Projet personnel encadré 2 :

DragN'Receive

Dev'iliser

CLEMENT Pierre de Dev'iliser

01/03/2022

Pour PROMEO - BTS Services informatiques aux organisations - Session 2022

1 - Définition de la situation professionnelle

Le contexte défini ci-après est une situation fictive mais réaliste d'un projet d'entreprise visant à mettre en place une solution logicielle.

1.1 - Présentation de l'entreprise cliente

L'entreprise cliente, nommée DragN'Receive, est une société fictive de service Click & Collect implanté à Compiègne.

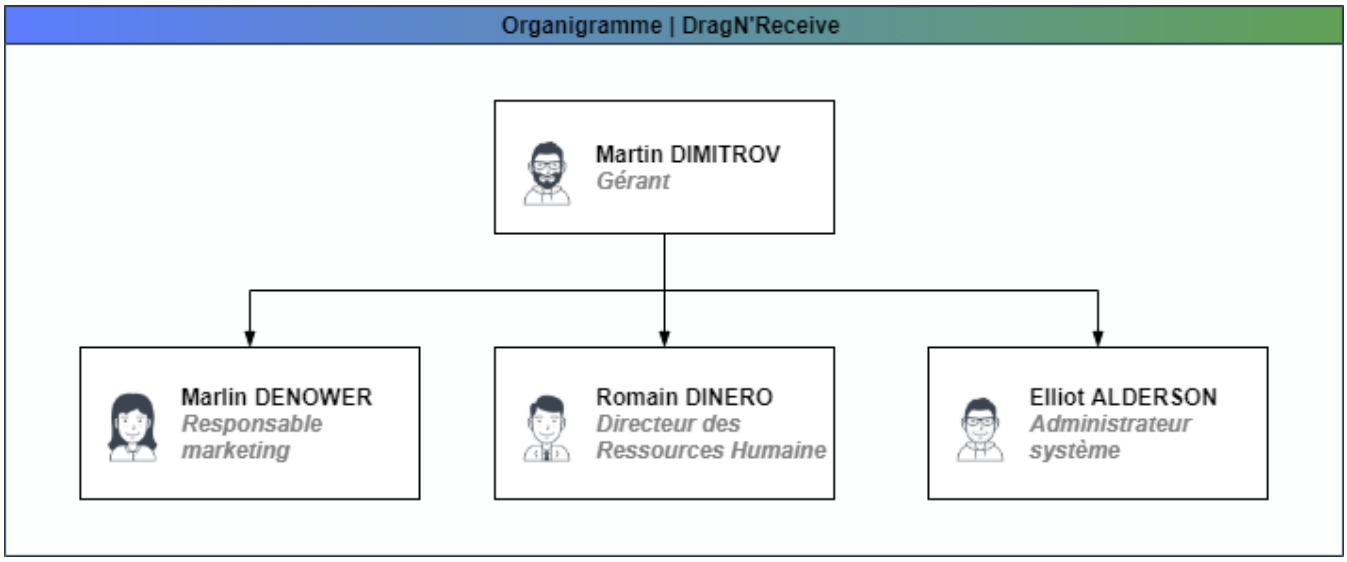
1.1.1 - Généralités

L'entreprise DragN'Receive est une SARL composée de 20 à 35 salariés, cette entreprise possède plusieurs contacts et partenaires dans les domaines de la restauration, de la distribution ainsi que des petits commerces.

La création de l'entreprise était d'abord à des fins opportunistes lors de la période de confinement puis s'est vu devenir un avantage primordial pour les personnes avec peu de disponibilités, les personnes à handicap et à mobilités réduites. La société se tourne également vers le développement durable avec des partenaires primant des produits possédant des impacts environnementaux faibles ainsi qu'un service de logistique verte, afin de réduire son impact sur l'environnement.

Leur capital social durant la période de 2020-2021 s'élève à 16 000 euros, pour un chiffre d'affaires de 245 000 euros.

Voici l'organigramme de l'entreprise DragN'Receive :



1.1.2 - Environnement technique

L'environnement technique de DragN'Receive est formé de plusieurs composants :

- Systèmes d'exploitation des appareils : Windows 10, Androids, IOS
- Serveurs : Dédiés
- Base de données : MySQL hébergé sur un serveur dédié
- Nombre d'utilisateurs simultanés potentiels : 500
- Site Web sous Vue (hébergé sur le serveur)
- API NodeJS (hébergé sur le serveur)

1.2 - Présentation du prestataire de service

Le prestataire pour ce projet sera l'entreprise Dev'iliser, étant une société spécialisée dans le développement de solution logicielle et applicative depuis plus de 16 années. Dans ce cadre d'activité, elle s'occupera du développement, suivis de la mise en service ainsi que du service technique de l'application mobile demandée par DragN'Receive. La principale personne chargée de la réalisation de ce projet ainsi que la communication avec la société DragN'Receive sera Clément Pierre, ayant réalisé le précédent projet de site web pour DragN'Receive.

2 - Conception graphique

2.1 - Brief créatif

Le graphisme du site se doit de retranscrire l'identité de DragN'Receive tout en s'adaptant aux normes d'une application mobile. Pour ce faire nous conserverons les touches de couleurs représentative des logos de la société que nous poserons sur un style graphique adapté aux smartphones.

2.2 - Aperçu de la charte graphique

2.2.1 - Logos

Durant le cadre du précédent projet, la réalisation de divers logos à l'effigie de l'entreprise ont étés réalisés :

2.2.2 - Palette de couleurs

Concernant les couleurs, nous utiliserons ces références de couleurs basés sur la charte graphique de DragN'Receive, ainsi que d'autres tonalités comme réalisé pour le site web précédemment, celles-ci seront les principales :

#91a16a

#afb854

#afb891

#014d5a

#24646f

#4a7f88

2.2.3 - Typographies

Nous avons choisi comme typographie Roboto, utilisée sur le site web, elle est très sobre et facile à lire, critères majeur pour une application sur un téléphone.

3 - Cahier des charges fonctionnel

3.1 - Présentation générale

La société DragN'Receive souhaite créer un système de Click & Collect à l'échelle de toute l'agglomération de Compiègne, en ayant pour partenaires des marchés et commerces centrés principalement sur la vente de denrées alimentaires et de produits de grande consommation.
Un site web permettant la gestion des boutiques partenaires, permettant ainsi à ces dernières de proposer leurs produits et d' ajouter de nouveaux produits et leurs stocks tout en gérant les commandes clients a donc été réalisé. Une API a également été réalisée et sera donc à faire évoluer.

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.

3.2 - 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 avec le client.

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.

3.3 - Solutions envisagées

Les solutions envisagés permettant la réponses des divers besoin 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.

3.4 - Solution retenue

3.4.1 - Présentation des choix

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

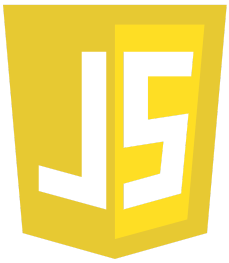
- D'utiliser le serveur dédié, permettant la mise en place de l'application mobile, de plus la base donnée et l'API créé pour le site web y sont stockés et seront utilisés pour ce projet.
- Nous avons retenu comme langage de programmation le JavaScript avec l'utilisation du framework Vue, également utilisé 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.

3.4.2 - Benchmark

3.4.2.1 - Langage de programmation




	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

3.4.2.2 - Framework



	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

3.4.3 - Explication des choix

Notre équipe à donc choisis d'utiliser le langage JavaScript qui, pour la réalisation du projet, est le plus approprié, utilisable avec de multiples framework, 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 choisis 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 choisis en Back-End le framework ExpressJS basé sur du NodeJS puisque nous allons utiliser l'API développée durant les travaux du site web.

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 choisis 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.

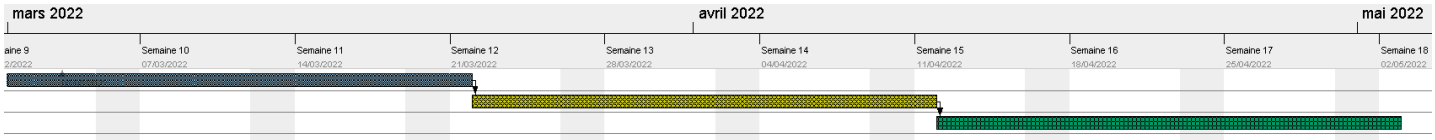
***Back-End :** Partie caché de l'utilisateur, permettant la gestion de la base donnée et les interactions de cette dernière permettant la communication avec le Front-End

****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.

4 - Analyses et estimations

4.1 - Planification

Nom	Date de début	Date de fin
Rédaction du Cahier des charges	01/03/2022	21/03/2022
Evolution de l'API	22/03/2022	11/04/2022
Développement mobile	12/04/2022	02/05/2022



4.2 - Budgétisation

Quantité	Description	Réduction	Coût*
1	Création d'une application mobile permettant l'achat par des clients des produits partenaires ainsi que la gestion et suivis de leurs commandes.	-	2 160 €
1	Évolution de l'API pour le Back-End		2 160 €
1	Adaptation de la charte graphique sur mobile	-	800 €
1	Référencement	-	350 €
1	Service technique	-	350 €

Total :

5 820 €

** Les coûts horaires sont de 18 euros.*

5 - Conception fonctionnelle

5.1 - Dictionnaire de données

5.1.1 - 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

5.1.2 - 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

5.1.3 - 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

5.1.4 - 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

5.1.5 - 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

5.1.6 - 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

5.1.7 - 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

5.1.8 - 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

5.1.9 - 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

5.1.10 - Web Token

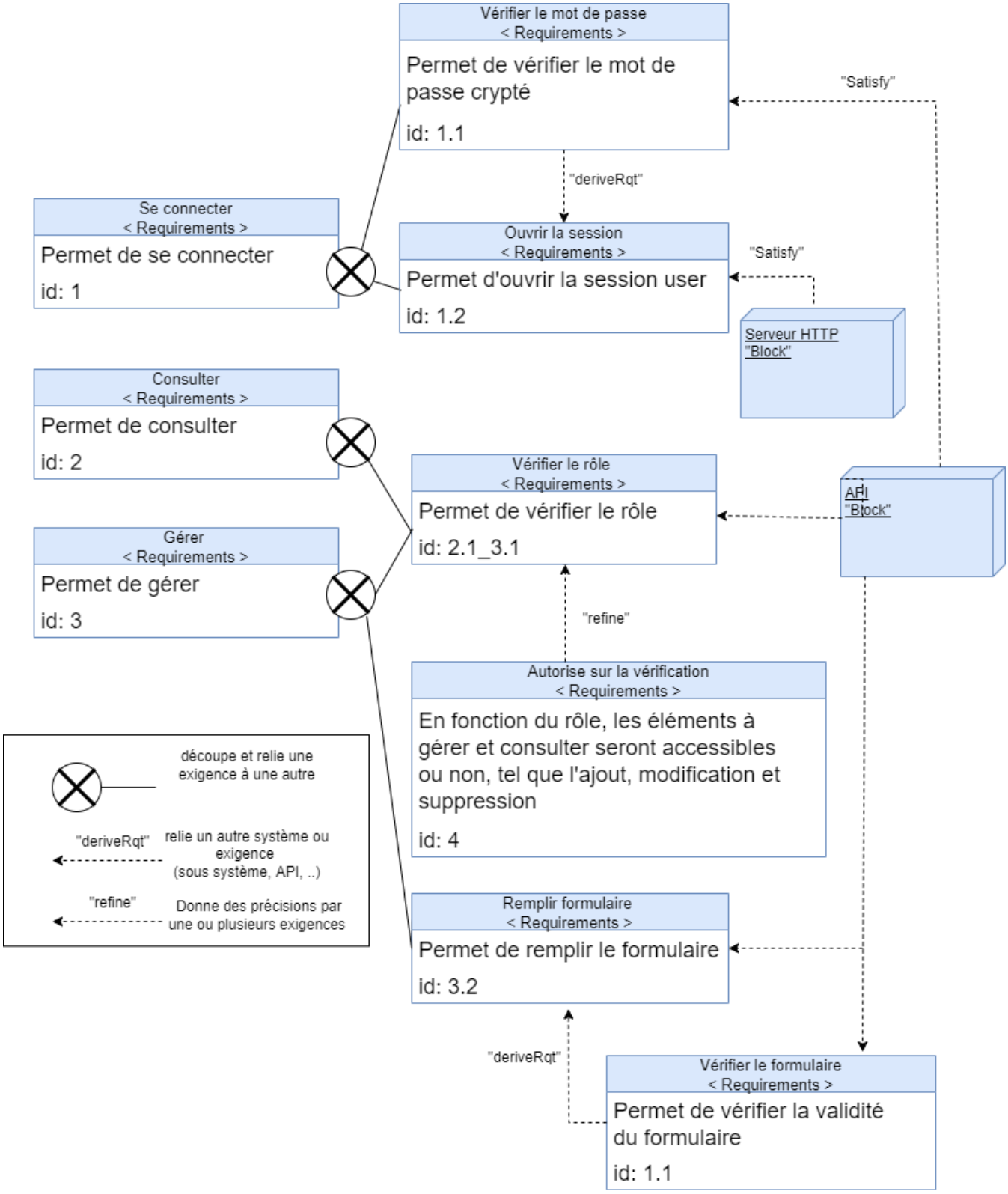
Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.
token	varchar	255	Obligatoire	token généré aléatoirement
createdAt	datetime	yyyy-mm-ddThh :mm:ss.000Z	Obligatoire	date de création
updatedAt	datetime	yyyy-mm-ddThh :mm:ss.000Z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

5.1.11 - Mobile Code

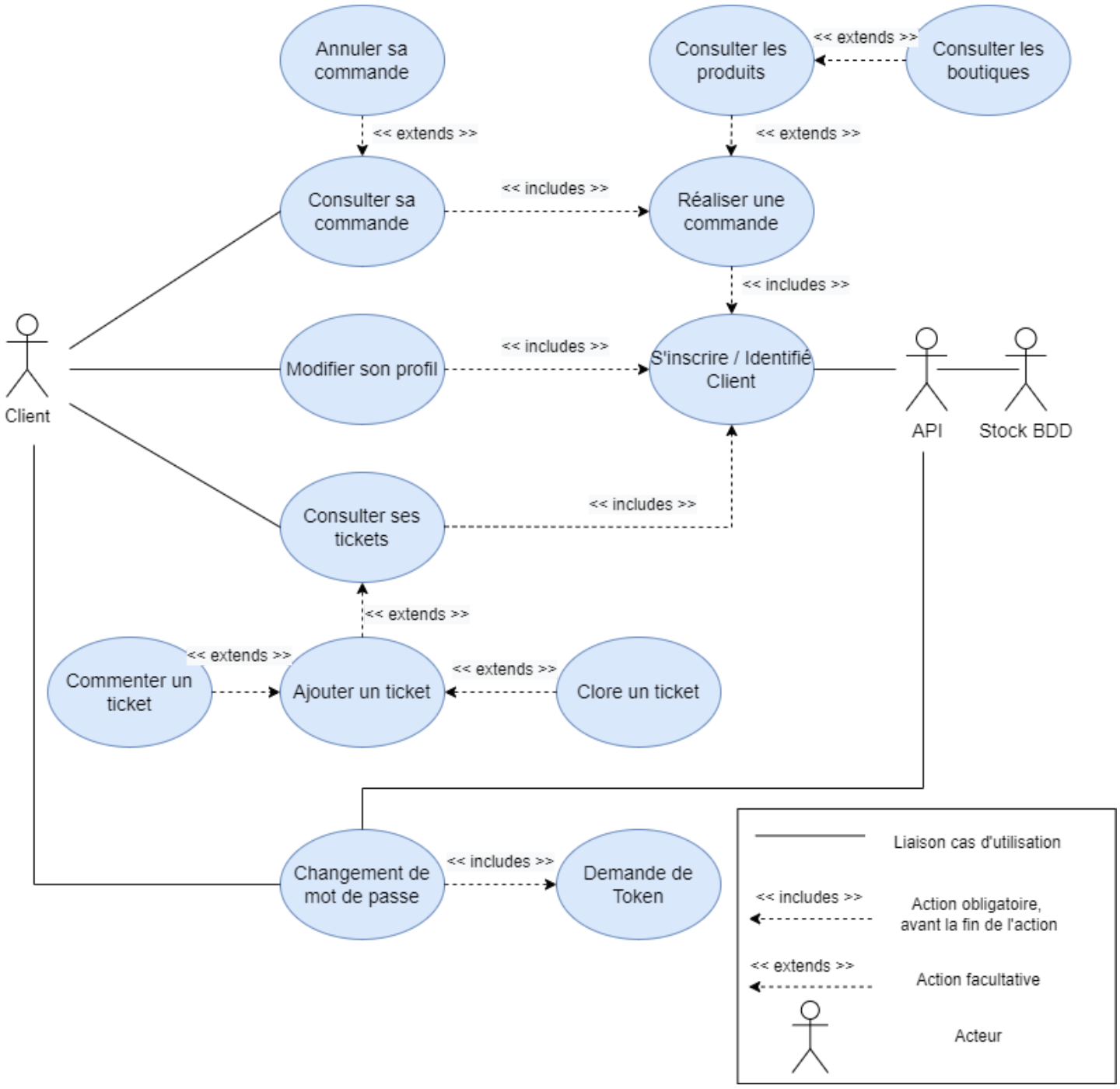
Champs	Types	Tailles	Contraintes	Descriptions
id	varchar	36	PK, Obligatoire	Chaîne de caractère unique générée aléatoirement.

code	varchar	4	Obligatoire	code généré aléatoirement
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

5.2 - Exigences



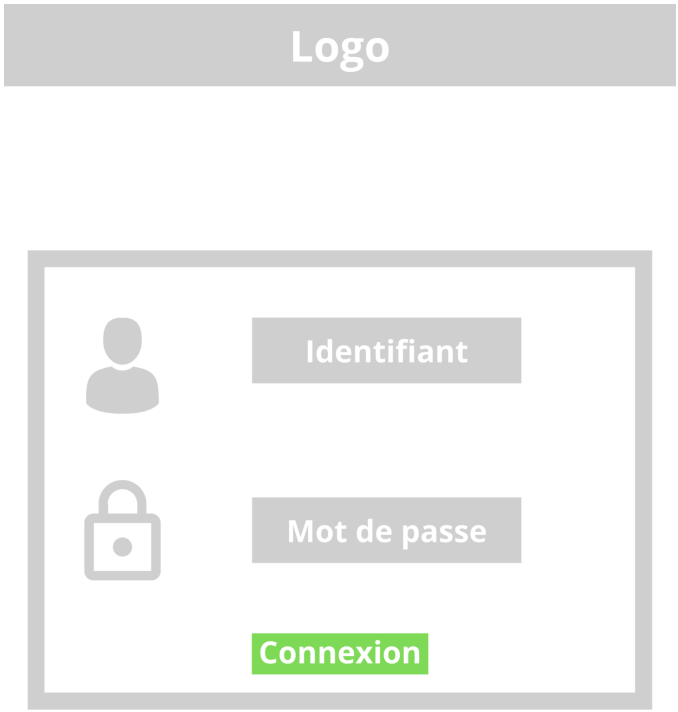
5.3 - Cas d'utilisations



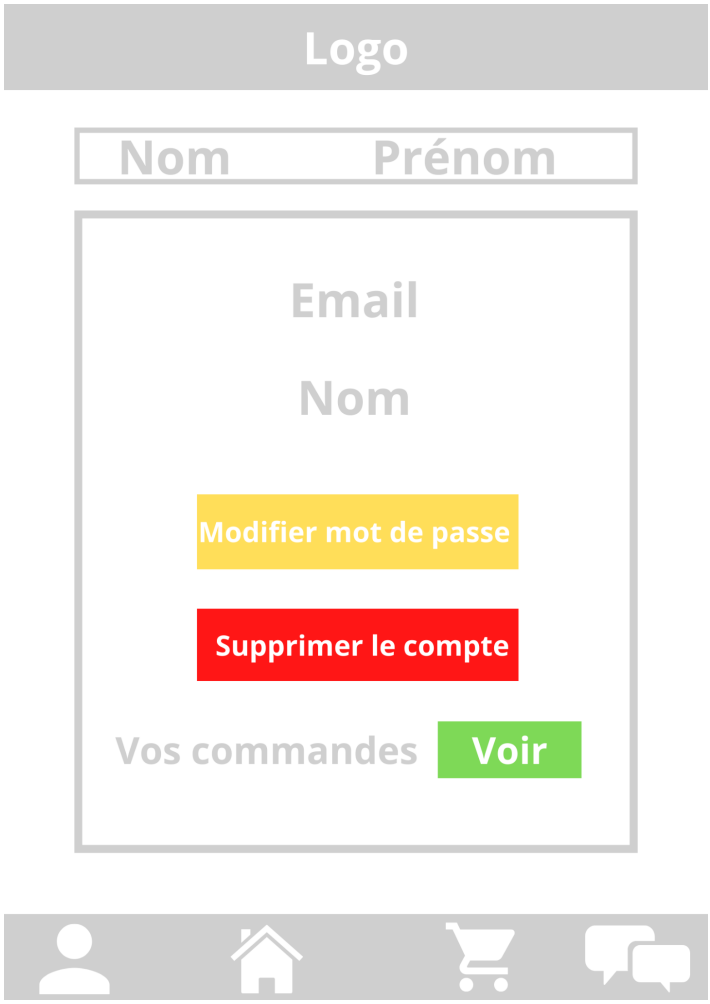
5.4 - Maquettes

⚠ Les maquettes suivantes sont des aperçus de la basse fidélité de l'application mobile. 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.

5.4.1 - Connexion



5.4.2 - Profil



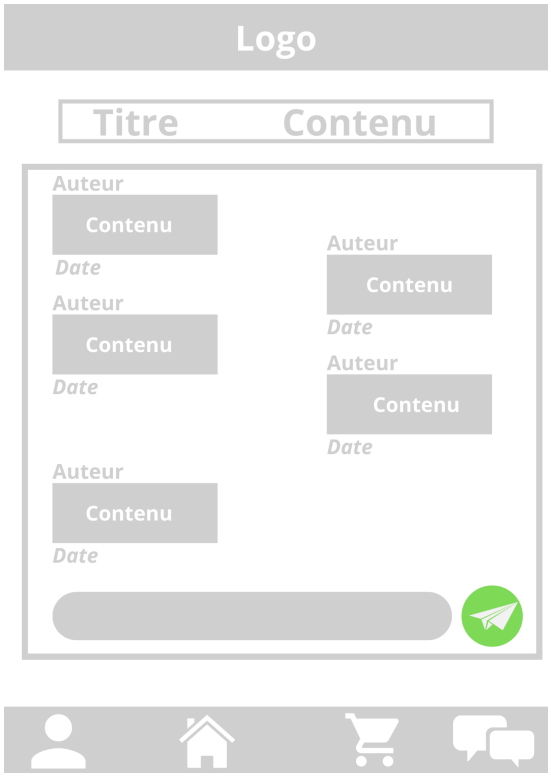
5.4.2 - Boutique

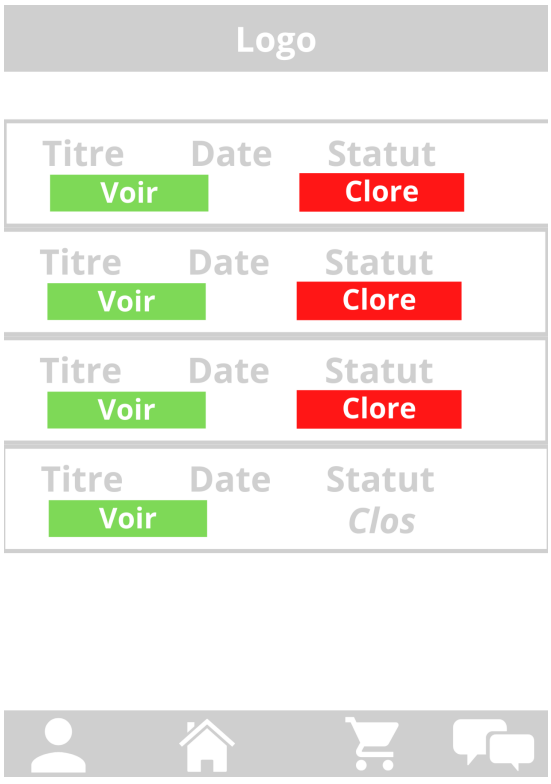
5.4.3 - Commande


Logo
Numéro de commande
Statut de commande | Nom de la boutique
Nom
x Quantité
Annuler
Statut de commande | Nom de la boutique
Nom
x Quantité
Logo
Numéro de commande
...
Voir
Statut
Date
Numéro de commande
Voir
Statut
Date
Numéro de commande
Voir

5.4.4 - Produit

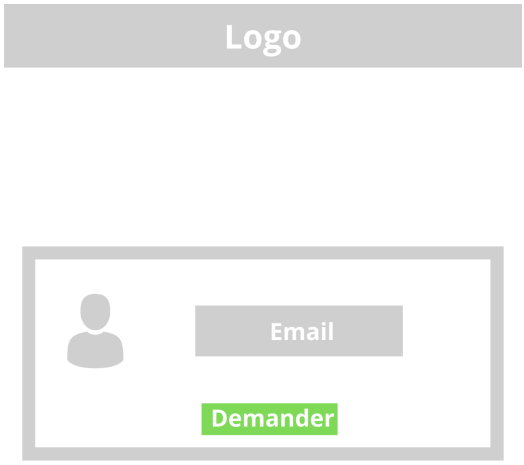
5.4.5 - Ticket



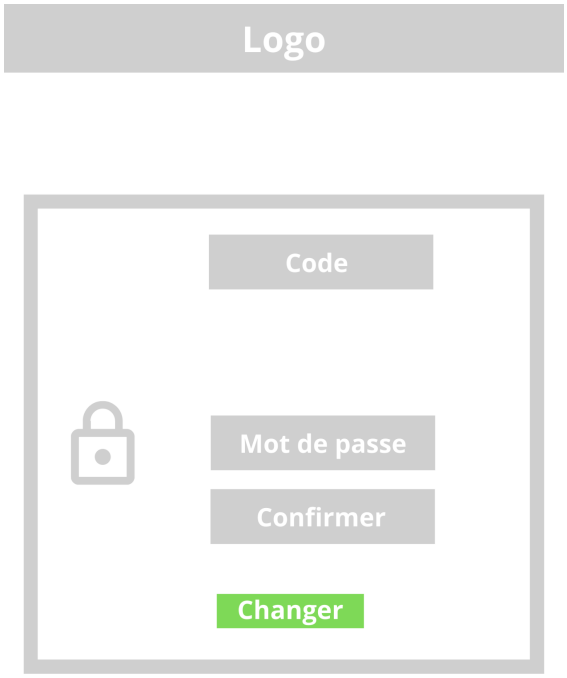


5.4.6 - Inscription

5.4.7 - Récupération de mot de passe



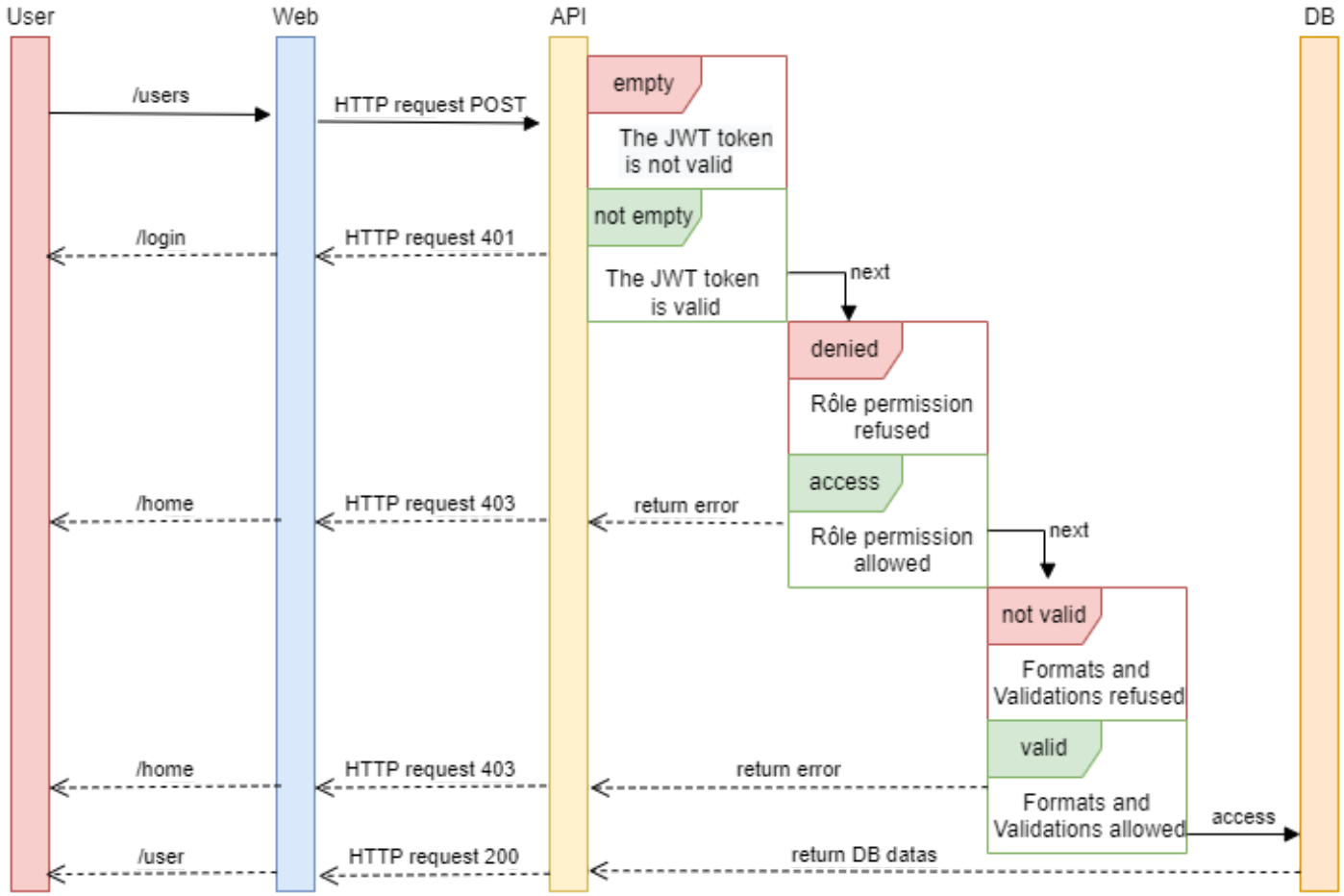
5.4.8 - Changement de mot de passe



6 - Cahier des charges technique

6.1 - Conception technique

6.1.1 - Séquence



6.1.2 - Spécificités des droits d'Administration des fonctionnalités

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. **(Pas encore développé côté front)**

Inscription : Vérifie que les formats du nom, prénom, email et mot de passe soient valide, 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"***. **(Pas encore développé côté front)**

Génération de code changement de mot de passe mobile : // décrire **(Pas encore développé côté back / front)**

Réinitialisation de mot de passe mobile : // décrire **(Pas encore développé côté back / front)**

Voir son profil : Vérification du Token, vérifie les permissions de l'utilisateur, retourne ses infos. **(Pas encore développé côté front)**

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. **(Pas encore développé côté front)**

Supprimer son compte : 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'en informer. **(Pas encore développé côté front)**

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. **(Pas encore développé côté front)**

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. **(Pas encore développé côté front)**

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". **(Pas encore développé côté front)**

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. **(Pas encore développé côté front)**

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. **(Pas encore développé côté front)**

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. **(Pas encore développé côté front)**

Faire une commande : // décrire **(Pas encore développé côté back / front)**

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.

Annuler une commande : // décrire **(Pas encore développé côté back / front)**

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

6.2 - Environnement technologique

6.2.1 - Hardware

Pour la réalisation de ce projet l'environnement technologique utilisé a été :

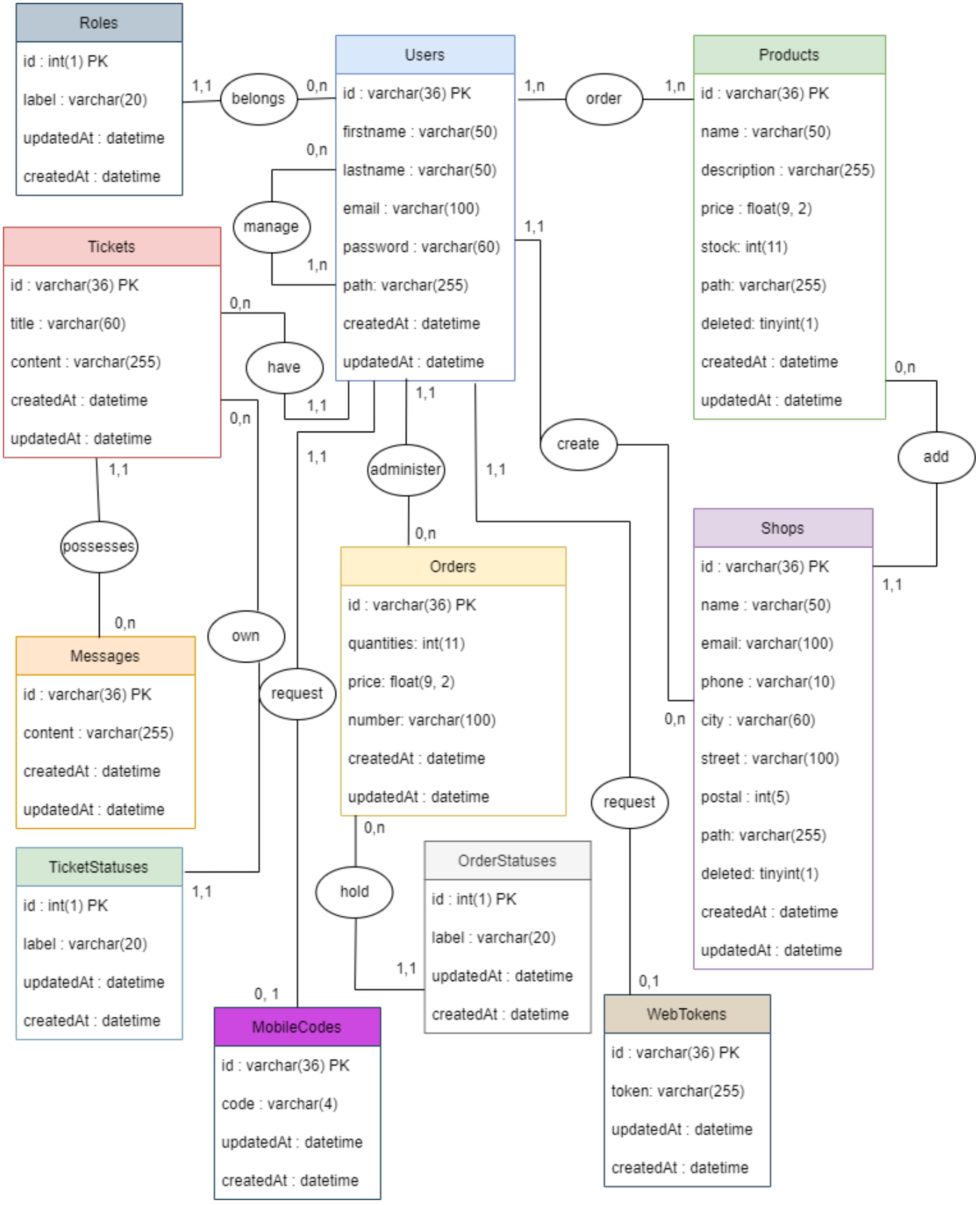
- Ordinateur fixe et portable
- Clavier, souris, écran
- Box Wifi

6.2.2 - Software

La conception de ce projet nous a fait utiliser les outils suivants :

- Windows comme système d'exploitation
- Visual Studio Code comme éditeur de texte et invite de commandes
- Android Studio pour l'émulation d'un Google Pixel 5 (393 x 851)
- GitHub pour gestion de code et de version mais aussi son stockage
- Postman pour faire des requêtes HTTP sur l'API
- NPM, gestionnaire de paquets / librairies JavaScript
- NodeJS pour JavaScript
- Ionic comme framework mobile, ainsi que Capacitor pour construire le projet sur Android et IOS
- Google Chrome pour tester l'application via navigateur web
- Draw.io pour la réalisation des diagrammes
- Jimdo pour la conception des logos
- Canva pour la conception des maquettes
- Gantt pour la planification des travaux
- Google doc pour la rédaction du cahier des charges, son stockage et sa gestion de versions

6.3 - Modélisation de données


Roles
id : int(1) PK
label : varchar(20)
updatedAt : datetime
createdAt : datetime
1,1
belongs
0,n
Users
id : varchar(36) PK
firstname : varchar(50)
lastname : varchar(50)
email : varchar(100)
password : varchar(60)
path: varchar(255)
createdAt : datetime
updatedAt : datetime
1,n
order
1,n
Products
id : varchar(36) PK
name : varchar(50)
description : varchar(255)
price : float(9, 2)
stock: int(11)
path: varchar(255)
deleted: tinyint(1)
createdAt : datetime
updatedAt : datetime
0,n
manage
1,n
Tickets
id : varchar(36) PK
title : varchar(60)
content : varchar(255)
createdAt : datetime
updatedAt : datetime
0,n
have
1,1
1,1
create
0,n
add
1,1
administer
1,1
1,1
possesses
0,n
Shops
id : varchar(36) PK
name : varchar(50)
email: varchar(100)
phone : varchar(10)
city : varchar(60)
street : varchar(100)
postal : int(5)
path: varchar(255)
deleted: tinyint(1)
createdAt : datetime
updatedAt : datetime
1,1
0,n
Orders
id : varchar(36) PK
quantities: int(11)
price: float(9, 2)
number: varchar(100)
createdAt : datetime
updatedAt : datetime
0,n
own
Messages
id : varchar(36) PK
content : varchar(255)
createdAt : datetime
updatedAt : datetime
request
request
0,n
hold
OrderStatuses
id : int(1) PK
label : varchar(20)
updatedAt : datetime
createdAt : datetime
TicketStatuses
id : int(1) PK
label : varchar(20)
updatedAt : datetime
createdAt : datetime
1,1
1,1
0, 1
0,1
MobileCodes
id : varchar(36) PK
code : varchar(4)
updatedAt : datetime
createdAt : datetime
WebTokens
id : varchar(36) PK
token: varchar(255)
updatedAt : datetime
createdAt : datetime

7 - Réalisation technique (à faire)

7.1 - Les points forts du projet

7.2 - Les problèmes rencontrés

8 - Tests (à faire)

8.1 - 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.

8.2 - Tests de validation

8.2.1 - Connexion

8.2.2 - Réinitialisation de mot de passe

9 - Indicateurs de performance (à faire)

9.1 - Analyse de la planification mise à jour

9.2 - Analyse du coût global

Quantité	Description	Réduction	Coût
1	Création d'une application mobile permettant l'achat par des clients des produits partenaires ainsi que la gestion et suivis de leurs commandes.	-	€
1	Évolution de l'API pour le Back-End		€
1	Adaptation de la charte graphique sur mobile	-	€
1	Référencement	-	€

1	Service technique	-	350 €

Total :

€

10 - Conclusion (à faire)