Dossier Compte-rendu

PROJET SMART H4111 - FAIS-MES-COURSES.FR

Historique des versions

1.0	Création du document	Zifan YAO	18/04/18
1.1	Ajout des exigences fonctionnelles	Yuan YE	21/04/18
1.2	Modification exigences	Hugo MARTIN	21/04/18
1.3	Passage en dossier compte-rendu	Zifan YAO	02/05/18

Sommaire

Introduction

Ce document présente les contraintes et les exigences vis-à-vis du projet SMART **fais-mes-courses.fr**. Des explications sur les cas d'utilisation, les exigences fonctionnelles et non-fonctionnelles seront décrites ci-dessous. Une présentation des exigences techniques sera également introduite en fin de document.

Objectifs

Nous proposons une application web destinée aux habitants de résidences en ville ou des résidents appartenant à des complexes immobiliers apparents (HLM etc…). La population visée sera constituée des étudiants et jeunes actifs principalement, mais pourra cibler les personnes âgées ou les personnes avec des difficultés pour se déplacer en ville.

Cette application propose aux habitants la possibilité de commander des articles de magasins autour de chez eux et de se les faire livrer à leur résidence sous 24H. Des coursiers contractuellement rattaché à notre entreprise réalisent les achats à la place des habitants et déposent les articles commandés dans des casiers dédiés à cet effet, dans les locaux communs de la résidence (hall d'entrée, hall d'accueil etc…). Ces casiers seront accessibles uniquement par les habitants et les coursiers en charge des commandes.

Le système proposera également un service pour les commerçants : ils pourront créer une vitrine virtuelle sur notre application et mettre en avant leurs produits. Notre système inclut les commerces de proximité (épiciers, boulangers etc…) et les grandes surfaces (supermarchés, grandes marques...).

Le client pourra consulter la vitrine des commerces proches de chez lui et constituer un panier d'articles pouvant provenir de une ou plusieur vitrine(s). Une limite sur la taille du panier sera imposée.

A partir de son panier d'articles, le client pourra passer commande, se traduisant par une transaction. Dès lors, une livraison sera créée puis délivrée par un coursier en fin de journée.

Le coursier recevra une notification pour les livraisons qui lui sont attribuées. Son travail sera de passer dans les magasins où une commande a été passée et de récupérer les articles commandés par le client.

Le marchand de son côté reçoit une notification également afin qu'il puisse préparer les commandes à l'avance. Il est obligatoire au marchand de disposer de l'article au moment de la commande et de mettre de côté pour le client.

Le client pourra également suivre l'état de sa commande via son accès à l'application.

Expression des besoins

Cas d'utilisation

Client :

UC1	Inscription de client
Acteur	Client
Flu basique	1. Un nouveau client clique sur le bouton 'S'inscrire'. Le système affiche l'interface d'inscription. 2. Le client complète les informations nécessaires. 3. Le client valide son inscription. Le système vérifie les informations. 4. Le système informe le succès d'inscription.
Flux alternatif 1	Mauvais infos -> Le système affiche les erreurs, demande au client de réécrire ces infos.

UC2	Commander
Acteur	Client
Flux basique	1. Le client choisit un magasin; Le système affiche la vitrine de ce magasin. 2. Le client choisit les articles et le nombre; Le système les ajoute dans le panier. 3. Le client valide son panier et paye. Le système attribue la commande à un livreur.
Flux alternatif 1	Le client recherche un nom de magasin -> Le système affiche la liste des magasins possibles
Flux alternatif 2	Le client recherche un nom de produit avant de choisir un magasin-> Le système affiche la liste des magasins qui possèdent ce produit.
Flux alternatif 3	Le client recherche un nom de produit après voir choisi un magasin-> Le système affiche ce produit.

UC3	Suivi d'avancement
Acteur	Client
Flux basique	1. Le client clique sur 'Mes commandes'. Le système affiche la page d'état de livraison. 2. Le client reçoit une notification quand ses produits arrivent. 3. Le client retire ses produits du casier(s) attribué(s).
Flux alternatif 1	Le client ne retire pas ses produits 24h après la livraison -> Le système libère la place attribué, le client est responsable.
Flux alternatif 2	Le client déclare un problème de livraison -> Le système contacte le livreur. Le litige peut être résolu via l'application et un remboursement peut s'effectuer.

Coursier :

UC4	Inscription de livreur
Acteur	Livreur
Flux basique	Un nouveau livreur clique sur le bouton 'S'inscrire comme livreur'. Le système affiche l'interface d'inscription. Le livreur complète les informations nécessaires. Un document contractuel est généré, une vérification humaine se déroule et le livreur est inscrit officiellement. Le livreur est attribué à une résidence.
Flux alternatif 1	Mauvaises infos -> le système corrige les informations ou demande une saisie nouvelle du livreur.

UC5	Livraison
Acteur	Livreur
Flux basique	1. Le livreur valide ses choix. Le système calcule un meilleur chemin à partir des points de passage et des contraintes horaires. 2. Le livreur valide une livraison une fois qu'il a livré. Le système envoie une notification au client correspondant.

Marchand :

UC6	Inscription de marchand
Acteur	Marchand
Flux basique	1. Un nouveau marchand clique sur le bouton 'S'inscrire comme marchand'. Le système affiche l'interface d'inscription. 2. Le marchand complète les informations nécessaires. 3. Le marchand valide son inscription. Une validation humaine est nécessaire. 4. Le système informe le succès d'inscription.

UC7	Publication de produit / Mise à jour de produit
Acteur	Marchand
Flux basique	1. Le marchand clique sur 'Ajouter/mettre à jour produit'. Le système affiche l'interface de publication. 2. Le marchand complète les informations nécessaires du produit. 3. Le marchand valide. Le système mets à jour sa base de données. 4. Le système informe le succès de publication.

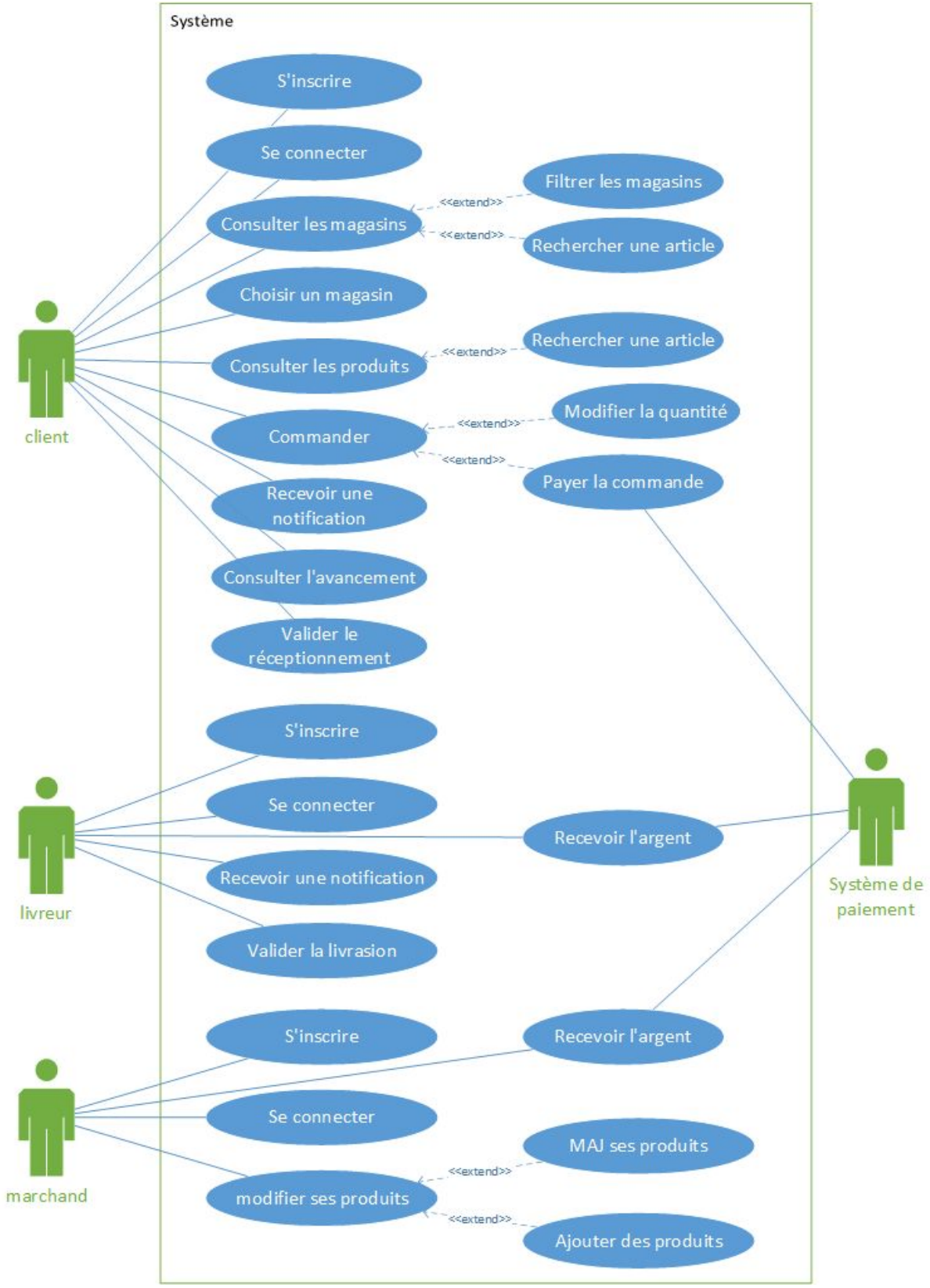


Diagramme des cas d'utilisation

Exigences fonctionnelles

Client :

1.S'inscrire

Le client pourra s'inscrire avec une adresse mail et un mot de passe, ces données seront enregistrées dans la base de donnée. Après l'inscription, le client doit fournir son adresse, et un moyen de paiement stable (Paypal, carte bancaire, etc…).

2.Se connecter

Le client devra se connecter avec son adresse mail et mot de passe. Si il oublie ses identifiants, il pourra créer un nouveau mot de passe avec son adresse e-mail.

3.Passer une commande

Pour trouver l'article souhaité, le client peut faire une recherche à l'aide d'un filtre, il existe plusieur types de filtre: distance, prix, mot clé, score du produit. Le client doit d'abord ajouter un article dans son panier avant de pouvoir valider son achat. Pour effectuer le paiement, le client utilise le moyen de paiement sauvegardé ou utiliser un autre moyen de paiement. Cette somme sera gardée en flottement par le système. Une fois que le livreur termine la livraison, la somme promise par le client est capturée puis versée au marchand. Pour la commande de la journée, le client doit passer sa commande avant 11h/ 17h pour se faire livrer après 12h/18h jusqu'à 14h/20h, sinon le client peut également passer une commande pour maximum une semaine à l'avance. Un ou plusieurs casiers seront attribués au client par le système. Il pourra accéder aux casiers avec une preuve de son identité après le dépôt des produits par le livreur.

4.Surveiller l'état des commandes

Pour connaître l'état d'avancement, le client peut consulter les informations suivantes: Le temps d'arrivée estimé et la localisation du livreur sur une carte interactive. Le client peut aussi contacter le livreur depuis l'application via un système de messagerie instantané. Une fois la livraison terminée, le client va recevoir une notification indiquant que sa commande a été livrée,il peut donc aller récupérer dans le(s) casier(s) attribué(s).

5.Retirer son article

Le client peut retirer sa commande dans le casier a l'aide d'une puce NFC/RFID. En cas de problème sur l'article, le client peut le signaler sur l'application et comme nous partirons sur le principe comme amazon: le client a toujours raison, mais lorsque le client a signalé une anomalie et que c'est prouvé faux, le client va perdre son crédit et ne peut plus signaler un problème.

Coursier :

1.S'inscrire

Le livreur pourra s'inscrire avec une adresse mail et un mot de passe, ces données sera enregistré dans la base de donnée .Après l'inscription, le livreur doit fournir numéro portable, plus sa carte bancaire pour recevoir son salaire.

2.Se connecter

Le livreur devait se connecter avec son adresse mail et mot de passe.

3.Faire des achats

Le livreur va recevoir une commande, indiquant l'adresse du client, l'adresse du magasin, le nom et la quantité de produit, le temps de livraison souhaité. La commande est distribué par le système et le coursier ne peut pas refuser. Le coursier n'a pas besoin de payer au magasin car c'est géré par l'application. Une fois l'achat terminé, le livreur doit déposer l'article dans le casier a l'aide de NFC, puis finaliser cet ordre sur son application, et le client va recevoir une notification.

4.Paiement

Le livreur recevra son salaire mensuel en fonction de nombre de livraison effectué.

Marchand :

1.S'inscrire

Le marchand pourra s'inscrire avec une adresse mail et un mot de passe, ces données sera enregistré dans la base de donnée. Après l'inscription, le marchand doit fournir numéro portable et l'adresse du magasin, ainsi que sa carte bancaire.

2.Se connecter

Le marchand devait se connecter avec son adresse mail et mot de passe.

3.Publier des articles

Pour publier un produit, il faut préciser le prix, une photo, une description, et des données supplémentaire.

4.Vendre un produit

Lorsque le client a payé son ordre d'achat, le magasin va recevoir une notification, qui lui donne du temps à préparer le produit avant que le livreur arrive. L'argent sera versé sur sa carte quand le client aura clôturer la commande.

Autre fonctionnalités

Un marchand a le droit de refuser une commande s'il ne dispose pas des produits demandés. Il pourra indiquer sur sa vitrine si un produit est en rupture de stock.

Si une commande ne dispose pas de tous les produits, la commande est livrée avec les produits disponibles mais la transaction ne sera pas complètement capturée : on ne capture que le montant correspondant aux produits livrés.

Une limite sera imposée sur le montant maximum, la taille maximum et le poids maximum d'une commande. Cette limite pourra être différente selon le livreur qui pourra prendre en charge la commande.

Concernant la répartition des commandes aux livreurs, notre stratégie est la suivante :

Un livreur est associé avec une ou plusieurs résidences. Il ne pourra livrer que des clients de ces résidences. Il peut prendre en charge une commande complète ou une partie de commande ou plusieurs parties de différentes commandes.

Une étude d'optimisation doit être effectuée avant l'attribution à tous les livreurs afin de déterminer quelle stratégie est à adopter.

Nous ne prévoyons pas une route à emprunter au livreur : vu qu'il livre que dans son quartier, il peut choisir l'itinéraire qu'il préférera, selon ses besoins et ses habitudes. Il sera par contre responsable de livrer les commandes à l'heure. Si il ne respecte pas les engagements, sa réputation baisse et il peut être amené à toucher une moindre marge.

Pour le système de réputation, nous prévoyons un système de notation de la part des utilisateurs associé avec un système de notation interne, qui prend en compte les retards, les litiges rencontrés, etc... La réputation finale sera constitué de ces deux facteurs.

Litiges

Nous allons discuter des cas de défaillances et litiges humains. Dans le cas d'un livreur, s'il est malade et qu'il déclare ne pas pouvoir livrer ce jour là, soit un livreur remplaçant pourra s'en occuper, soit les livraisons seront simplement suspendu ce jour là. Toutefois, si le livreur n'est pas honnête (disparition de marchandise), puisqu'il s'agit de livraison en résidence, souvent où les livreurs peuvent eux-mêmes habiter, ils se feront tout d'abord des voisins ennemis (socialement, c'est bien risqué pour eux), ils pourront être très rapidement radié de notre application, et avoir des problèmes juridiques du fait des engagements (signé lorsque accepté en tant que livreur) pris envers les clients. Si la défaillance humaine a lieu du côté des vendeurs, elle peut aussi avoir deux impacts principaux. Le premier est que le marchand ne donne pas vraiment ce à quoi il s'est engagé (comptant sur le fait que le livreur ne puisse pas tout vérifier), toutefois, cela serait plus lors de grosses commandes à des gros magasins, on pourrait ainsi demander des sacs scellés aux magasin, le livreur aurait juste à vérifier qu'il n'est pas été ouvert, ensuite il dépose les sacs tel qu'elle dans les casier, et s'il y a un problème sur un sac bien fermé, alors on sait que le problème vient du magasin. Il y a alors une démarche juridique classique, parfois bien longue, et nous stockerons le nombre de commande effectué à chaque magasin et le nombre de commandes ayant eu ce genre de gros problèmes, afin de les afficher aux clients. L'autre raison de la défaillance humaine avec un client, est que le vendeur déclare avoir le produit, accepte l'argent, mais n'a pas mis de côté le produit et n'en a plus lorsque le livreur arrive. Il y a alors un remboursement au client, et une baisse de sa réputation sur la page du magasin.

Paiements

Les paiements sont aujourd'hui gérés par un service tiers. Le service se charge de vérifier les fonds du client lors de l'autorisation de prélèvement et garantie la présence des fonds pendant une certaine durée. Ce montant est en flottement durant la durée de vie de la commande et le système peut choisir de pouvoir soit la capturer, soit la retourner à son propriétaire.. Ainsi, les marchands ne peuvent toucher la somme due que lorsqu'ils confirment avoir les produits pour la livraison (et sont alors supposé les mettre de côté). Un remboursement intégral ou partiel est

prévu pour la résolution des litiges.

Exigences non fonctionnelles

Sur l'ergonomie et le design de l'IHM, on privilégiera une interface fluide et claire. La responsivité des sites web est à mettre en priorité. Si on souhaite migrer le système pour l'adapter pour des plateformes mobiles, il est nécessaire de produire des interfaces qui respectent les bonnes pratiques.

Pour la fiabilité, différents points sont à aborder. Tout d'abord, la sécurité des données est essentielle. Pour cela, nous aurons un système stable bien formé, des transferts d'informations sécurisés, mais cela sera se basera ainsi surtout sur les différentes exigences techniques de la gestion des données. Au niveau des sites web, nous utiliserons HTTPS pour assurer des données cryptées. Par contre, il reste l'aspect de la confidentialité des données. Il est ainsi décidé que les marchands ne pourront pas avoir accès aux informations personnelles des clients, et qu'ils ne rentreront ainsi en contact qu'avec le coursier. De même, les clients n'auront pas accès aux informations des autres clients. Finalement, pour le coursier, il aura bien entendu accès aux commandes des clients, ainsi que ce qu'il devra déposer dans chaque panier, toutefois, comme les casiers n'appartiennent pas à un client et ne sont que attribués que pour 24h, le livreur ne peut pas exploiter de façon malveillante les informations qu'il dispose.

Au niveau de la scalabilité, préférer une solution technique où la logique permet de mettre en place du stockage d'informations à volume important.

Extensibilité/évolutivité :: choisir une voie de développement avec possibilité de modulariser les composants, et d'ajouter des services après la phase primaire de développement.

Performance : le système doit satisfaire les besoins basiques d'une application web. La disponibilité doit être quasi totale, et l'infrastructure doit pouvoir supporter des charges de trafic important.

Exigences techniques

Le système doit utiliser une architecture en 3 couches : la couche vue, la couche traitement et la couche données.

La couche vue doit permettre l'accès depuis des navigateurs web, donc utiliser les technologies HTML et Javascript. La couche traitement doit assurer les besoins en terme de concurrence d'accès : tous les utilisateurs doivent pouvoir accéder à la plateforme sans famine.

Il doit être possible de migrer les couches sur différentes plateformes, différentes infrastructures et garantir le bon fonctionnement du système avec peu de configuration.

Réalisation POC

Pour illustrer le projet fais-mes-courses, nous proposons la réalisation d'un Proof-of-Concept avec une simplification des fonctionnalités décrites plus haut, en raison du facteur temps.

Simplifications

Une commande ne peut que provenir d'un seul magasin. Nous n'effectuons pas de vérifications au niveau du poids ou du volume. Une limite sur le prix est cependant présente.

Nous n'utilisons pas de système de répartition des commandes pour les livraisons : une commande est attribuée à un livreur selon une répartition homogène entre tous les livreurs candidats.

Le paiement n'accepte pas le remboursement après capture du montant.

L'ouverture des casiers se fait actuellement avec une paire de clés générées, à usage unique. Nous simulons les casiers avec une interface web.

Le système de réputation n'est pas implémenté.

Une interface a été prévue pour faire évoluer le système d'authentification vers un protocole OAuth2 avec génération de bearer tokens.

La plateforme de résolution des litiges n'est également pas implémentée.

Architecture technique

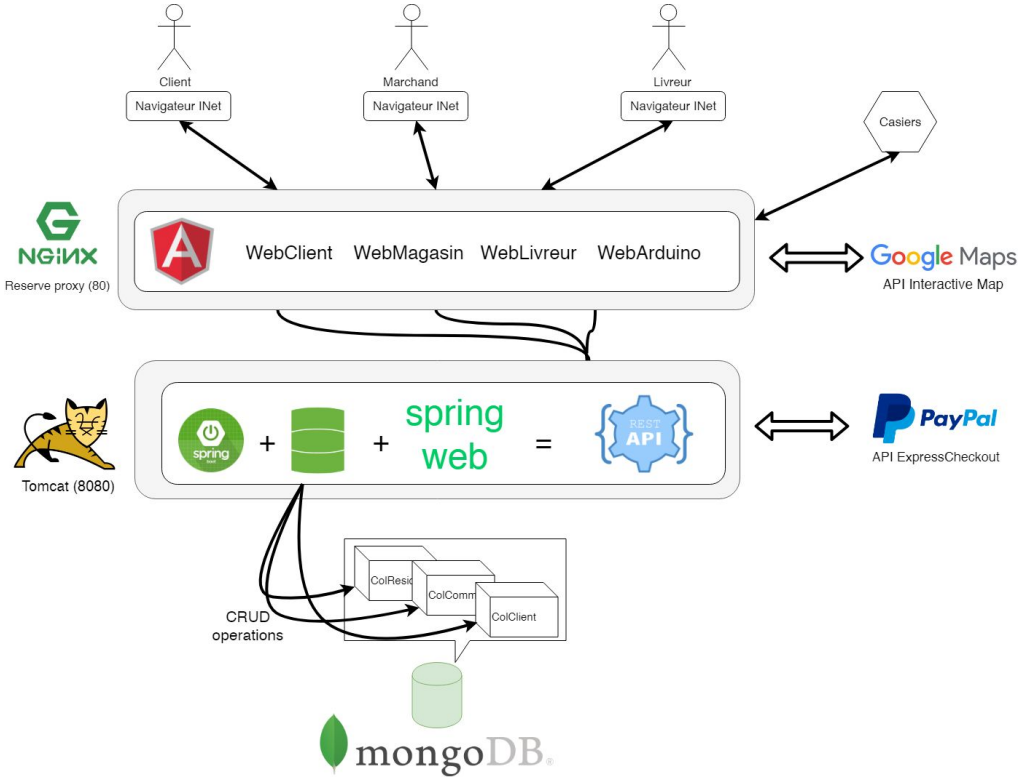
Vue Globale

L'architecture est répartie en 3 composants principaux : nous utilisons AngularJS pour la plateforme de production des pages HTML, un service Java basé sur Spring pour l'accès et le traitement des données et mongoDB pour le stockage.

Nous avons choisi AngularJS pour la modularité et l'évolutivité du système : le framework permet de générer des composants rapidement avec peu de code.

Nous utilisons mongoDB pour prévoir une exploitation rapide de données. Les opérations sont principalement des opérations CRUD et mongoDB réponds parfaitement à nos besoins.

Nous utilisons Spring comme framework back-end afin de faciliter le développement des services d'accès aux données, et des endpoints au service du front-end.



Le schéma illustre de façon globale le découpage de l'architecture du POC.

Front-end

Le front-end est découpé en 4 applications séparées et distribuables avec un cloisonnement sur les accès. Chaque application dessert un utilisateur particulier et n'interagit pas avec les autres.

Un serveur NGINX sert de dispatcher : nous accédons aux différentes applications via des sous-domaines différents et cela de manière sécurisée via HTTPS.

Nous interagissons également avec l'API Google Maps pour générer une carte interactive pour le livreur.

Back-end

L'application back-end est une application Spring Boot avec des dépendances sur Spring Data (pour les accès à mongoDB) et Spring Web (création de contrôleurs web). Le design du back-end respecte les normes REST : les requêtes ne dépendent pas du contexte et nous prenons en compte différents types de requêtes HTTP (GET, POST, DELETE) pour un même URI. La description des routes pour l'accès du front-end est décrite dans le tableau ci-dessous.

Description des routes REST

La coloration des paramètres spécifie le type JSON utilisé :

Object Array Number String Date Array<Object>

Routes pour accès front-end

Route	Paramètres	Réponse	Description
Côté client:			
POST /api/register	user, password	token, user	création compte client
POST /api/authenticate	email, password	token, user	connection compte client
POST /api/updatePanier	commande	“ok” / ”ko”	ajouter un article
GET /api/getMagasinsOfResidence	residencedid	magasin	récupérer les magasins autour de chez lui
GET /api/getPanier	userid	commande	récupérer son panier
GET /api/getCommandesEnCours	userid	commande	récupérer ses commandes en cours
GET /api/getLastCommandes	userid	commande	récupérer les 3 dernières commandes
GET /api/getCommandeArchiver	userid	commande	récupérer tous les commandes passées
GET /api/findResidenceFromCodePostal	codePostal	residence	récupérer les résidences à partir du code postal
POST /api/authenticateToken	email, token	user	
Côte livreur:			
GET /api/livreur/getCommandesEnCours	userid	commande	récupérer les commandes en cours
GET /api/livreur/getCommandesArchiver	userid	commande	récupérer les commandes terminées
POST /api/livreur/authenticate	email, password	token, livreur	connection compte livreur
Cote magasin:			
POST /api/authenticateMarchand	email, password	token, marchand	connection compte marchand
POST /api/registerMarchand	marchand, password	token, marchand	création compte marchand
POST /api/updateproduit	produit, idMagasin	“ok” / ”ko”	création / MAJ d’un produit
GET /api/getProduits	marchandid	produit	récupérer les produits

			d'un magasin
GET /api/getProduit	marchandid, produitid	produit	récupérer un produit d'un magasin
POST /api/deleteProduit	marchandid, produitid	ok" / "ko"	supprimer un produit d'un magasin
POST /api/validation	marchandid, commandeid	boolean	valider une commande
GET /api/getMarchand	idmarchand	marchand	récupérer un marchand

Normes de développement

Le groupe sera divisée en deux équipes : une petite équipe pour le front-end, et une équipe pour la réalisation du back-end et des opérations de maintenance et de déploiement.

Le code est hébergé sur Github. L'intégration continue des projets est hébergée sur travis-ci.org.

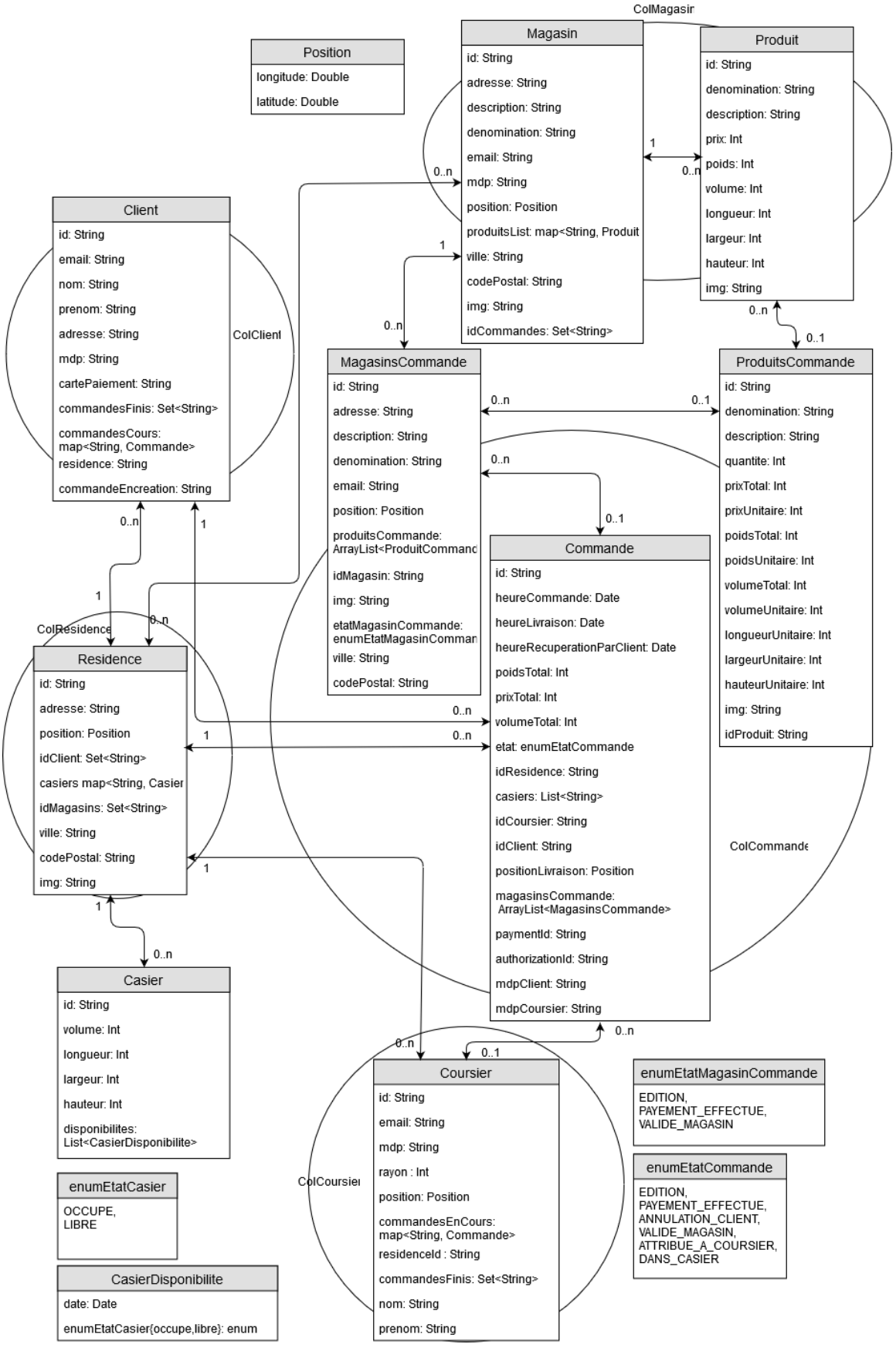
Les modifications sur la branche master sont interdites pour protéger le code source. Les développeurs doivent travailler sur des branches naissant de master, puis de demander une pull request qui sera approuvée par un autre collaborateur.

Le projet est réalisé en 3 sprints : le premier sprint est dédié à l'installation des outils, à la conception du sujet tant fonctionnel que technique et à la construction des éléments de base pour commencer le développement

Le sprint 2 est le sprint où la plupart des fonctionnalités de base sont implémentée. Les différents projets doivent passer les tests unitaires.

Le sprint 3 permet l'intégration des différents projets sur une même plateforme. Plusieurs tests d'intégration sont effectués en local et sur la plateforme de production pour vérifier la cohérence du scénario choisi. Si il reste du temps, des fonctionnalités supplémentaires seront développées.

Modèle E-A des données mongoDB



Difficultés rencontrées

Lors du développement du projet, la plus grosse difficulté rencontrée est le manque du temps : la première semaine, il a été difficile de travailler en synchronisation à cause d'obligations personnelles des différents membres. En plus, avec un temps de développement de 2 semaines, nous avons sous-estimé la complexité du projet et donc beaucoup de fonctionnalités proposées n'ont pas pu être implémentées dans le POC.

Le travail de conception en amont a été globalement bien réalisé, avec certaines erreurs à cause de la prise en main des outils : certaines fonctions de Spring lors de la sérialisation des données mongoDB ont été mal comprises et à l'exécution cela a provoqué des bugs qu'il a fallu résoudre.

Au final, l'intégration entre front et back nous a pris plus de temps que ce que nous avions estimé à la base. Le scénario proposé est donc finalement plus simple que le sujet en lui-même.

La gestion du projet est globalement bonne. Nous avons utilisé Trello pour le suivi des tâches. Si cela ne nous donne pas d'indications précises, c'est suffisant pour répertorier les tâches et faire des prévisions.

Glossaire

Client/Acheteur : le client représente la personne physique habitant dans un complexe immobilier éligible à l'utilisation de notre application. Un client sera valide si il dispose d'un moyen de paiement vérifié (à son nom) et actif. Un client dispose nécessairement d'une adresse (non-fixe) et d'une identité (fixe) basée sur son nom et ses identifiants. Les identifiants d'un client seront : une adresse mail vérifiée et un mot de passe.

Coursier/Livreur : le coursier est la personne physique associée à notre entreprise et avec comme mission de récupérer les commandes des clients et de délivrer les livraisons à leur adresse. Nous considérons ces coursiers comme disposant à tout moment d'une connexion internet et pouvant travailler aux heures convenues. Leurs définitions sont immuables pour simplifier le cas.

Marchand/Vendeur : le marchand représente une personne morale disposant d'un commerce à accès public. Il dispose d'une vitrine sur notre application. Nous imposons la contrainte que pour tout article passée en commande, le marchand doit réserver l'article et garantir sa disponibilité à l'arrivée du coursier. Le marchand doit également disposer d'une adresse et d'un moyen de versement (compte bancaire).

Vitrine : la vitrine est un espace sur notre application dédié et complété par le marchand que le client pourra consulter afin de voir les articles en vente chez le marchand. La vitrine est actualisée sur demande du marchand. Le client pourra consulter la vitrine de tous les magasins en ligne mais il ne pourra commander que sur des vitrines dont il est éligible.

Article : une article est un produit destinée à la vente périssable ou non périssable disponible chez le marchand. Un article dispose nécessairement d'un nom et d'un prix à l'unité. Un article doit appartenir à une vitrine mais il peut appartenir à plusieurs vitrines en même temps avec des prix différents. Pour simplifier le cas d'étude, un marchand pourra publier un article sur sa vitrine uniquement si il est en magasin. Un article périssable (alimentaire) pourra avoir une quantité limitée par jour, mais un article non-périssable sera considéré comme en quantité infinie. **Un article n'est pas à être confondu avec un produit, qui est le bien physique que représente l'article.**

Casier : un casier est un objet physique pour stocker une commande livrée. Un casier est attribué à un client lorsque sa commande sera livrée. Si une commande est de taille trop importante pour un casier, l'application pourra choisir d'attribuer plusieurs casiers à un même client. Le client dispose de 24h après la livraison de sa commande pour récupérer les produits. Si ce délai est dépassé, les casiers peuvent être attribués pour un autre client et c'est à la charge du client en retard de régler tout incident.

Commande : une commande représente l'action de valider un panier et de régler une transaction. Après cet instant, une commande représente la liste des articles achetés par le client. Une commande peut couvrir 1 ou plusieurs marchands. Une commande passée après avant les heures de travail des coursiers sera traitée le lendemain. Une commande sera validée seulement après validation de la transaction.

Livraison : une livraison représente l'action nécessaire réalisée par le coursier pour récupérer

les produits achetés par le client chez les marchands puis de les délivrer dans un ou plusieurs casiers du complexe d'habitation du client. Une fois les produits délivrés, le coursier valide la livraison.

Transaction : une transaction se résume à un paiement en ligne par le client pour acheter les articles de son panier. La somme est débitée sur un compte tiers mais cette somme n'est pas versée immédiatement au marchand. La somme sera créditée au marchand après validation de la livraison. Une transaction est effectuée quand le client est débitée puis clôturée quand le marchand est crédité.

Notification : une notification est un message délivré à un destinataire soit après la validation d'une commande, soit après validation d'une livraison. Pour le moment, une notification prendra la forme d'un message texte envoyée à l'adresse mail du destinataire.