

UE Full Stack

TP1 UE Full Stack

1. Introduction	3
1.1. Objet Du Document	3
1.2. Objectifs Du Projet	3
1.3. Contenu Du Document	3
1.4. Convention	3
2. Exigences Fonctionnelles Et Opérationnelles	4
2.1. Définitions préalables	4
2.2. Généralités et missions du système	4
2.3. Gestion des livreurs	4
2.4. Gestion des tournées	6
2.5. Gestion des livraisons	7
2.6. Rôle des utilisateurs (Facultatif)	7
3. Exigences Techniques	8
3.1. Architecture	8
3.2. APIs	8
3.3. Technologies	9
4. Exigences Du Projet	9

1. Introduction

1.1. Objet Du Document

Le présent document décrit le contexte et le périmètre du TP1 proposé aux étudiants du Master « Génie de l'Informatique Logicielle » pour l'année universitaire 2023-2024 de l'UE Full Stack. Ce document sert de référence pour présenter l'ensemble des exigences fonctionnelles et de qualités associées au TP. Il définit également la liste des fournitures relatives au projet.

1.2. Objectifs Du Projet

Ce projet est un système qui permet la gestion de livreurs.

1.3. Contenu Du Document

La première partie du document est consacrée à une description générale du projet et à la définition d'un certain nombre de notions relatives aux fonctionnalités ou aux technologies à mettre en œuvre dans le projet.

La lecture de ces définitions constitue un prérequis indispensable pour la bonne compréhension de la suite du document.

Chacune des fonctions de ce TP fait l'objet d'une description dans laquelle sont exprimées les exigences à satisfaire. Ce document définit ensuite les exigences techniques applicables à la conception et au développement du projet. Enfin, le document définit les exigences de management qui devront être respectées pour la conduite du projet.

1.4. Convention

Le présent document a été rédigé en respectant les règles suivantes :

- Les exigences sont référencées selon le format suivant :
 - la lettre « E » pour « exigence » suivie d'un tiret bas « _ »,
 - trois lettres pour codifier la catégorie fonctionnelle de l'exigence, suivies d'un tiret bas « _ »,
 - un numéro d'incrément de 10 en 10.
- Les codes utilisés au deuxième point sont les suivants:
 - LVR pour les exigences liées à la gestion des livreurs
 - ARC pour les exigences liées à l'architecture du système
 - API pour les exigences de réalisations des API
 - TEC pour les exigences techniques de réalisation du système

- PRO pour les exigences du projet

2. Exigences Fonctionnelles Et Opérationnelles

2.1. Définitions préalables

Dans la suite du document, les définitions ci-après seront utilisées pour la formulation des exigences :

- Application Programming Interface (API) : Interface de programmation permettant d'accéder à des fonctions, des procédures ou des classes d'objets mises à disposition par un composant logiciel.
- CRUD (Create, Read, Update, Delete) : Désigne les quatre opérations de base pour la persistance des données, création, lecture, mise à jour, suppression.
- Continuous Integration / Continuous Delivery (CI/CD) : Système permettant l'intégration et la livraison continue de composants logiciels
- SSO (Single Sign On) : Système de gestion centralisé des identités, de l'authentification et des droits.
- Docker : Technologie de conteneurisation qui permet la création et l'utilisation de conteneurs Linux

2.2. Généralités et missions du système

Le système doit permettre la gestion de livreurs.

Les fonctionnalités devront être développées au travers d'une IHM dédiées et accessibles au travers d'une application web (SPA) reposant sur des APIs web.

2.3. Gestion des livreurs

Le système de gestion des livreurs permet d'effectuer des opérations comme la recherche et le tri selon différents critères, la consultation, la modification ou la suppression des livreurs.

Les exigences ci-après sont applicables au système de gestion des livreurs.

[E_LVR_10](#) Le système permet aux utilisateurs de créer un nouveau livreur. Lors de la création d'un livreur, les informations minimum à renseigner sont :

- Le nom
- Si le livreur est disponible

E_LVR_20 Le système permet aux utilisateurs de sélectionner un livreur existant et de modifier les champs renseignés lors de la création du livreur tels que définis par E_LVR_10.

E_LVR_30 Le système permet aux utilisateurs de supprimer un livreur.

E_LVR_40 Le système permet d'effectuer une recherche paginée sur tous les livreurs qui sont présents en affichant :

- Leur nom
- Leur date de création
- Si le livreur est disponible

E_LVR_50 Le système permet d'effectuer une recherche sur les livreurs avec un ou une combinaison de filtres qui sont :

- Si le livreur est disponible
- Date de création
 - Après une date précise
 - Avant une date précise
 - Entre deux dates précises

E_LVR_60 Le système permet d'effectuer une recherche sur les livreurs en triant par :

- Le nom
- La date de création

E_LVR_70 Le système permet d'afficher le détail d'un livreur en affichant les champs et les entités tels que définis par E_LVR_10 et E_LVR_40

3. Exigences Techniques

3.1. Architecture

Le système est basé sur une API REST.

E_ARC_10 Le système est composé d'un client et d'un serveur.

E_ARC_20 Le serveur dispose de son propre système de gestion de données.

E_ARC_30 Les échanges client-serveur doivent être uniformes. Par exemple, si les échanges entre le client et le serveur sont en JSON, alors tous les échanges doivent être en JSON.

3.2. APIs

Le projet doit pouvoir être intégré facilement. Pour se faire, les différentes fonctionnalités mises en œuvre doivent être accessibles au travers d'APIs web.

E_API_10 Le système met à disposition une API REST permettant d'accéder aux fonctionnalités de gestion des livreurs détaillées en 2.3.

E_API_20 Le système met à disposition des APIs sans état (RESTFull) afin de permettre leur redondance et leur mise à l'échelle.

3.3. Technologies

E_TEC_10 Le serveur doit être développé en s'appuyant sur un framework.

E_TEC_20 Le client doit être développé en s'appuyant sur un framework.

E_TEC_30 Toute requête sur le système doit se faire en moins de 5 secondes.

4. Exigences Du Projet

Pour cette UE une "tenue" de projet est obligatoire.

E_PRO_10 Des outils de gestion du code source, de configuration doivent être mis en place. Ils intègrent :

- Une gestion du code source Git
- L'utilisation de Gitflow
- Une configuration docker qui permet de lancer le serveur, le client et la base de données

E_PRO_20 Le serveur doit tourner sur le port 8080.

E_PRO_30 Le client doit tourner sur le port 4200.

E_PRO_40 Facultatif Une installation client-serveur sur un serveur distant est possible, dans ce cas le lien pour accéder à ce serveur doit être fourni dans la documentation du projet.

E_PRO_50 Facultatif Lors du développement du projet, du CI/CD peut être mis en place pour toutes les tâches répétitives. Par exemple, lors d'un hotfix avec gitflow une fois que le code est sur la branche main, il faut penser à mettre à jour la branche develop.

E_PRO_60 Chaque personne participant au projet doit rédiger un document descriptif de ses travaux personnels au sein du projet listant ses principales contributions et en insistant fortement sur les principales difficultés rencontrées, et/ou les modes de résolution associés.

E_PRO_70 Lors du développement, les commits git du projet doivent utiliser la convention angular pour les commits¹.

E_PRO_80 Lors du développement, le code doit être homogène, respecter les conventions de code et l'agencement des fichiers comme recommandé le framework que vous allez utiliser.

E_PRO_90 Le serveur doit avoir une gestion des exceptions et des erreurs permettant de prévenir les futurs clients des éventuels problèmes dans les appels API.

E_PRO_100 Le serveur doit éviter de solliciter la base de données avec plusieurs appels successifs si un seul appel peut suffire. Par exemple, faire 10 appels différents pour rechercher les 10 premiers produits d'une boutique, dans ce cas, un seul appel peut suffire.

E_PRO_110 Le système doit être conçu de sorte à ce qu'il soit facile à faire évoluer. Par exemple, l'ajout d'une entité ou d'un champ dans une entité, ne doit pas remettre en cause toute la structure du système.

E_PRO_120 Le système doit s'appuyer au maximum sur des technologies existantes (APIs externes, librairies/framework, algorithmes).

E_PRO_130 Le client doit avoir une gestion des exceptions et des erreurs qui permettent de prévenir l'utilisateur de sa mauvaise utilisation du système.

E_PRO_140 Une documentation complète de votre API doit être disponible et fournie dans la documentation du projet².

E_PRO_150 Les participants du projet doivent expliquer succinctement dans la documentation du projet le choix de la base de données qui va être utilisée.

E_PRO_160 Le projet sera rendu au format tar.gz. Le nom du dossier sera le nom des participants du projet, il contiendra :

- Le code source et le git³ du client
- Le code source et le git du serveur

¹ <https://github.com/angular/angular/blob/main/CONTRIBUTING.md#commit>

² A titre d'exemple : <https://petstore.swagger.io/>

³ Les dépôts git du projet pourront être séparés ou non.

- Un rapport par participant du projet comme décrit pour [E_PRO_60](#)
- La configuration du docker
- La documentation du projet
- Et toute autre documentation que vous jugerez utile.