UE Full Stack

Projet UE Full Stack

1. Introduction

1.1. Objet Du Document

Le présent document décrit le contexte et le périmètre du projet proposé aux étudiants du Master « Génie de l'Informatique Logicielle » pour l'année universitaire 2022-2023 de l'UE Full Stack. Ce document sert de référence pour présenter l'ensemble des exigences fonctionnelles et de qualités associées au projet. 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 boutiques de produits et de catégories associées aux produits.

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 projet 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:
  - BTQ pour les exigences liées à la gestion des boutiques
  - PRD pour les exigences liées à la gestions des produits
  - CAT pour les exigences liées à la gestion des catégories
  - 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
- ROL pour les exigences liées aux rôles des utilisateurs

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.
- Simple Page Application (SPA) : Application Web exécutée dans un navigateur web et dont la navigation est pas gérée de façon autonome (non gérée par un serveur)
- 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
- IHM (Interface Homme Machine) : Composants logiciel permettant aux utilisateur d'interagir avec le système
- SSO (Single Sign On) : Système de gestion centralisé des identités, de l'authentification et des droits.

2.2. Généralités et missions du système

Le système doit permettre la gestion de boutiques de produits et de catégories associées aux produits.

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 boutiques

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

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

E_BTQ_10 Le système permet aux utilisateurs de créer une nouvelle boutique. Lors de la création d'une boutique, les informations minimum à renseigner sont :

- Le nom
- Les horaires d'ouverture pour les jours de la semaine
- Si la boutique est en congé

E_BTQ_20 Le système permet aux utilisateurs de sélectionner une boutique existante et de modifier les champs renseignés lors de la création de la boutique tels que définis par E_BTQ_10.

E_BTQ_30 Le système permet aux utilisateurs de supprimer une boutique.

E_BTQ_40 Le système permet aux utilisateurs d'associer un produit à une seule boutique.

E_BTQ_50 Le système permet d'effectuer une recherche paginé sur toutes les boutiques qui sont présente en affichant :

- Leur nom
- Leur date de création
- Le nombre de produits
- Le nombre de catégorie distincte qui sont associées aux produits de la boutique
- Si la boutique est en congé

E_BTQ_60 Le système permet d'effectuer une recherche sur les boutiques avec un ou une combinaison de filtres qui sont :

- Si la boutique est en congé
- Date de création
  - Après une date précise
  - Avant une date précise
  - Entre deux dates précises

E_BTQ_70 Le système permet d'effectuer une recherche sur les boutiques en triant par :

- Le nom
- La date de création
- Le nombre de produits

E_BTQ_80 Le système permet d'afficher le détail d'une boutique en affichant les champs et les entités tels que définis par E_BTQ_10, E_BTQ_40 et E_BTQ_50

2.4. Gestion des produits

La gestion des produits permet d'effectuer des opérations comme la recherche et le tri selon différents critères, la consultation, la modification ou la suppression des produits.

Les exigences ci-après sont applicables au système de gestion des produits :

E_PRD_10 Le système permet aux utilisateurs de créer un nouveau produit. Lors de la création d'un produit, les informations minimum à renseigner sont :

- Le nom
- Le prix

E_PRD_20 Le système permet aux utilisateurs de sélectionner un produit existant et de modifier les champs renseignés lors de la création du produit tels que définis par E_PRD_10.

E_PRD_30 Le système permet aux utilisateurs de sélectionner un produit existant et d'ajouter une description.

E_PRD_40 Le système permet aux utilisateurs de supprimer un produit.

E_PRD_50 Le système permet aux utilisateurs d'associer une ou plusieurs catégories à un produit.

E_PRD_60 Le système permet d'effectuer une recherche paginé sur l'ensemble des produits appartenant à une boutique en affichant :

- Le nom
- Le prix
- Les catégories qui sont associées
- La description

E_PRD_70 Le système permet de filtrer les produits d'une boutique qui appartiennent à une catégorie.

E_PRD_80 Le système permet aux utilisateurs d'internationaliser le nom et la description du produit.

2.5. Gestion des catégories

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

E_CAT_10 Le système permet aux utilisateurs de créer une nouvelle catégorie. Lors de la création d'une catégorie, les informations minimum à renseigner sont :

- Le nom

E_CAT_20 Le système permet aux utilisateurs de sélectionner une catégorie existante et de modifier le champ renseigné lors de la création de la catégorie tels que définis par E_CAT_10.

E_CAT_30 Le système permet aux utilisateurs de supprimer une catégorie.

E_CAT_40 Le système permet aux utilisateurs d'associer une catégorie à un ou plusieurs produits.

E_CAT_50 Le système permet d'effectuer une recherche paginée sur toutes les catégories qui sont présentes en affichant :

- Le nom

E_CAT_70 Le système permet d'afficher le détail d'une catégorie en affichant le champ tel que défini par E_CAT_10.

2.6. Rôle des utilisateurs (Facultatif)

Les exigences ci-après sont applicables à la définition des rôles de l'application et à l'authentification sur le système.

E_ROL_10 Le projet définit trois rôles principaux :

- un rôle anonyme : utilisateur qui est uniquement en lecture seule sur le projet
- un rôle administrateur : utilisateur ayant accès à toutes les fonctionnalités du projet
- un rôle de vendeur-livreur : utilisateur qui possède toutes les fonctionnalités liée à une seule boutique

E_ROL_20 L'accès à toutes les fonctionnalités du projet (gestion des produits, boutiques, catégories) n'est possible que pour les utilisateurs possédant le rôle d'administrateur.

E_ROL_30 L'accès aux fonctionnalités liées à une seule boutique n'est possible que pour les utilisateurs possédant le rôle de vendeur-livreur.

E_ROL_40 Le rôle de vendeur-livreur permet d'effectuer des opérations de CRUD sur sa boutique

E_ROL_50 Le rôle de vendeur-livreur permet d'effectuer des opérations de CRUD sur tous les produits de sa boutique.

E_ROL_60 Le rôle de vendeur-livreur ne permet pas d'effectuer des opérations de modification sur les catégories.

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 boutiques détaillées en 2.3.

E_API_20 Le système met à disposition une API REST permettant d'accéder aux gestion des produits détaillées en 2.4.

E_API_30 Le système met à disposition une API REST permettant d'accéder aux gestion des catégories en 2.5.

E_API_40 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 opération sur l'IHM du système doit se faire en moins de 5 secondes (affichage d'une page, etc..).

E_TEC_40 Facultatif La sécurisation du serveur développé, la gestion du login et des droits doivent se faire en intégrant un SSO Open Source.

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 intégrent :

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

E_PRO_20 Lors du lancement des conteneurs docker en local, le projet devra alimenter le système de données d'exemple.

- Au moins 10 boutiques
- Au moins 100 produits
- Au moins 10 catégories

E_PRO_30 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 Lors du développement du projet, du CI/CD doit être mis en place pour toutes les tâches répétitives. Par exemple, lors d'un hotfix avec git flow 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 commits[1].

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

E_PRO_90 Le serveur doit avoir une gestion des exceptions et des erreurs permettant de prévenir le client 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 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_MAN_140 Une documentation complète de votre API doit être disponible et fournie dans la documentation du projet[2].

E_PRO_150 Les participants du projet doivent expliquer succintement 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 git[3] 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.