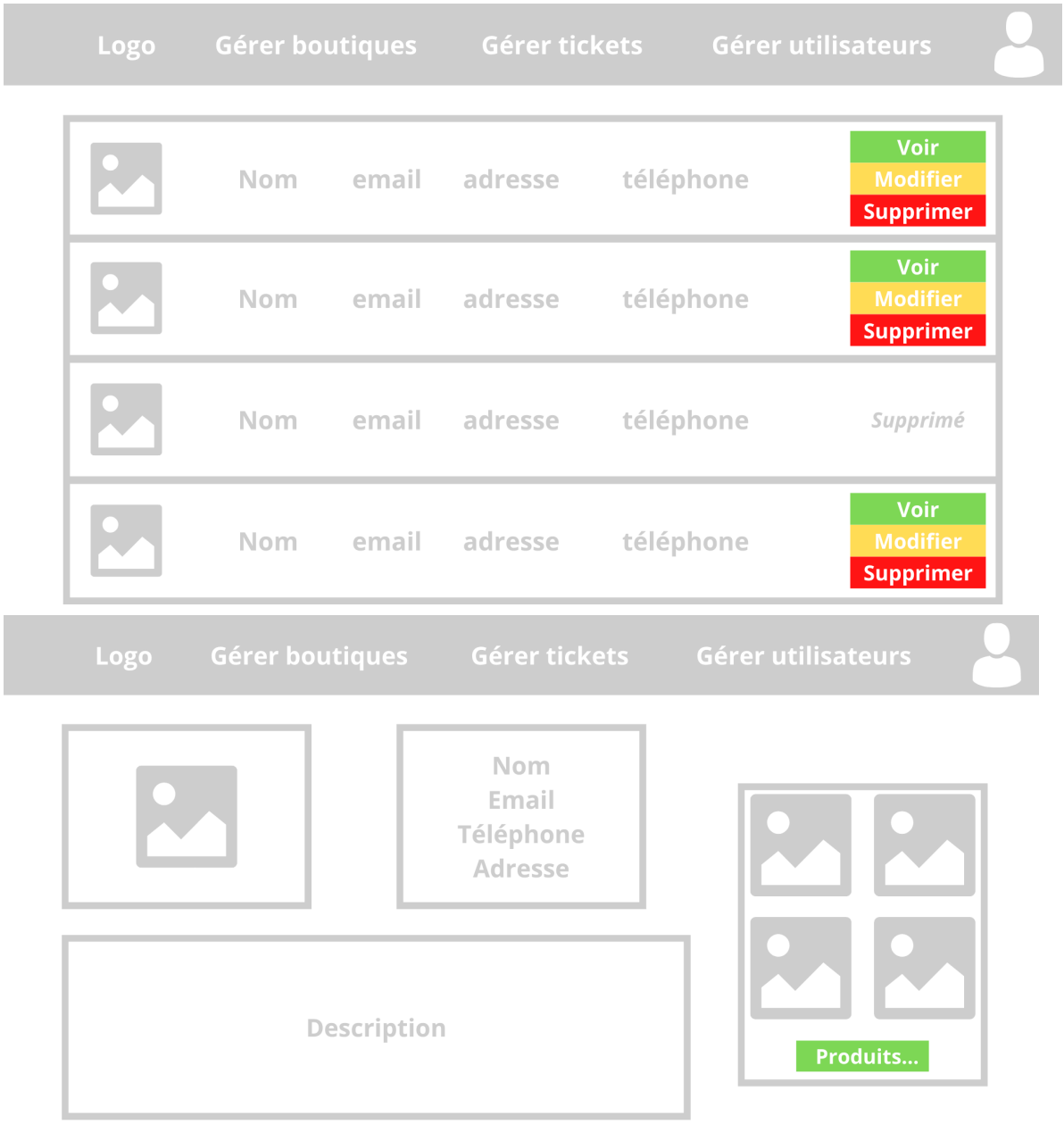
2.5.1 - Page de connexion



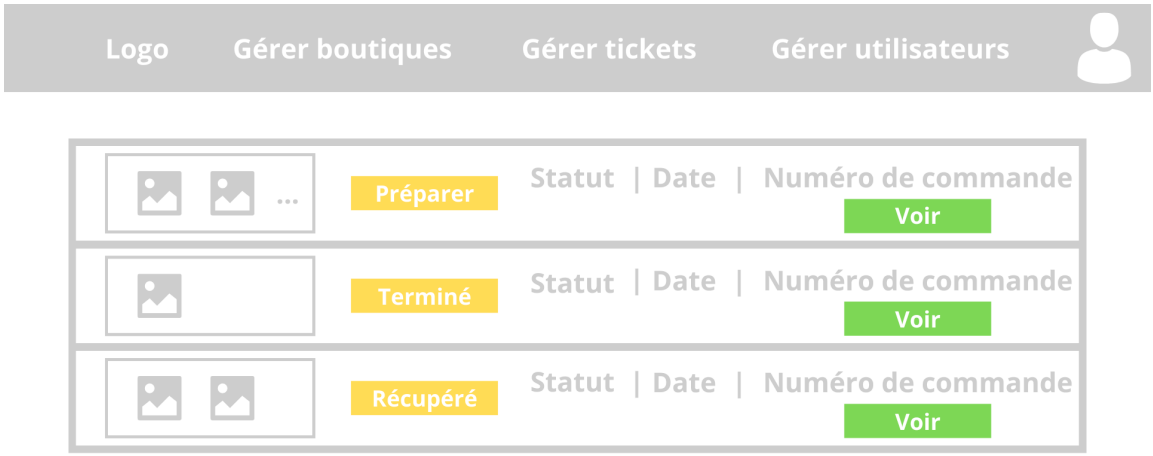
2.5.2 - Profil



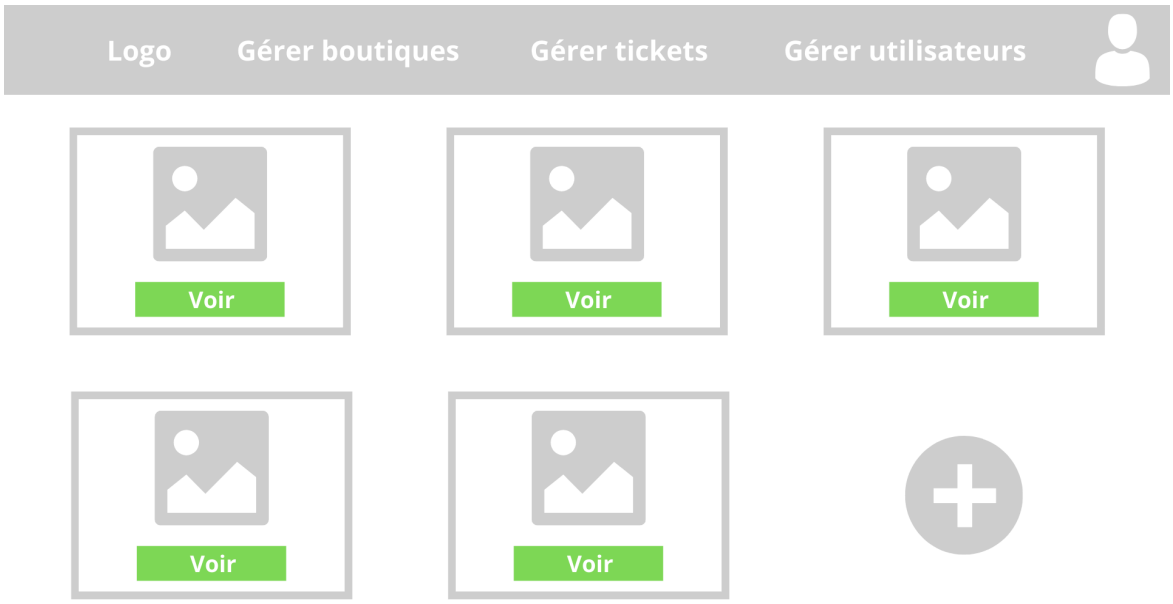
2.5.2 - Boutique



2.5.3 - Commande

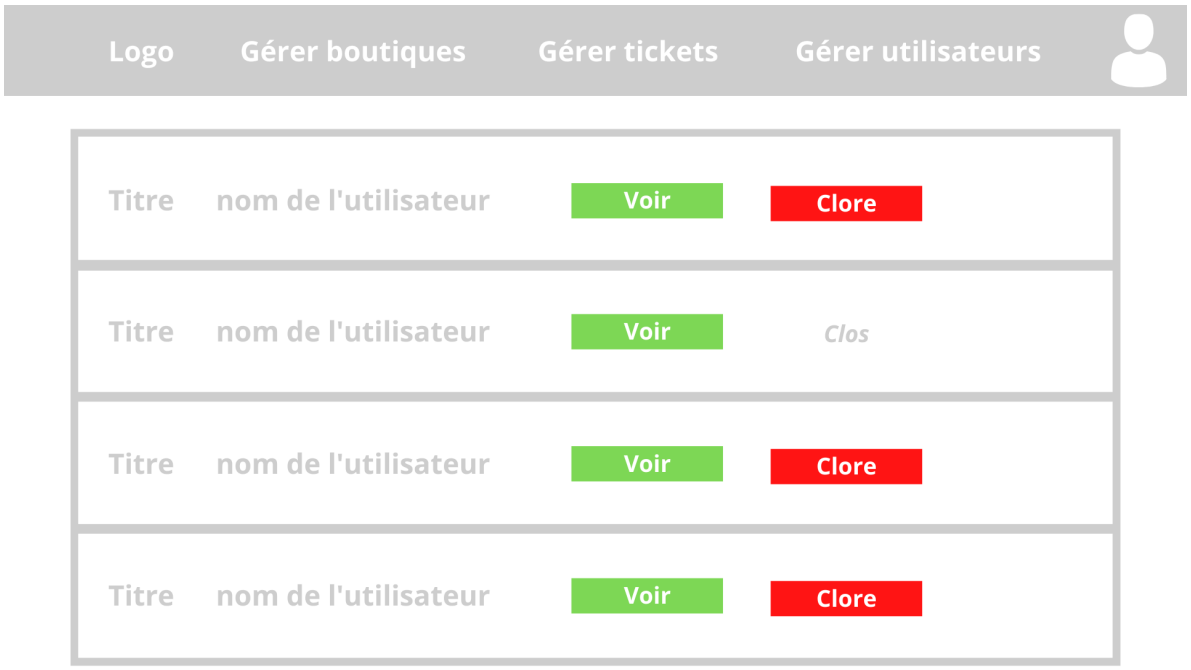


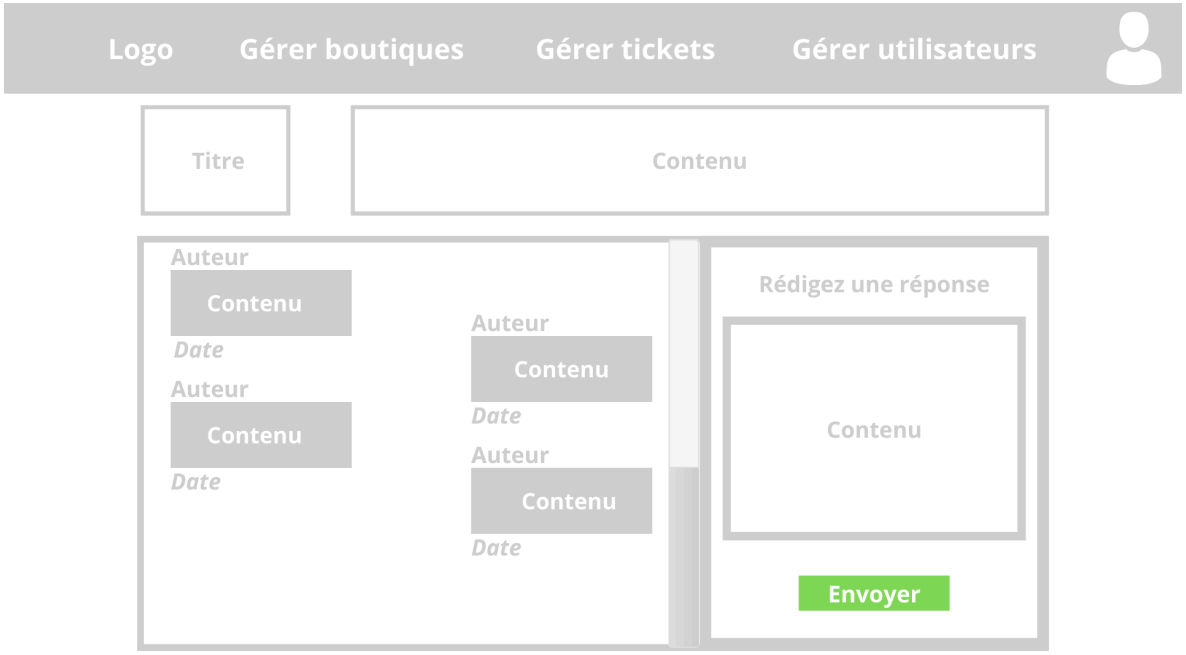
2.5.4 - Produit


Logo
Gérer boutiques
Gérer tickets
Gérer utilisateurs
Voir
Voir
Voir
Voir
Voir

Logo
Gérer boutiques
Gérer tickets
Gérer utilisateurs
Prix
Stock
Description

2.5.5 - Ticket

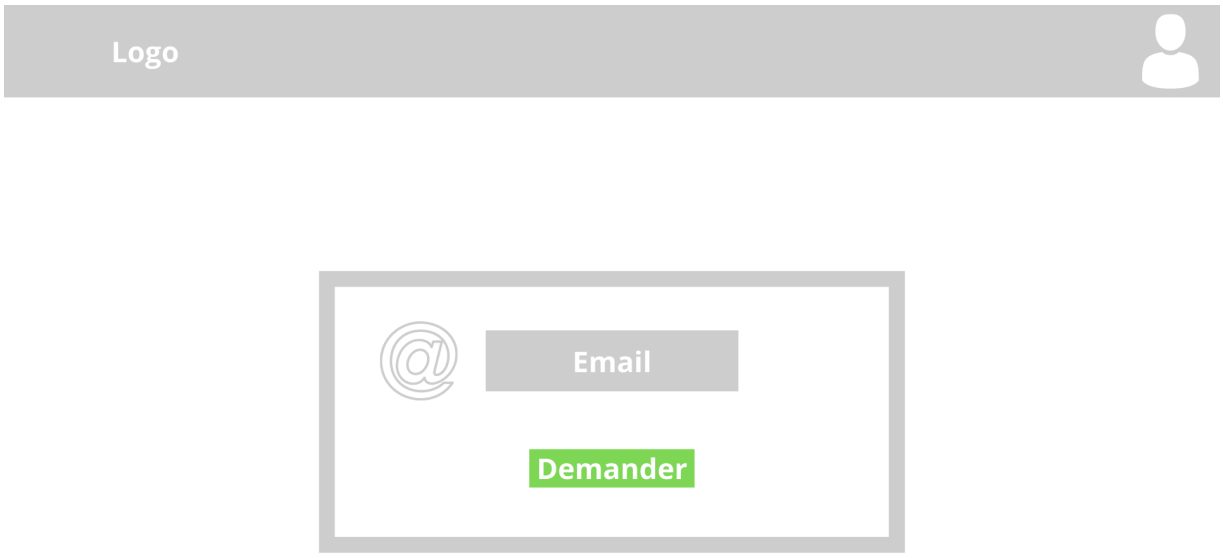

Logo
Gérer boutiques
Gérer tickets
Gérer utilisateurs
Titre
nom de l'utilisateur
Voir
Clore
Titre
nom de l'utilisateur
Voir
Clos
Titre
nom de l'utilisateur
Voir
Clore
Titre
nom de l'utilisateur
Voir
Clore


Logo
Gérer boutiques
Gérer tickets
Gérer utilisateurs
Titre
Contenu
Auteur
Contenu
Date
Auteur
Contenu
Date
Auteur
Contenu
Date
Auteur
Contenu
Date
Rédigez une réponse
Contenu
Envoyer

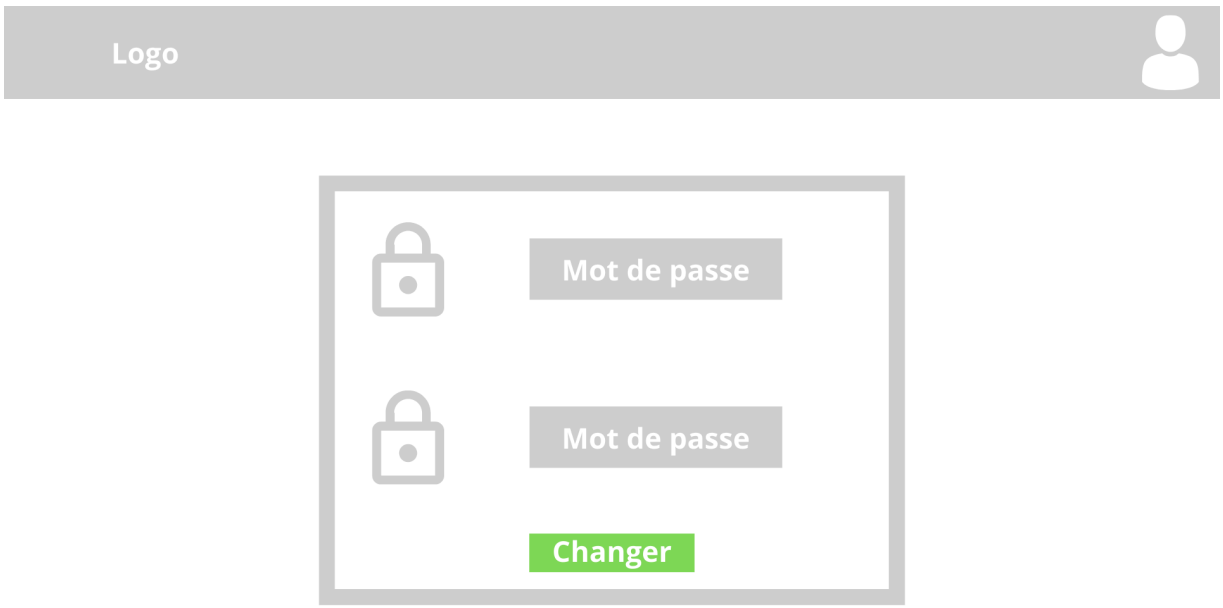
2.5.6 - Utilisateur

Nom	email	Rôle	(Boutique)	Modifier Supprimer
Nom	email	Rôle	(Boutique)	Modifier Supprimer
Nom	email	Rôle	(Boutique)	Modifier Supprimer
Nom	email	Rôle	(Boutique)	Modifier Supprimer

2.5.7 - Récupération de mot de passe



2.5.8 - Changement de mot de passe



2.6 - Droits d'Administration des fonctionnalités

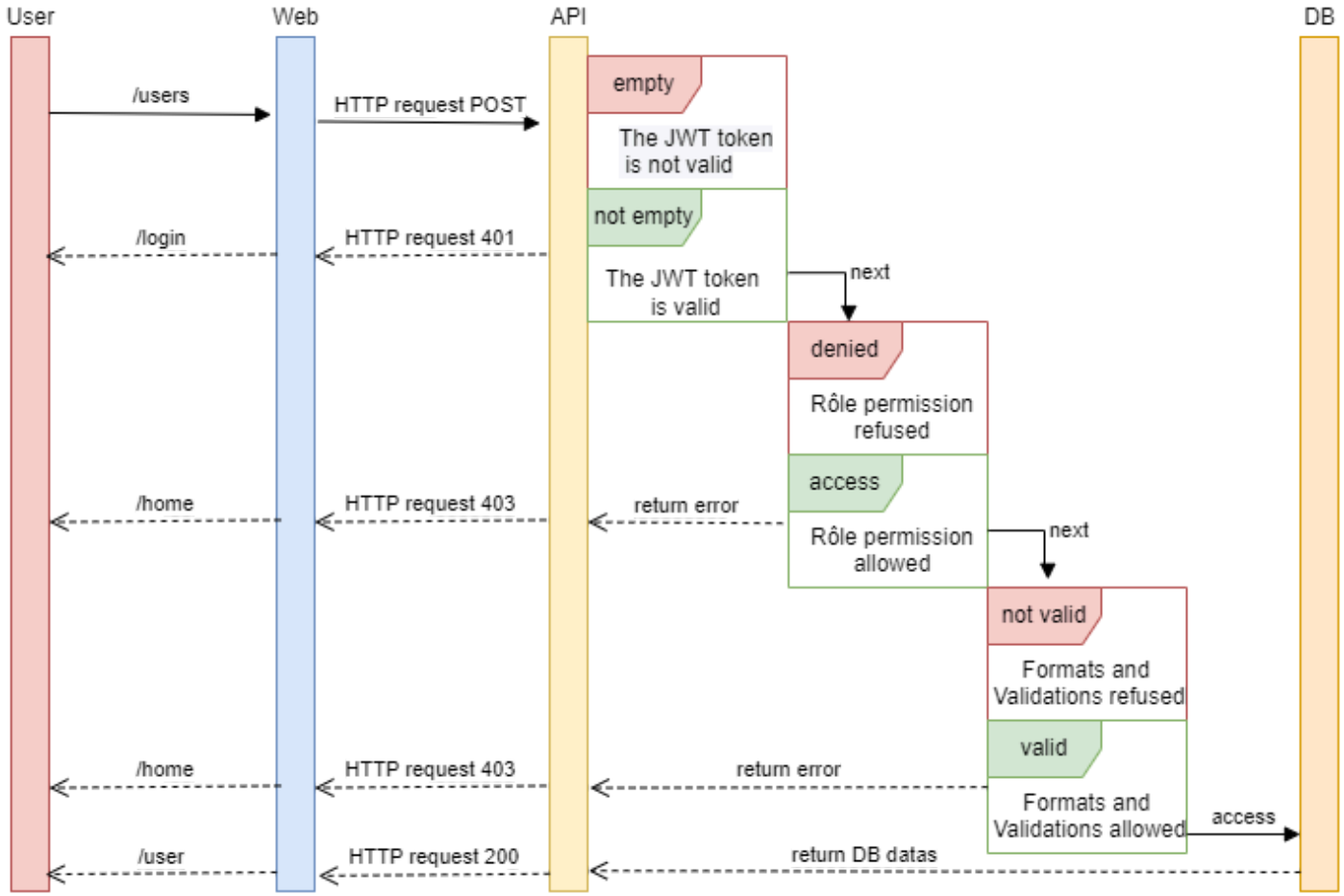
Fonctionnalités / Rôles	Super Admin	Admin	Partenaire	Client
Connexion au site web	Oui	Oui	Oui	Non
Déconnexion	Oui	Oui	Oui	Non
Génération token changement de mot de passe Web	Oui	Oui	Oui	Non
Réinitialisation de mot de passe Web	Oui	Oui	Oui	Non
Créer un utilisateur	Oui	Partenaire uniquement	Non	Non
Voir un utilisateur	Oui	Soi-même, Partenaire et Client uniquement	Soi-même et ses partenaires de boutique	Soi-même
Modifier un utilisateur	Oui	Soi-même, Partenaire et Client uniquement	Soi-même	Soi-même
Supprimer un utilisateur	Oui	Non	Non	Non
Créer une boutique	Oui	Oui	Non	Non
Voir une boutique	Oui	Oui	Les non-supprimées	Les non-supprimées
Modifier une boutique	Non	Non	La sienne si non-supprimées	Non
Supprimer une boutique	Oui	Non	Non	Non
Créer un produit	Non	Non	Oui dans sa boutique	Non

Voir un produit	Oui	Oui	Si non-supprimé et boutique non-supprimée	Si non-supprimé et boutique non-supprimée
Modifier un produit	Non	Non	Oui ceux de sa boutique et non-supprimés	Non
Supprimer un produit	Non	Non	Oui ceux de sa boutique	Non
Créer un ticket	Non	Non	Oui	Oui
Voir un ticket	Oui	Oui	Les siens	Les siens
Clore un ticket	Oui	Oui	Les siens	Les siens
Répondre à un ticket	Oui	Oui	Les siens	Les siens
Voir une commande	Oui	Oui	Celles de sa boutique	Les siennes
Modifier le statut d'une commande	Non	Non	Avancer les statuts d'expédition	Annuler une commande

3 - Partie Technique

3.1 - Conception technique

3.1.1 - Séquence



3.1.2 - Spécificités des droits d'Administration des fonctionnalités

3.1.2.1 - Mobile

Connexion sur l'application mobile : Vérifie que l'utilisateur existe via son email, Vérifie que l'utilisateur est un client. Comparer le mot de passe avec celui en base de données. Si tout est valide, retourne le Token JWT de l'utilisateur ainsi que ses informations personnelles.

Déconnexion: Suppression du Token et fin de la session.

Inscription : Vérifie que les formats du nom, prénom, email et mot de passe soient valides, vérification que l'email n'existe pas. Si valide, hash le mot de passe et envoie par email à l'utilisateur et création de son dossier sur le dossier **"Store"***.

Génération de code changement de mot de passe mobile : Vérifie la validité de l'email, vérifie que l'utilisateur existe et qu'il soit un client. Si valide, cela crée un code de 4 chiffres valable pendant 1h et envoie par email du lien de réinitialisation.

Réinitialisation de mot de passe mobile : Vérifie la validité du mot de passe, celle du Code (existe et non expiré), que l'utilisateur existe et que l'utilisateur soit un client. Si valide, supprime le Code et change le mot de passe du client et envoie par email pour lui confirmer le changement.

Voir son profil : Vérification du Token, vérifie les permissions de l'utilisateur, retourne ses infos.

Modifier son profil : Vérification du Token, vérifie les permissions de l'utilisateur. L'utilisateur peut modifier son mot de passe et sa photo profil, vérifier le nouveau mot de passe et comparer le mot de passe actuel avec celui en base de données. Pour la photo de profil, vérifie la validité de son format et stock l'image dans le dossier de l'utilisateur dans le "Store". Si valide, modifie les informations de l'utilisateur.

Supprimer son compte : Vérification du Token, vérifie les permissions de l'utilisateur. Si valide et qu'il n'a aucune commande en cours, supprime l'utilisateur en base de données, ainsi que son dossier dans le "Store" puis envoie un email à ce dernier pour l'en informer.

Voir une boutique : Vérification du Token, vérifie les permissions de l'utilisateur, retourne la ou les boutique(s) avec les produits ainsi que ses partenaires.

Voir un produit : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que le produit existe bien ainsi que la boutique du produit, si valide, retourne le ou les produit(s) ainsi que sa boutique.

Créer un ticket : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que les formats du titre et du contenu soient valides. Si valide, crée un ticket avec pour statut "Ouvert".

Voir un ticket : Vérification du Token, vérifie les permissions de l'utilisateur, retourne ensuite le ou les ticket(s) contenant les messages qui eux mêmes contiennent un utilisateur.

Clore un ticket : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que le ticket existe et que son utilisateur existe également. Vérifie que le ticket ne soit pas de statut "Clos". Si valide, passe le statut en "Clos" et avertit l'utilisateur possédant le ticket par email.

Répondre à un ticket : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que le ticket existe et que l'utilisateur existe également. Vérifie le format du contenu. Si valide, crée un nouveau message au ticket et envoie par email à la personne possédant le ticket si c'est une réponse venant d'un Admin ou Super-Admin.

Faire une commande : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que les produits existent, vérifie que les quantités sont valides en fonction des stocks et au bon format. Si valide, crée les commandes en définissant le numéro de commande, tout en supprimant les quantités des stocks. Pour finir un envoi par mail vient confirmer la commande à l'utilisateur, ainsi qu'aux boutiques partenaires, joignant les factures en formats pdf, les pdf sont stockés dans le dossier de facture de l'utilisateur dans le "Store".

Voir une commande : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que la commande existe, retournant ainsi la ou les commande(s) contenant les produits en question.

Annuler une commande : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que les boutiques et les commandes attribuées aux numéros et boutiques existent. Vérifie également que chaque commande possède le statut "Validée", si tout est valide, modifie le statut des commandes, réattribue les stocks aux produits et envoie des mails de confirmation d'annulation des commandes.

*** Le dossier du nom de "Store" est le lieu stockant les fichiers des différents acteurs de l'application.**

3.1.2.2 - Web

Connexion au site web : Vérifie que l'utilisateur existe via son email, Vérifie les **permissions de l'utilisateur***. Comparer le mot de passe avec celui en base de données. Si tout est valide, retourne le Token JWT de l'utilisateur ainsi que ses informations personnelles.

Déconnexion: Suppression du Token et fin de la session.

Génération de token changement de mot de passe Web : Vérifie la validité de l'email, vérifie que l'utilisateur existe et les permissions de l'utilisateur. Vérifie que l'utilisateur n'ai aucun Token valide durant 1h pour en générer un nouveau. Si valide, envoie par email du lien de réinitialisation.

Réinitialisation de mot de passe Web : Vérifie la validité du mot de passe, celle du Token (existe et non expiré), que l'utilisateur existe et les permissions de l'utilisateur. Si valide, supprime le Token et change le mot de passe de l'utilisateur et envoie par email pour confirmer le changement à l'utilisateur.

Créer un utilisateur : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que les formats du nom, prénom et email soient valides, vérification que l'email n'existe pas, génération d'un mot de passe aléatoire. Si valide, hash le mot de passe et envoie par email à l'utilisateur son mot de passe temporaire et création de son dossier sur le dossier **"Store"****.

Voir un utilisateur : Vérification du Token, vérifie les permissions de l'utilisateur, retourne le ou les utilisateur(s) ainsi que sa boutique si Partenaire.

Modifier un utilisateur : Vérification du Token, vérifie les permissions de l'utilisateur.
- Sur soi-même, un utilisateur peut modifier son mot de passe et sa photo profil, vérifier le nouveau mot de passe et comparer le mot de passe actuel avec celui en base de donnée. Pour la photo de profil, vérifie la validité de son format et stock l'image dans le dossier de l'utilisateur dans le "Store". Si valide, modifie les informations de l'utilisateur.
- Si admin, peut modifier le nom, le prénom et l'email. vérifie que les formats soient valides puis vérifie que l'email n'existe pas. Si valide, modifie les informations de l'utilisateur.

Supprimer un utilisateur : Vérification du Token, vérifie les permissions de l'utilisateur. Si valide, supprime l'utilisateur en base de données, ainsi que son dossier dans le "Store" puis envoie un email à ce dernier pour l'informer.

Créer une boutique : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie ensuite que l'email et le nom de boutique n'existe pas. Vérifie ensuite que les formats d'email et de nom soient valides. Si valide, crée la boutique ainsi que le dossier de la boutique dans le "Store", puis envoie par la suite un email à l'adresse de la boutique.

Voir une boutique : Vérification du Token, vérifie les permissions de l'utilisateur, retourne la ou les boutique(s) avec les produits ainsi que ses partenaires.

Modifier une boutique : Vérification du Token, vérifie les permissions de l'utilisateur, possibilité de modifier, l'email, le nom, le téléphone, la ville, la rue, le code postal ainsi que le logo de la boutique. Les différentes valeurs se feront vérifier leurs formats et leurs validités et disponibilités pour le nom, l'email, et le téléphone. Si valide, les modifications de la boutique seront réalisées, si le logo a été changé, l'image sera stockée dans le dossier de la boutique dans le "Store".

Supprimer une boutique : Vérification du Token, vérifie les permissions de l'utilisateur. Si valide, passage du statut de la boutique en "Supprimée". Changement également du statut de tous les produits de la boutique en "Supprimée". Mais supprime également définitivement les partenaires affiliés à cette dernière, supprimant leurs dossier dans le "Store". Envoie également par email la confirmation aux partenaires et à la boutique.

Créer un produit : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que la boutique existe, vérifie que les formats du nom, de la description, du prix, du stock ainsi que de l'image du produit soient valides. Si valide, crée un sous-dossier du produit dans le dossier de la boutique dans le "Store".

Voir un produit : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que le produit existe bien ainsi que la boutique du produit, si valide, retourne le ou les produit(s) ainsi que sa boutique.

Modifier un produit : Vérification du Token, vérifie les permissions de l'utilisateur, possibilité de modifier le nom, la description, le prix, le stock ainsi que l'image du produit. Les différentes valeurs se feront vérifier leurs formats et leurs validités. Si valide, les modifications du produit seront réalisées, si l'image a été changée, l'image sera stockée dans le sous-dossier du produit lui-même dans le dossier de la boutique dans le "Store".

Supprimer un produit : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que le produit existe. Si valide, modifie le statut du produit en "Supprimé".

Créer un ticket : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que les formats du titre et du contenu soient valides. Si valide, crée un ticket avec pour statut "Ouvert".

Voir un ticket : Vérification du Token, vérifie les permissions de l'utilisateur, retourne ensuite le ou les ticket(s) contenant les messages qui eux mêmes contiennent un utilisateur.

Clore un ticket : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que le ticket existe et que son utilisateur existe également. Vérifie que le ticket ne soit pas de statut "Clos". Si valide, passe le statut en "Clos" et avertit l'utilisateur possédant le ticket par email.

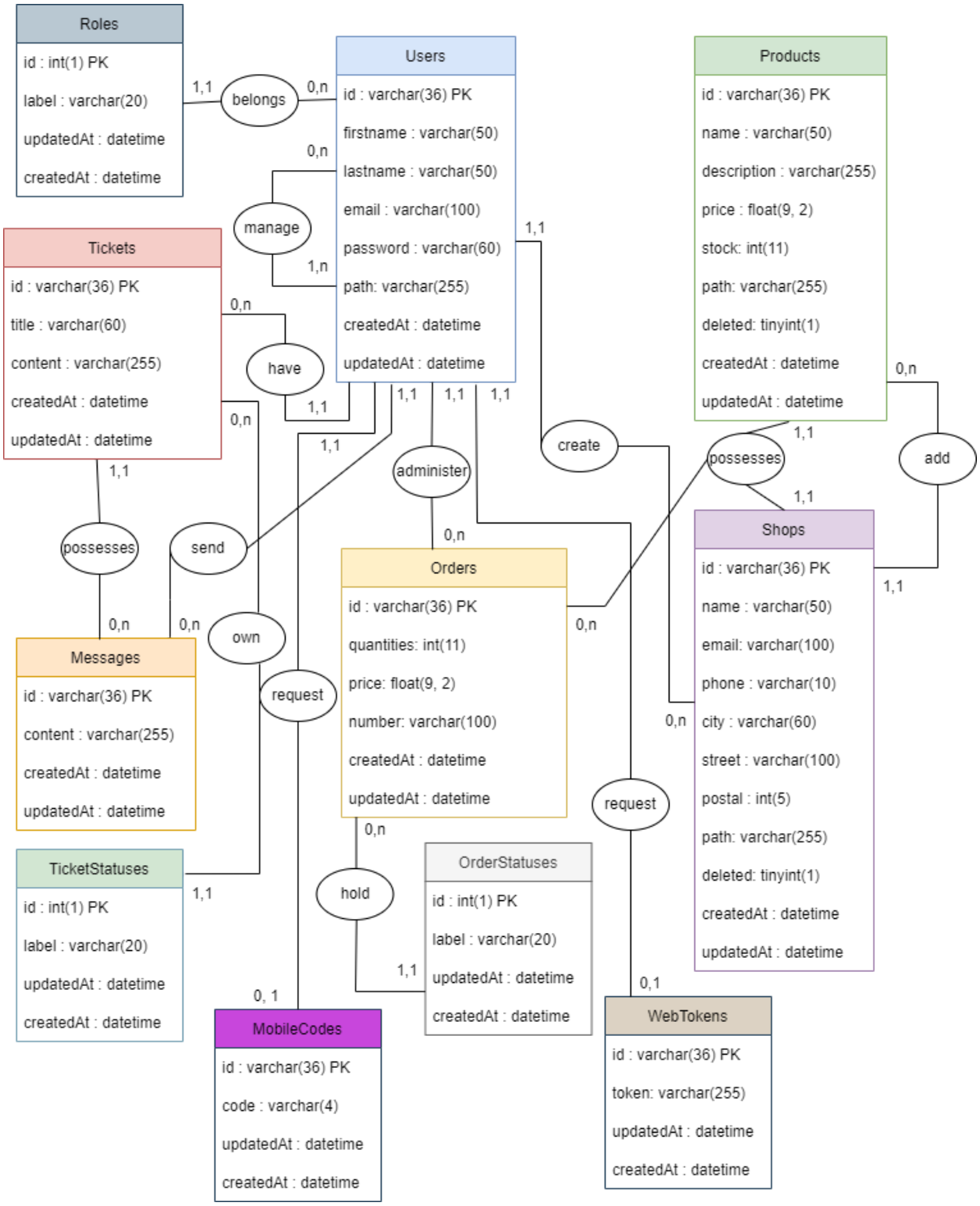
Répondre à un ticket : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que le ticket existe et que l'utilisateur existe également. Vérifie le format du contenu. Si valide, crée un nouveau message au ticket et envoie par email à la personne possédant le ticket si c'est une réponse venant d'un Admin ou Super-Admin.

Voir une commande : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que la commande existe, retournant ainsi la ou les commande(s) contenant les produits en question ainsi que l'utilisateur ayant réalisé la commande.

Modifier le statut d'une commande : Vérification du Token, vérifie les permissions de l'utilisateur, vérifie que la commande existe. Si valide, modifie en fonction des permissions de l'utilisateur le statut de la commande et envoie ainsi par email les modifications concernant le statut de la commande du client.

*** Pour les permissions se référer aux droits d'Administration fonctionnels.**
**** Le dossier du nom de "Store" est le lieu stockant les fichiers des différents acteurs de l'application.**

3.2 - Modélisation de données (MCD)



4 - Tests

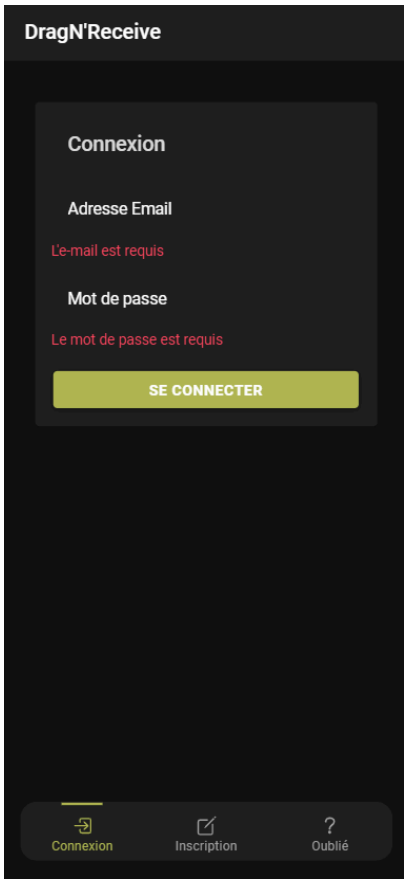
4.1 - Tests unitaires

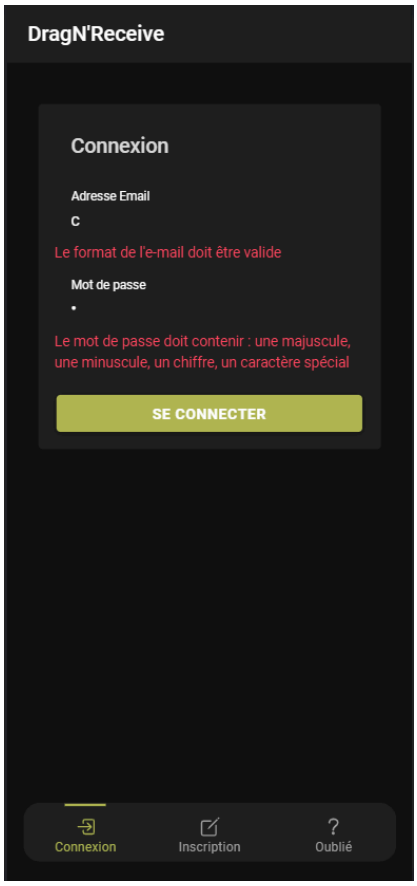
Pour le passage des tests unitaires de mon API, j'ai choisi d'utiliser Mocha et Chai, des librairies JavaScript permettant d'effectuer des tests sur différentes fonctionnalités et vérification de l'API, notamment via requêtes HTTP.

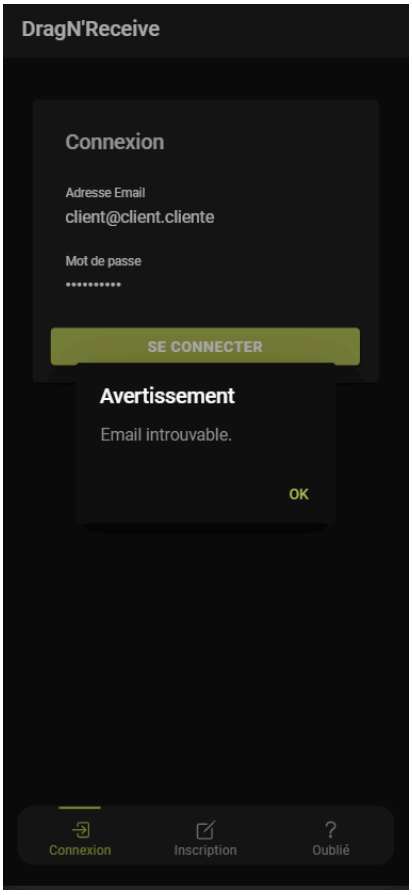
```
Exists
  Email exist
    ✓ Email exist in Database (59ms)
    ✓ Email dont exist in Database
  Shop exist
    ✓ Shop exist in Database
    ✓ Shop dont exist in Database

Formats
  Email format
    ✓ Email format is valid
    ✓ Email should be a string
    ✓ Email should be "string@string.string" format
    ✓ Email length should be less or equal to 100
  Password format
    ✓ Password format is valid
    ✓ Password should be a string
    ✓ Password should have one upper case character
    ✓ Password should have one lower case character
    ✓ Password should have one number character
    ✓ Password should less or equal to 60 characters
    ✓ Password should be more than 7 characters

Requests API
  GET /api/currentUser
    ✓ Response is an object
    ✓ Response is null because no token in header
  POST /api/loginClient
    ✓ Sadmin, Admin or Partner cant be loggedin
    ✓ Wrong password cannot loggedin
    ✓ Wrong email cannot loggedin
    ✓ Valid Client can be logged in
```

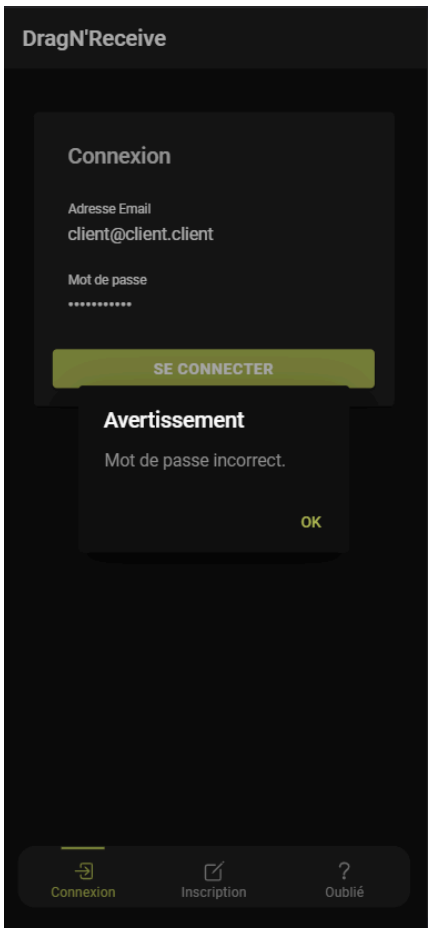
4.2 - Tests de validation

4.2.1 - Connexion



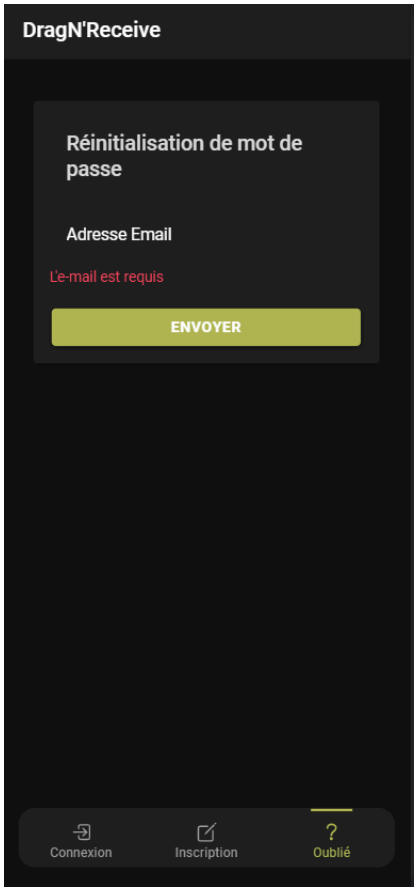


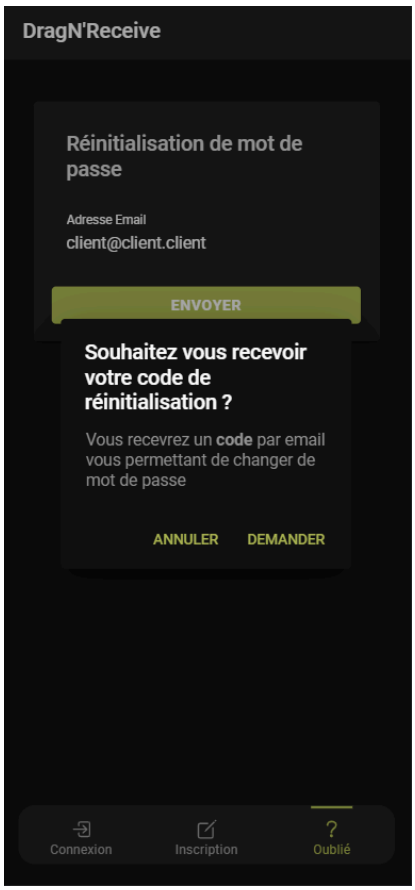


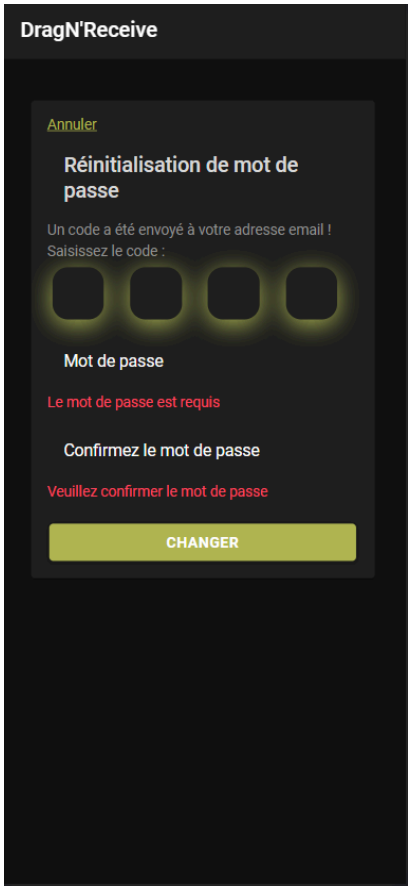

DragN'Receive
Connexion
Adresse Email
client@client.client
Mot de passe
••••••••••
SE CONNECTER
Avertissement
Mot de passe incorrect.
OK
Connexion
Inscription
Oublié

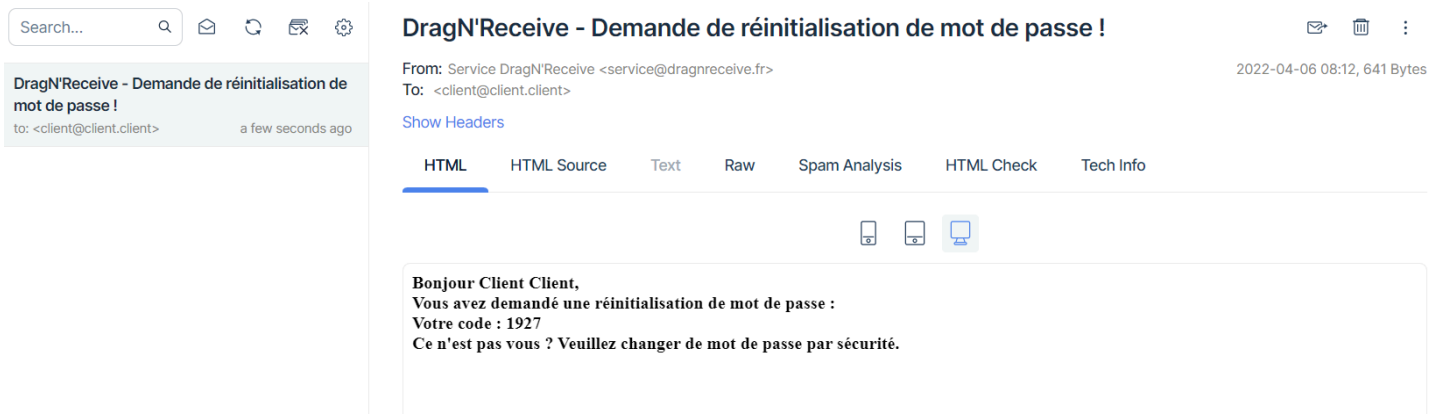

Cherchez des produits / boutiques
Home

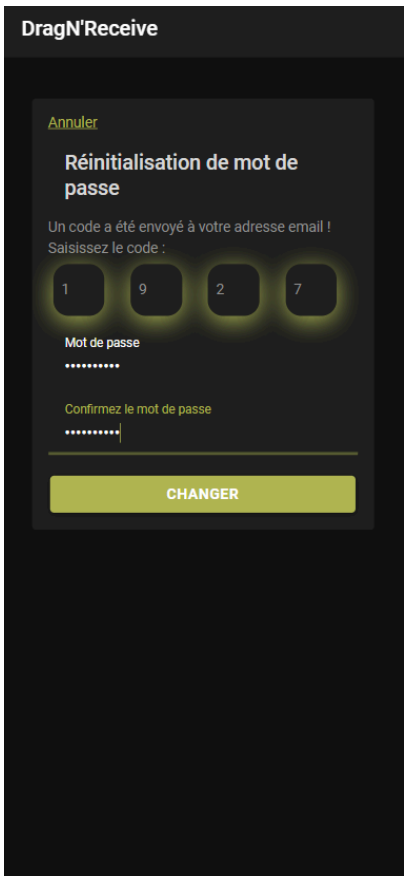
4.2.2 - Réinitialisation de mot de passe

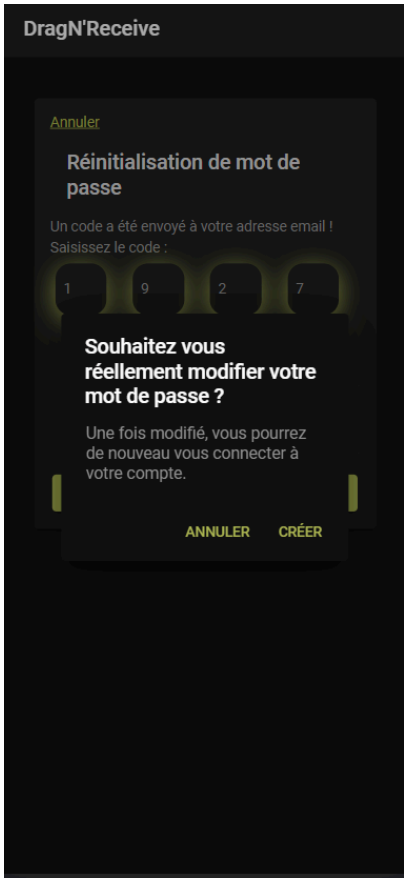


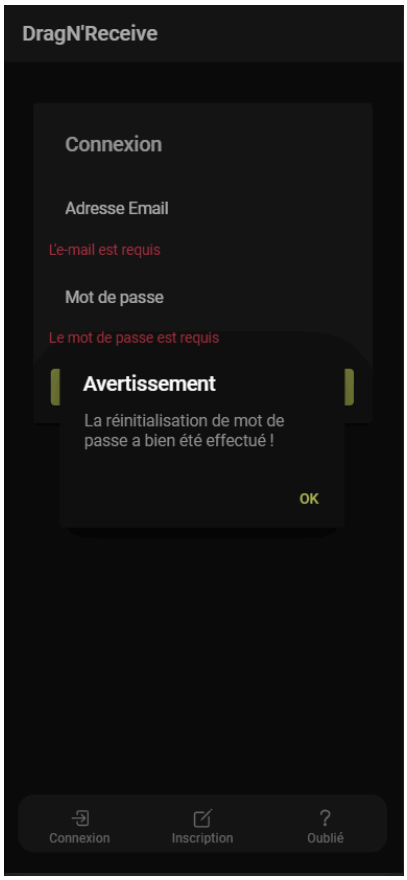







DragN'Receive
Annuler
Réinitialisation de mot de passe
Un code a été envoyé à votre adresse email !
Saisissez le code :
1
9
2
7
Mot de passe
Confirmez le mot de passe
CHANGER


DragN'Receive
Annuler
Réinitialisation de mot de passe
Un code a été envoyé à votre adresse email !
Saisissez le code :
1
9
2
7
Souhaitez vous réellement modifier votre mot de passe ?
Une fois modifié, vous pourrez de nouveau vous connecter à votre compte.
ANNULER
CRÉER


DragN'Receive
Connexion
Adresse Email
L'e-mail est requis
Mot de passe
Le mot de passe est requis
Avertissement
La réinitialisation de mot de passe a bien été effectué !
OK
Connexion
Inscription
Oublié