# Cahier des charges

## 1. Contexte du projet

1. Besoins du client

L'application Livrai était initialement une application servant à centraliser des livraisons de marchandises en grosses quantités à destination des professionnels. L'objectif a cependant changé et il s'agit désormais de réaliser une application de plus grande échelle ayant pour but la centralisation de livraisons de tout type de colis dans l'ensemble de la France. L'application devra être en lien avec les tendances actuelles en matière de performance et de sécurité. Les technologies choisies se voient dont être **Java / Spring** pour le back-end, **Angular** pour le front-end, **MySQL** pour le stockage des données de façon relationnelle et **Docker** pour la conteneurisation.

2. Objectifs du projet

L'objectif est ici de moderniser et d'améliorer le code de l'application de sorte à la rendre plus robuste face à sa nouvelle clientèle. L'utilisation d'Angular a également pour objectif de permettre, au besoin, l'utilisation de librairies de composants telle qu'Angular Material de sorte à offrir une expérience visuelle plus en lien avec les standards actuel en termes d'esthétique. L'ajout de responsiveness aidera également à l'utilisation de l'application sur les smartphones, l'un des moyens de navigation moderne qu'il n'est plus possible d'ignorer en 2025.

3. Mesure de la réussite du projet

La réussite du projet pourra, dans un premier temps, être mesurée par la complétion et la couverture du code des différents applicatifs par des batteries de tests. Ces derniers pourront être réalisés avec les librairies suivantes :

- Jest pour la partie front-end
- JUnit / Mockito pour la partie back-end
- Cypress pour la réalisation de tests E2E
- TestContainers de sorte à disposer lors des tests d'intégration de bases de données non liées à la production de sorte à ne pas perturber le fonctionnement de base de l'application finale

L'ensemble des tests devront avoir une couverture d'au moins 80% pour être en lien avec les standards de l'industrie.

L'ensemble du code source sera déposé dans un système de versioning Git. La plateforme en ligne GitHub sera privilégiée de par sa capacité à gérer les vulnérabilités de dépendances (**Dependabot**), le soulèvement de soucis techniques (**GitHub Issues**) et la collaboration à plusieurs sur des dépôts multiples (**GitHub Projects**). L'utilisation de pull requests sera obligatoire de part l'ajout de sécurisation de la branche principale (**main / master**) ainsi que des branches de **release**. Dans le but de respecter le **Git Flow**, le projet devra utiliser également un ensemble de branches préfixées de **hotfix/*** et de **feature/*** de sorte à permettre à l'équipe de s'y retrouver et de ne pas empiéter sur le travail des uns et des autres.

Les tests seront obligatoires lors de la réalisation de pull requests, qui seront d'ailleurs déclenchés au moyen d'une pipeline réalisée via **GitHub Actions**. Cette pipeline aura pour but de permettre, dans un pull request vers **master / main :**

- Build de l'application
- Test de l'application (unitaires)
- Déploiement dans un environnement de reviewing
- Réalisation de tests E2E dans cet environnement
- Packaging de l'application en images Docker
- Upload des images Docker sur un registre d'image de conteneur

Une fois le pipeline réalisé et le pull request vérifié par un ou plusieurs reviewer(s), il sera ensuite possible de procéder à l'exécution d'un autre pipeline, celui-ci ayant pour but de déploier l'applicatif dans notre environnement de production. Ce pipeline devra se connecter en SSH à l'environnement et réaliser l'ensemble des commandes Docker de sorte à pouvoir re-déployer l'applicatif en cas de mise à jour.

4. Contraintes réglementaires

*Dans le but de respecter la RGPD, l'ensemble des données utilisateurs telles que le compte et les mots de passe pourront être supprimés à leur demande. L'ensemble des livraisons se verront anonymiser dans le but de respecter la vie privée des clients. Cependant, dans le but de ne pas avoir d'éventuels problèmes comptables, les factures seront conservées dans le serveur pour toute demande éventuelle.*

*Les données seront stockées sur le territoire français, dans une infrastructure située dans les locaux de l'entreprise Livrai, celle-ci ayant déjà son siège social en France.*

5. Éléments hors périmètre

Des améliorations futures pourront avoir lieu, par exemple :

- Ajout d'une application mobile pour permettre aux utilisateurs de réaliser des commandes depuis leur téléphone mobile, ce partout en France du moment que le réseau le permettrait.
- Modification de l'architecture de l'application de sorte à passer par une architecture en micro-services (séparation des fonctionnalités en services qui seront ensuite conteneurisés et déployer dans un environnement de type Kubernetes)
- Utilisation d'un cloud provider dans le but de déployer en nuage l'applicatif (par exemple, utilisation d'Azure, région française de sorte à respecter le RGPD)
- Utilisation d'une architecture cloud privée, managée via Openstack et Terraform / Ansible dans le but de pouvoir facilement passer d'une version à l'autre de notre architecture interne et de garder le contrôle des données, celles-ci se voyant alors être à coup sur sur le territoire français.

# 1. Description fonctionnelle

1. Liste des fonctionnalités

L'application aura pour but de permettre à plusieurs catégorie d'utilisateurs de se connecter et de manipuler leur profil :

- Utilisateurs clients
- Utilisateurs en lien avec le service de livraison
- Utilisateurs en lien avec le service commercial

Ainsi, les fonctionnalités de l'application devront demander une sécurité pourvue d'un RBAC (Role Based Access Control) de sorte à éviter les modifications de données ou d'état de l'applicatif par un individu ne devant pas le pouvoir.

Les fonctionnalités de l'application, pour tout type d'utilisateur, seront :

- **Authentification** : Au moyen d'un formulaire d'authentification, les visiteurs de l'application pourront réaliser une connexion ou un enregistrement. Le formulaire devra comporter les champs suivants : email et mot de passe pour le formulaire de connexion. Le formulaire d'enregistrement, quant à lui, comportera également une confirmation de mot de passe ainsi que des informations utilisateur telles que le nom, prénom et nom d'utilisateur. Le mot de passe devra comporter une lettre majuscule, une minuscule, un chiffre, un caractère spécial et être d'une longueur minimale de 8 caractères.
- **Gérer son profil** : Tout utilisateur aura la possibilité, dans une page dédiée, de vérifier ses informations personnelles. Il sera également possible, en cas d'appui sur un bouton, de procéder à l'édition du mot de passe ou des informations autres que le nom d'utilisateur (ce dernier étant modifiable uniquement par les membres du service commercial).

Dans le cas où l'utilisateur est un client, alors il pourra utiliser les fonctionnalités suivantes :

- **Afficher les commandes** : Tout utilisateur pourra voir les commandes qu'il a généré et en suivre l'évolution. Une commande comportera une quantité, un produit, une somme totale à régler, le lien vers une facture (en cas de validation par le service concerné) et un statut.
- **Suivre une commande** : Le suivi de la commande sera également disponible via son statut. Le statut de la commande sera géré au moyen d'une énumération comportant l'ensemble des statuts relatifs au trajet ou à son paiement.
- **Créer une nouvelle commande** : Au moyen d'un formulaire dédié, le client aura la possibilité de générer une nouvelle commande. Celle-ci se verra alors pourvue d'une date de création, de sorte à ce que le service livraison ou commercial puisse la retrouver sans trop de mal via un tri par date décroissant ou un filtrage entre deux dates.

Un utilisateur du service de livraison pourra, de son côté, également :

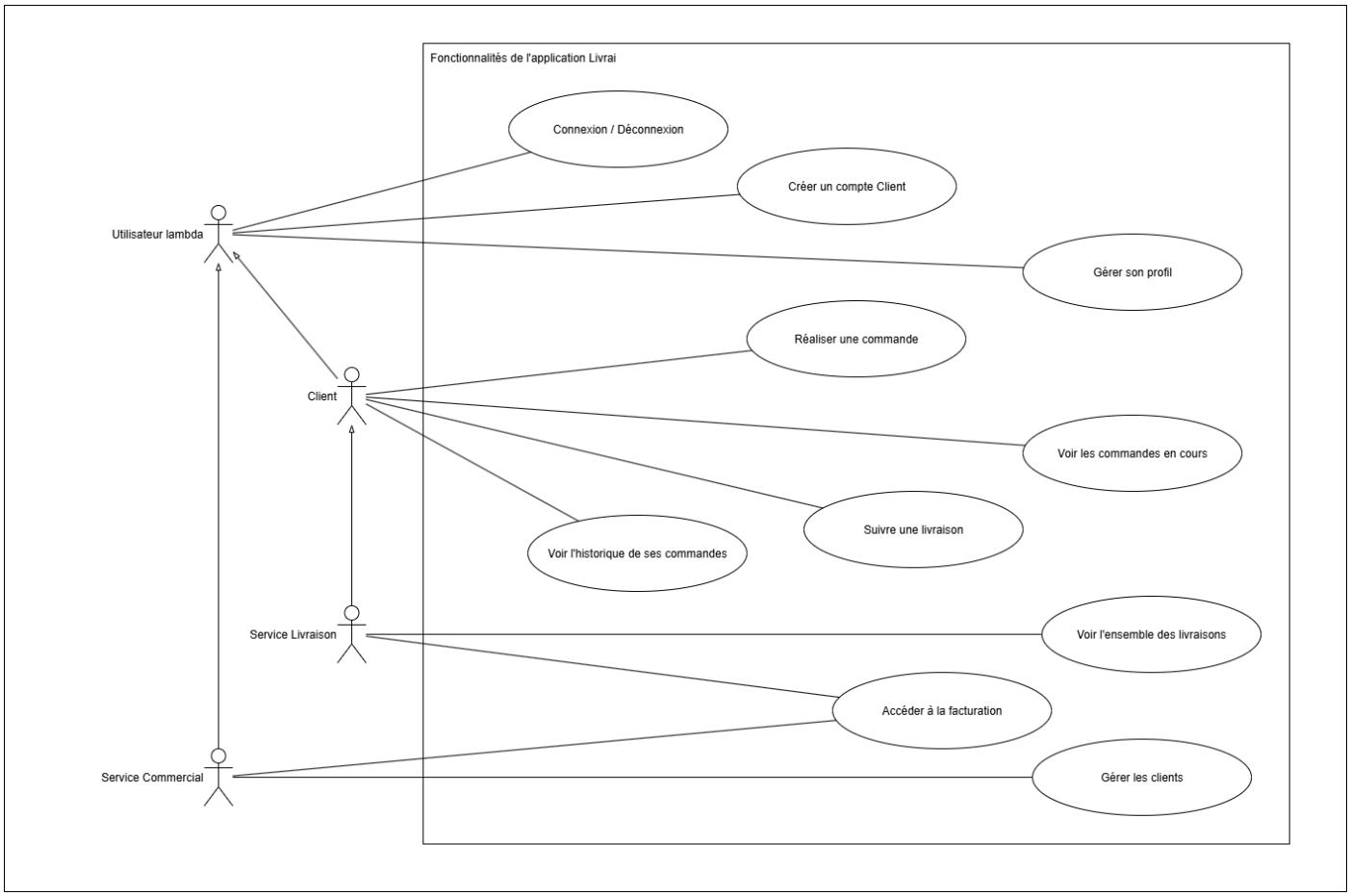
- **Voir l'ensemble des livraison / commandes** : Dans une page dédiée, le service de livraison pourra observer l'ensemble des livraisons. Via un système de filtres multiples, il sera également plus aisé pour ce service de retrouver une livraison spécifique. En cas de connaissance du numéro de commande, il sera d'ailleurs possible de directement filtrer la commande voulue.
- **Accéder à la facturation** : Dans le cas où la commande voulue est trouvée, alors il sera possible de cliquer sur un bouton dans le but d'observer sa facturation. Celle-ci comportera l'ensemble des champs relatifs à l'objet de la commande, à la personne l'ayant réalisée et au service bancaire concerné par la transaction.

Enfin, un utilisateur du service commercial, pourra de son côté :

- **Gérer l'ensemble des utilisateurs** : Il sera possible pour les membres du service financier de retrouver des informations clients. Il sera d'ailleurs possible, en cas de demande d'un client, de réinitialiser son mot de passe ou de modifier son nom d'utilisateur. Le verrouillage d'un compte utilisateur suite à une mauvaise conduite ou à un nombre de tentatives infructueuses de connexion trop important pourra également être levé par un membre du service client.
- **Consulter la facturation** : Le service client aura accès à l'ensemble des factures concernant Livrai, de sorte à pouvoir au besoin en modifier le statut et en retrouver les informations, qu'il pourra alors, en cas de demande éventuelle d'un client, transmettre.

2. Fonctionnement de l'application

Le diagramme de cas d'utilisation suivant décrit globalement l'ensemble des fonctionnalités de l'application et permet d'observer les relations entre les différents acteurs et leurs droits d'accès :




3. Impact Social

Dans le but de permettre une utilisation adaptée à chacun, l'application devra disposer de l'ensemble des fonctionnalités relatives à la navigation sans souris, par les logiciels permettant aux mal-voyants de naviguer sur le réseau internet, mais également être adapté aux individus sensibles aux mouvements et aux animations. Pour cela, le remplissage des balises **aria-*** sera rendu obligatoire pour l'équipe en charge de la partie front-end de l'applicatif, de même que l'utilisation des média-queries et des paramètres du navigateur pour palier à une navigation autre que par curseur de souris.

Les contrastes seront adaptés aux individus possédant un trouble de la vision des couleurs, ce via l'ajout de plusieurs thèmes via Angular Material.

Dans le but de permettre aux individus handicapés moteurs de bénéficier d'une livraison adaptée à leurs besoins, il sera aussi possible, lors de la création d'une commande, de pouvoir ajouter une série de commentaire et de cocher une case informant la future équipe de livraison que celle-ci demandera un soin particulier suite à une situation de handicap.

# 2. Description technique

### 1. Choix de technologie pour le front-end

Le **front-end** de l'application sera développé en **Angular**, un framework robuste, modulaire et maintenu par Google. Il s'agit d'une solution SPA (Single Page Application) qui permet d'assurer une expérience utilisateur fluide et dynamique.
Angular sera couplé à **Angular Material**, une bibliothèque de composants UI standardisés, modernes et accessibles, permettant :

- un design responsive adapté aux mobiles et tablettes ;
- une accessibilité améliorée (balises ARIA, navigation clavier, thèmes à contraste élevé) ;
- une uniformisation de l'interface avec des composants prêts à l'emploi, réduisant ainsi la charge de développement.

Cette stack front-end permettra également l'intégration de tests unitaires avec **Jest**, et de tests end-to-end avec **Cypress**, afin de garantir la fiabilité de l'interface.

### 2. Choix de technologie pour le back-end

Le **back-end** sera développé en **Java**, langage déjà maîtrisé par l'équipe informatique de Livrai. Le framework choisi est **Spring Boot**, dans sa dernière version LTS, pour les raisons suivantes :

- Rapidité de développement grâce à une configuration minimale ;
- Intégration facile avec Spring Security pour le contrôle des accès via RBAC (Role Based Access Control) ;
- Prise en charge native de REST APIs, idéal pour les échanges entre le front-end Angular et le back-end ;
- Intégration fluide avec JPA/Hibernate pour la gestion de la base de données relationnelle.

Les tests seront réalisés avec **JUnit** et **Mockito**, et des **TestContainers** seront utilisés pour les tests d'intégration, notamment avec la base MySQL.

### 3. Choix de technologie pour la base de données

La base de données utilisée sera **MySQL**, un système de gestion de base de données relationnelle largement éprouvé, open-source, et compatible avec les ORM comme Hibernate. MySQL permettra :

- La gestion robuste des données client, des commandes et de la facturation ;
- Un bon support transactionnel ;
- Une compatibilité native avec Spring Data JPA.

Les bases de test seront isolées de la production via l'utilisation de **containers Docker** (TestContainers) pour assurer des tests fiables sans impacter les données en production.

### 4. Choix de technologie pour le déploiement

Le déploiement se fera au moyen de **Docker**, pour une conteneurisation de tous les services applicatifs. Chaque partie (front-end, back-end, base de données) sera empaquetée dans des images Docker distinctes, ce qui permettra :

- Une portabilité de l'application quel que soit l'hébergeur final (cloud public, privé ou infrastructure interne) ;
- Une gestion simplifiée des mises à jour via pipelines CI/CD ;
- Une mise en production via connexion SSH automatisée et relance des containers.

Le registre Docker servira également à stocker les images applicatives construites lors des pipelines CI exécutées avec **GitHub Actions**. Le pipeline automatisera le build, les tests (unitaires + E2E), le packaging Docker et le déploiement dans des environnements de test et de production.

Pour appuyer les choix techniques préalablement exprimés, un diagramme de la nouvelle architecture a également été réalisé :