******

***Projet***

***Architecture Orientée Services :***

***Victory Vault Tunisia***

***Elaborée par :***

***Najd Mseddi***

***Marwen Kammoun***

***Mohamed Youssef Ben Chehida***

***Supervisé par : prof. Ahmad Jmal***

**Table des matières**

[**Liste des figures :** 3](#_Toc155916502)

[**1.** **Introduction :** 4](#_Toc155916503)

[**2.** **Description du projet :** 5](#_Toc155916504)

[**3.** **Technologies utilisées :** 6](#_Toc155916505)

[**a.** **Backend : Spring Boot** 6](#_Toc155916506)

[**b.** **Eureka, Gateway et LoadBalancer** 7](#_Toc155916507)

[**c.** **Frontend : Angular avec Material** 7](#_Toc155916508)

[**d.** **Base de données : MySQL** 7](#_Toc155916509)

[**e.** **OpenAPI (Swagger) et Postman** 7](#_Toc155916510)

[**4.** **Cahier des charges :** 8](#_Toc155916511)

[**5.** **Analyse des exigences :** 9](#_Toc155916512)

[**6.** **Architecture du système :** 10](#_Toc155916513)

[**a.** **Microservice : SeasonService** 11](#_Toc155916514)

[**b.** **Microservice : MatchesService** 11](#_Toc155916515)

[**7.** **Conception du Système :** 13](#_Toc155916516)

[**8.** **Maquettes d’interfaces :** 14](#_Toc155916517)

[**9.** **Implémentation :** 14](#_Toc155916518)

[**10.** **Conclusion :** 16](#_Toc155916519)

[**11.** **Références :** 17](#_Toc155916520)

# **Liste des figures :**

[Figure 1: Architecture du système 10](#_Toc155916358)

[Figure 2: Conception de la base de données 13](#_Toc155916359)

# **Introduction :**

Entrez dans l'univers fascinant du projet "Victory Vault Tunisia", une initiative ambitieuse destinée à moderniser la gestion des résultats de football en Tunisie. Le marché du football tunisien requiert une touche informatique innovante, et notre équipe de trois passionnés relève le défi avec enthousiasme.

La nécessité d'un développement informatique dans le domaine du football en Tunisie se fait de plus en plus sentir. Pour répondre à cette demande, le projet "Victory Vault Tunisia" prend son envol. Avec trois esprits dévoués à sa barre, l'objectif est clair : révolutionner la manière dont les résultats des matchs de football sont traités et gérés dans le pays.

Se penchant sur les problématiques, notre équipe a identifié les lacunes existantes dans la gestion actuelle des résultats de la Ligue Pro1 du championnat de Tunisie de Football. Ces défis sont autant de moteurs pour notre projet, nous incitant à créer une plateforme non seulement robuste mais aussi conviviale pour la saisie et la gestion des informations relatives aux matchs.

Les technologies qui alimentent notre ambition sont à la pointe, avec une architecture de microservices exploitant la puissance de Spring Boot pour le backend, l'agilité d'Angular avec Material pour le frontend, et la solidité de MySQL pour la base de données. Deux piliers, SeasonService et MatchesService, sont coordonnés harmonieusement par Eureka, Gateway et LoadBalancer.

Chacun dans notre équipe apporte une contribution unique et essentielle. La créativité du frontend, la robustesse du backend et la stabilité de la base de données s'entremêlent pour donner vie à notre vision. Chaque membre de l'équipe joue un rôle clé, apportant son expertise pour surmonter les défis et faire avancer le projet.

La structure du projet est pensée avec soin, garantissant une plateforme fluide et intuitive. La simplicité de l'interface cache la complexité du système, offrant aux utilisateurs une expérience transparente pour la saisie et la gestion des données liées aux matchs de football.

Les contributions majeures de notre projet se révèlent à travers plusieurs aspects. Des résultats des matchs captivants aux détails minutieux des rencontres, en passant par le classement des équipes et la composition des formations, notre plateforme offre une vision globale du monde du football tunisien. En repoussant les limites, "Victory Vault Tunisia" s'inscrit comme un acteur majeur dans l'évolution numérique du football en Tunisie.

# **Description du projet :**

"Victory Vault Tunisia" est un projet innovant qui vise à transformer la gestion des résultats des matchs de football en Tunisie. Il s'agit d'une plateforme numérique conçue pour gérer efficacement les informations relatives aux matchs de la Ligue Pro1 du championnat de Tunisie de Football.

Le projet est structuré autour de deux microservices principaux : SeasonService et MatchesService. Le microservice SeasonService est responsable de la gestion des informations relatives aux différentes saisons de football, tandis que le microservice MatchesService se concentre sur la gestion des détails des matchs.

La plateforme permet aux utilisateurs de saisir et de consulter diverses informations, y compris les résultats des matchs, les détails des matchs, le classement des équipes et la composition des équipes. Par exemple, les utilisateurs peuvent consulter les résultats d'une journée spécifique, obtenir des détails sur un match particulier, voir le classement des équipes après une journée spécifique, et consulter la composition d'une équipe pour une saison donnée.

La plateforme est construite en utilisant une architecture de microservices avec Spring Boot pour le backend, Angular avec Material pour le frontend, et MySQL pour la base de données. Elle utilise également Eureka pour la découverte de services, Gateway pour le routage des requêtes, et un LoadBalancer pour assurer une distribution équilibrée des requêtes entre les instances de services.

En somme, "Victory Vault Tunisia" est une solution complète pour la gestion des résultats des matchs de football, offrant une interface conviviale et des fonctionnalités robustes pour répondre aux besoins des utilisateurs.

# **Technologies utilisées :**

Dans le cadre de ce projet, nous avons utilisé une variété de technologies pour développer une solution robuste et efficace. Voici une description détaillée des technologies utilisées :

## **Backend : Spring Boot**

Nous avons utilisé Spring Boot pour le développement du backend. Spring Boot est un framework Java qui simplifie la configuration et le bootstrap d'une application Spring. Il offre une multitude de fonctionnalités telles que l'injection de dépendances et la sécurité, ce qui nous a permis de nous concentrer sur le développement des fonctionnalités métier.

Dans le cadre de ce projet, nous avons développé deux microservices : SeasonService et MatchesService. Ces services sont responsables de la gestion des saisons de football et des matchs respectivement.

## **Eureka, Gateway et LoadBalancer**

Pour orchestrer nos microservices, nous avons utilisé Eureka, Gateway et LoadBalancer. Eureka est un service de découverte qui permet aux microservices de localiser les autres services dans l'architecture. Gateway est utilisé pour le routage des requêtes, tandis que LoadBalancer assure une distribution équilibrée des requêtes entre les instances de services.

## **Frontend : Angular avec Material**

Pour le développement du frontend, nous avons utilisé Angular, un framework JavaScript développé par Google. Angular nous a permis de créer une interface utilisateur interactive et réactive. Nous avons également utilisé Angular Material, une bibliothèque de composants d'interface utilisateur, pour créer une interface utilisateur moderne et attrayante.

## **Base de données : MySQL**

Pour la persistance des données, nous avons utilisé MySQL, un système de gestion de base de données relationnelle. MySQL est largement utilisé en raison de sa fiabilité, de sa performance et de sa facilité d'utilisation.

## **OpenAPI (Swagger) et Postman**

Pour la documentation et le test de nos API, nous avons utilisé OpenAPI (Swagger) et Postman.

* OpenAPI (Swagger) :

OpenAPI, anciennement connu sous le nom de Swagger, est un cadre de spécification pour la conception, la construction, la documentation et la consommation d'API REST. Il fournit un ensemble d'outils pour générer des spécifications d'API à partir du code source et pour générer du code à partir de spécifications d'API. Dans notre projet, nous avons utilisé Swagger pour documenter nos API, ce qui a facilité leur utilisation et leur compréhension par les développeurs et les utilisateurs.

* Postman :

Postman est un outil populaire pour tester les API. Il permet d'envoyer des requêtes HTTP à une API et de visualiser les réponses. Postman offre une interface utilisateur conviviale qui facilite la création, l'envoi et la gestion des requêtes HTTP. Dans notre projet, nous avons utilisé Postman pour tester nos API et vérifier qu'elles fonctionnent correctement.

# **Cahier des charges :**

Le cahier des charges de notre projet "Victory Vault Tunisia" est centré sur la gestion des résultats des matchs de football pour le championnat de la Tunisie de Football (Ligue Pro1). Voici les détails :

* Gestion des résultats de matchs de football :

Notre système doit permettre la saisie et la gestion des informations nécessaires à l’établissement des tableaux de résultats pour chaque journée de la saison.

* Détails d'un match :

Le système doit également permettre la saisie et la gestion des détails spécifiques à chaque match, y compris le stade, l'arbitre, la date du match, le nombre de spectateurs, les buteurs, les entraîneurs et la composition des équipes qui ont joué.

* Classement après chaque journée :

Le système doit permettre la saisie et la gestion du classement des équipes après chaque journée.

* Composition d'une équipe :

Le système doit permettre la saisie et la gestion de la composition d'une équipe pour une saison donnée.

Ces exigences constituent la base de notre cahier des charges pour le projet "Victory Vault Tunisia". Notre objectif est de développer un système qui répond à ces exigences de manière efficace et conviviale.

# **Analyse des exigences :**

L'analyse des exigences est une étape cruciale dans tout projet de développement logiciel. Elle nous permet de comprendre en profondeur ce que le système est censé faire et comment il doit fonctionner. Voici comment nous avons procédé à l'analyse des exigences pour notre projet "Victory Vault Tunisia" :

* Compréhension du cahier des charges :

Nous avons commencé par une lecture approfondie du cahier des charges fourni. Cela nous a permis de comprendre les objectifs du projet, les fonctionnalités attendues du système, et les contraintes techniques et fonctionnelles.

* Identification des acteurs :

Nous avons identifié les différents acteurs qui interagiront avec le système. Cela comprend les utilisateurs qui saisiront et consulteront les informations sur les matchs de football, ainsi que les administrateurs qui géreront le système.

* Définition des fonctionnalités :

Nous avons défini les différentes fonctionnalités que le système doit offrir. Cela comprend la gestion des résultats des matchs de football, la gestion des détails des matchs, la gestion du classement des équipes après chaque journée, et la gestion de la composition d'une équipe pour une saison donnée.

# **Architecture du système :**

Notre système est structuré autour de deux microservices principaux, chacun ayant une responsabilité spécifique dans le système global.

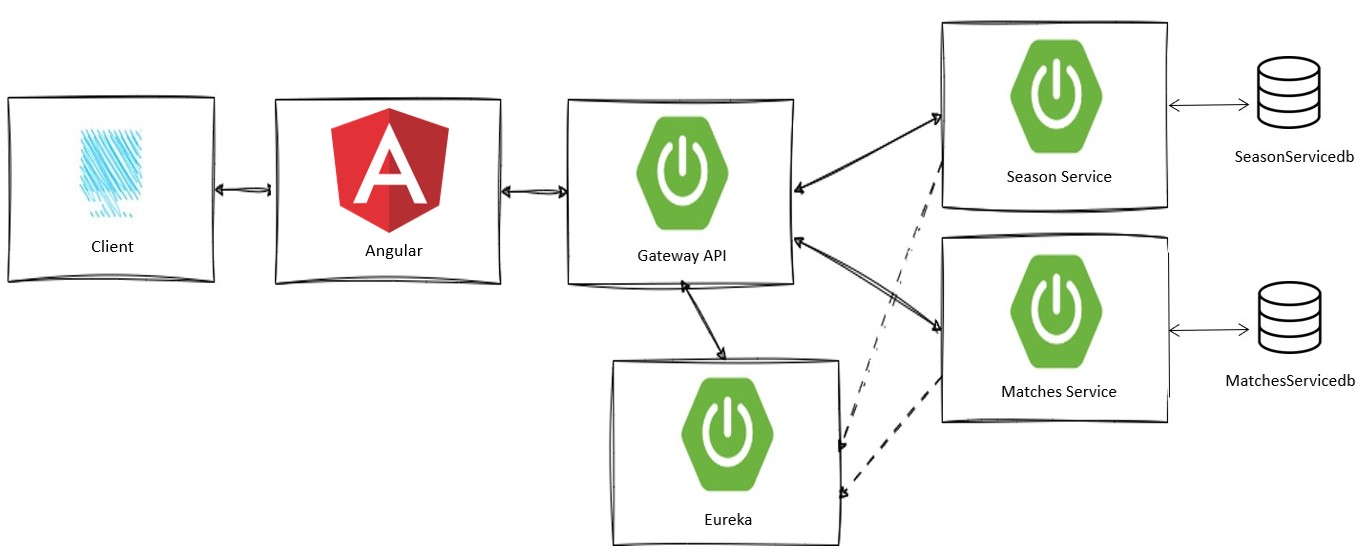


Figure 1: Architecture du système

## **Microservice : SeasonService**

Le premier microservice, SeasonService, est responsable de la gestion des saisons de football. Il contient trois contrôleurs principaux : SeasonController, RoundController et StandingController. Chacun de ces contrôleurs a une responsabilité spécifique liée à la gestion des saisons de football.

- SeasonController : Responsable de la gestion des saisons de football. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à une saison de football.

- RoundController : Responsable de la gestion des journées de chaque saison. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à une journée de football.

- StandingController : Responsable de la gestion du classement des équipes après chaque journée. Il permet de créer, lire, mettre à jour et supprimer les informations relatives au classement des équipes.

## **Microservice : MatchesService**

Le deuxième microservice, MatchesService, est responsable de la gestion des matchs de football. Il contient plusieurs contrôleurs, chacun ayant une responsabilité spécifique liée à la gestion des matchs de football.

- CoachController : Responsable de la gestion des entraîneurs des équipes. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à un entraîneur.

- LineupController : Responsable de la gestion de la composition des équipes qui ont joué un match. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à la composition d'une équipe.

- MatchController : Responsable de la gestion des matchs. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à un match.

- PlayerController : Responsable de la gestion des joueurs. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à un joueur.

- PlayerInPositionController : Responsable de la gestion des positions des joueurs dans un match. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à la position d'un joueur dans un match.

- RefereeController : Responsable de la gestion des arbitres des matchs. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à un arbitre.

- ReplacementController : Responsable de la gestion des remplacements de joueurs pendant un match. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à un remplacement de joueur pendant un match.

- ScorerController : Responsable de la gestion des buteurs d'un match. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à un buteur d'un match.

- StadiumController : Responsable de la gestion des stades où se déroulent les matchs. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à un stade.

- TeamController : Responsable de la gestion des équipes. Il permet de créer, lire, mettre à jour et supprimer les informations relatives à une équipe.

Ces deux microservices interagissent entre eux via Eureka, Gateway et LoadBalancer. Eureka permet aux microservices de se découvrir mutuellement, Gateway sert de point d'entrée unique pour toutes les requêtes entrantes, et LoadBalancer assure une distribution équilibrée des requêtes entre les instances de services.

Le frontend de notre système est construit en utilisant Angular. Il interagit avec les microservices via Gateway pour récupérer les données nécessaires et les présenter à l'utilisateur.

Enfin, nous utilisons MySQL pour la persistance des données. Chaque microservice a sa propre base de données pour assurer l'indépendance des données.

# **Conception du Système :**

La conception de notre base de données est cruciale pour assurer des performances optimales et une organisation efficace des données.

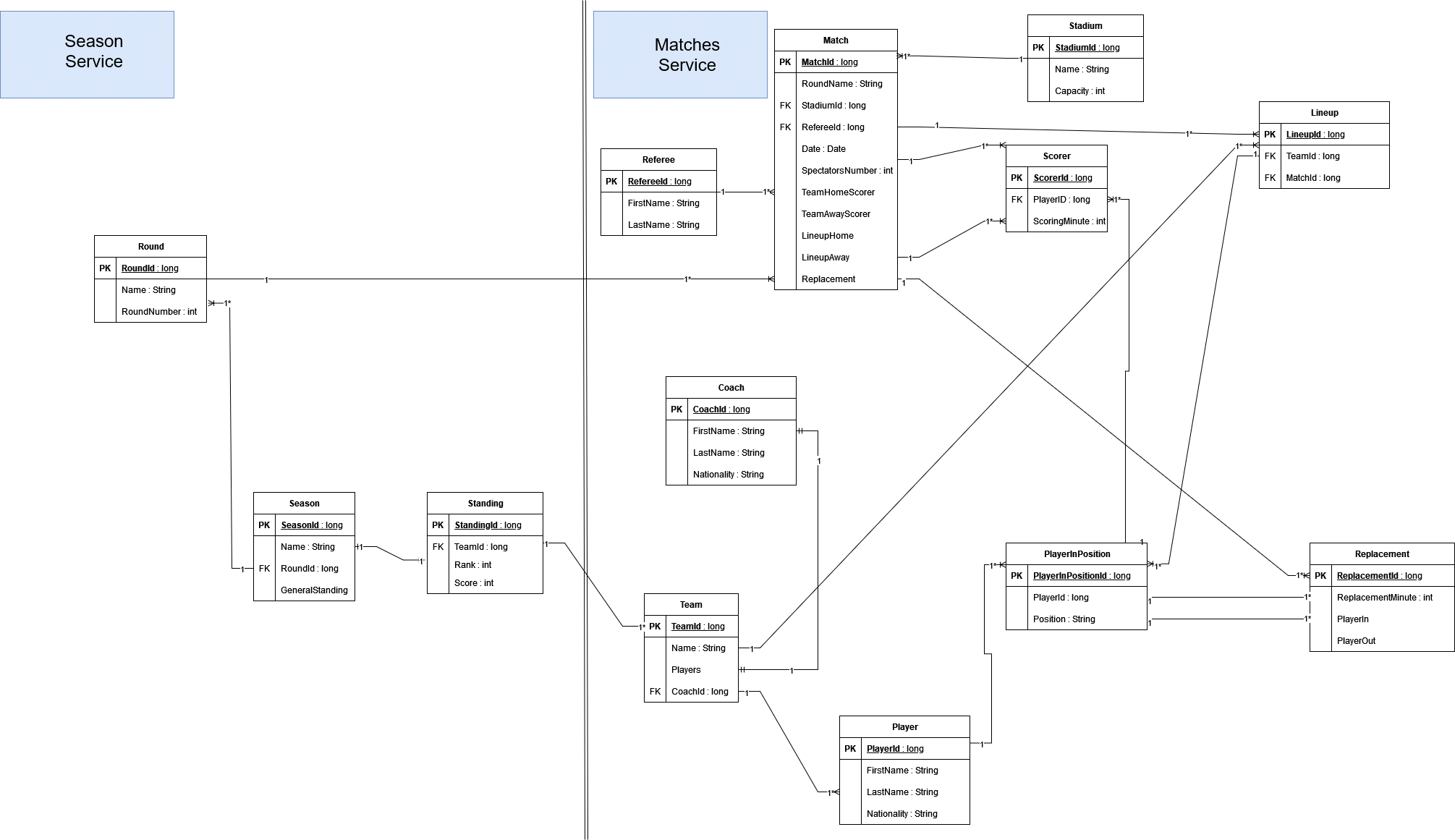


Figure 2: Conception de la base de données

# **Maquettes d’interfaces :**

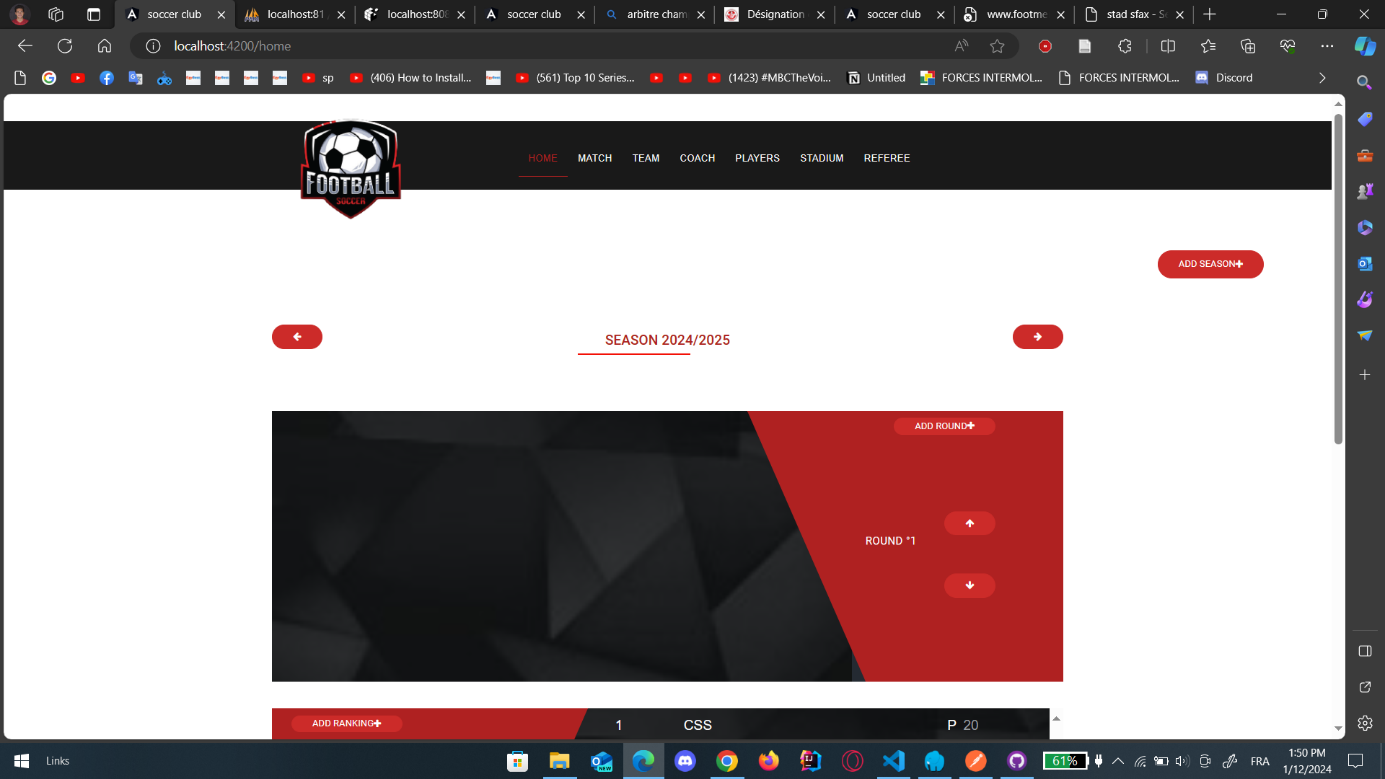
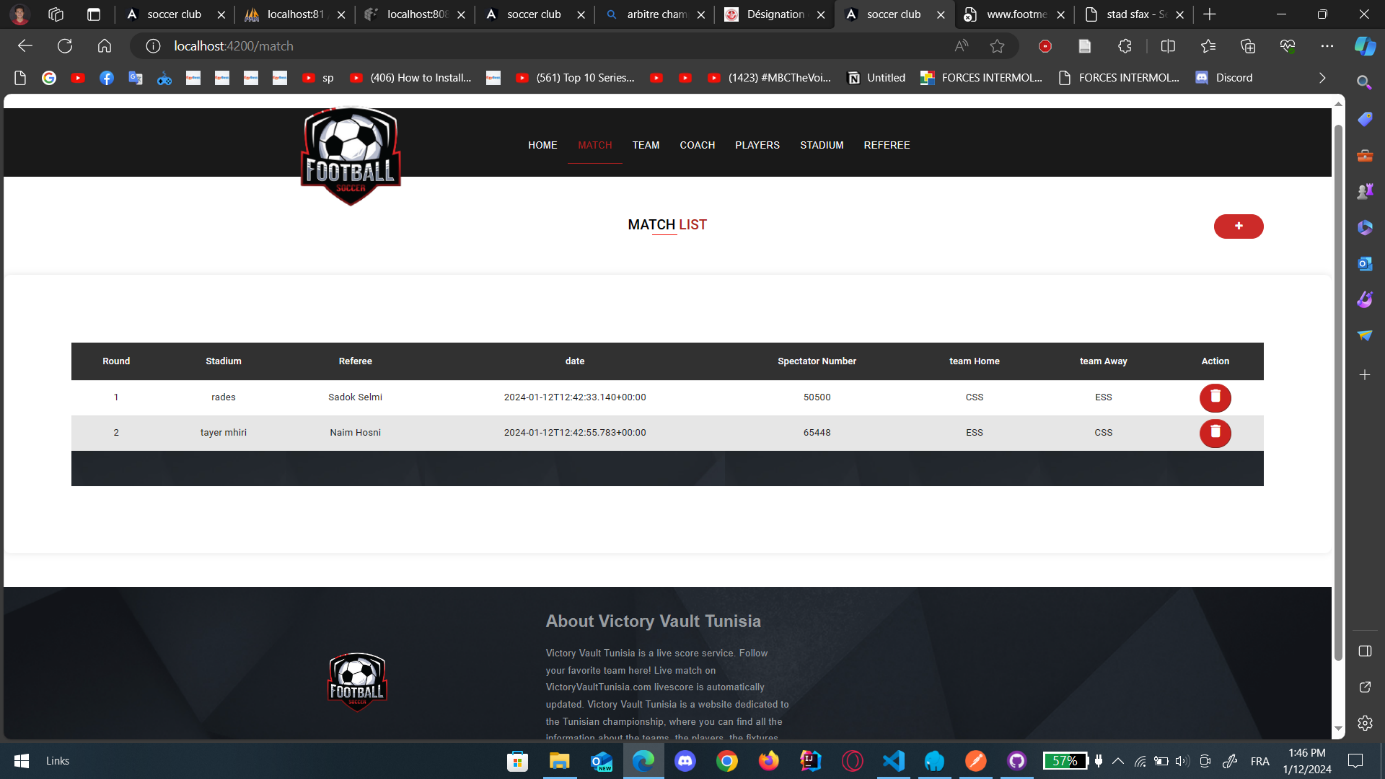
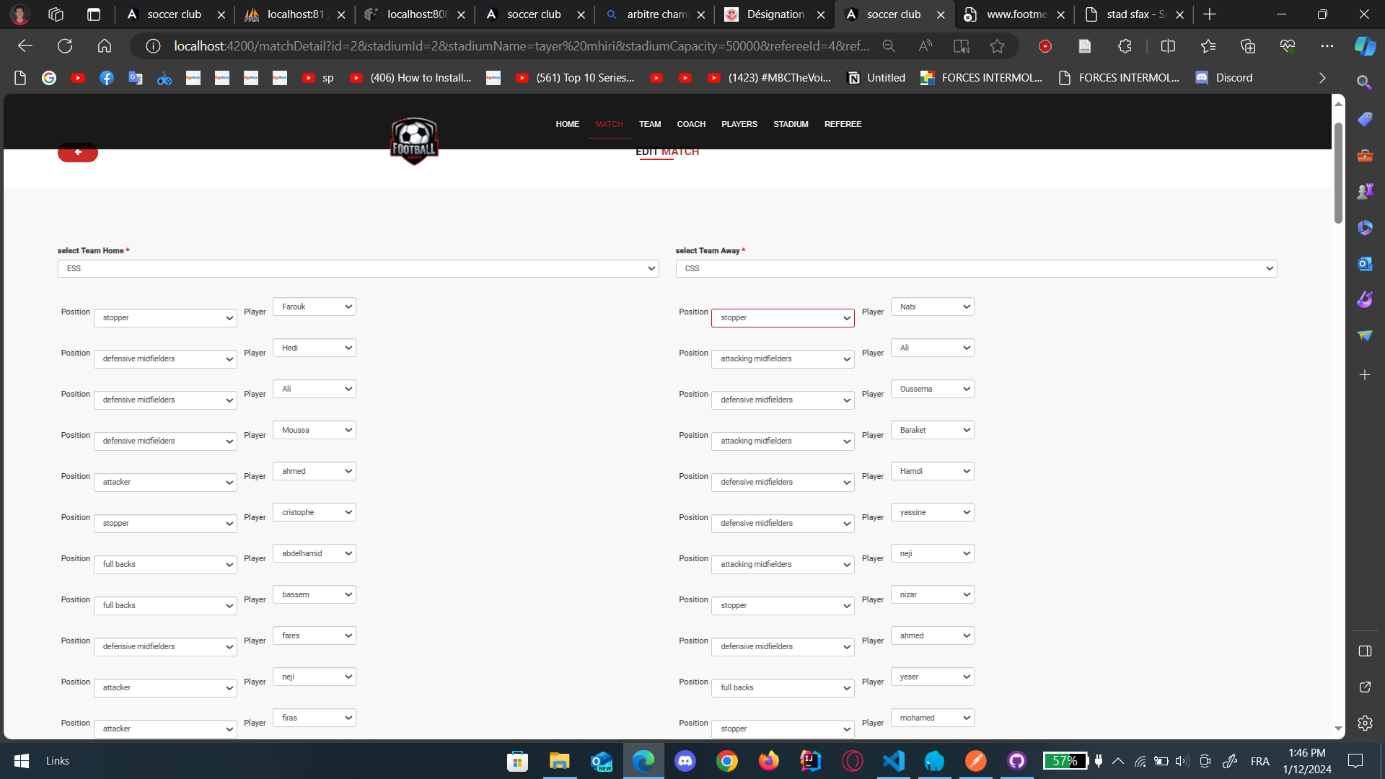
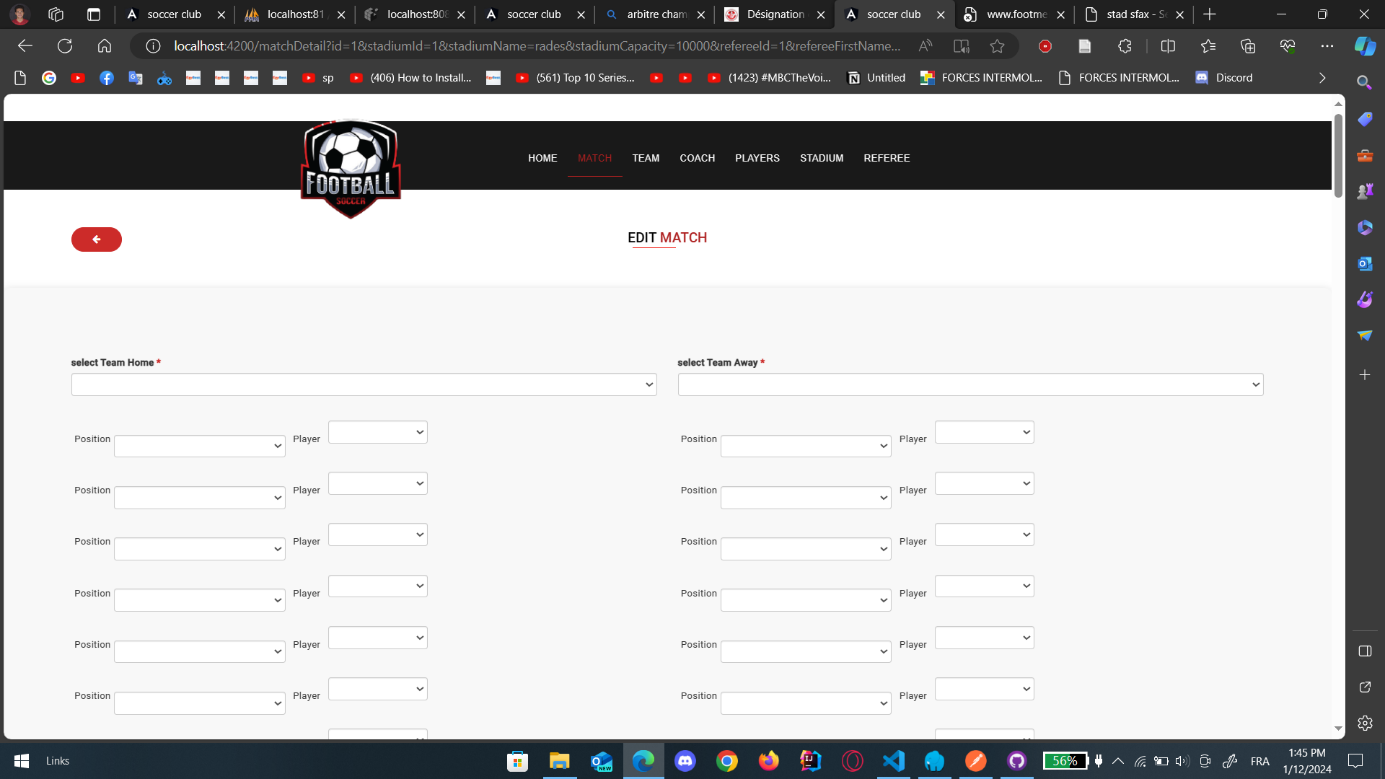
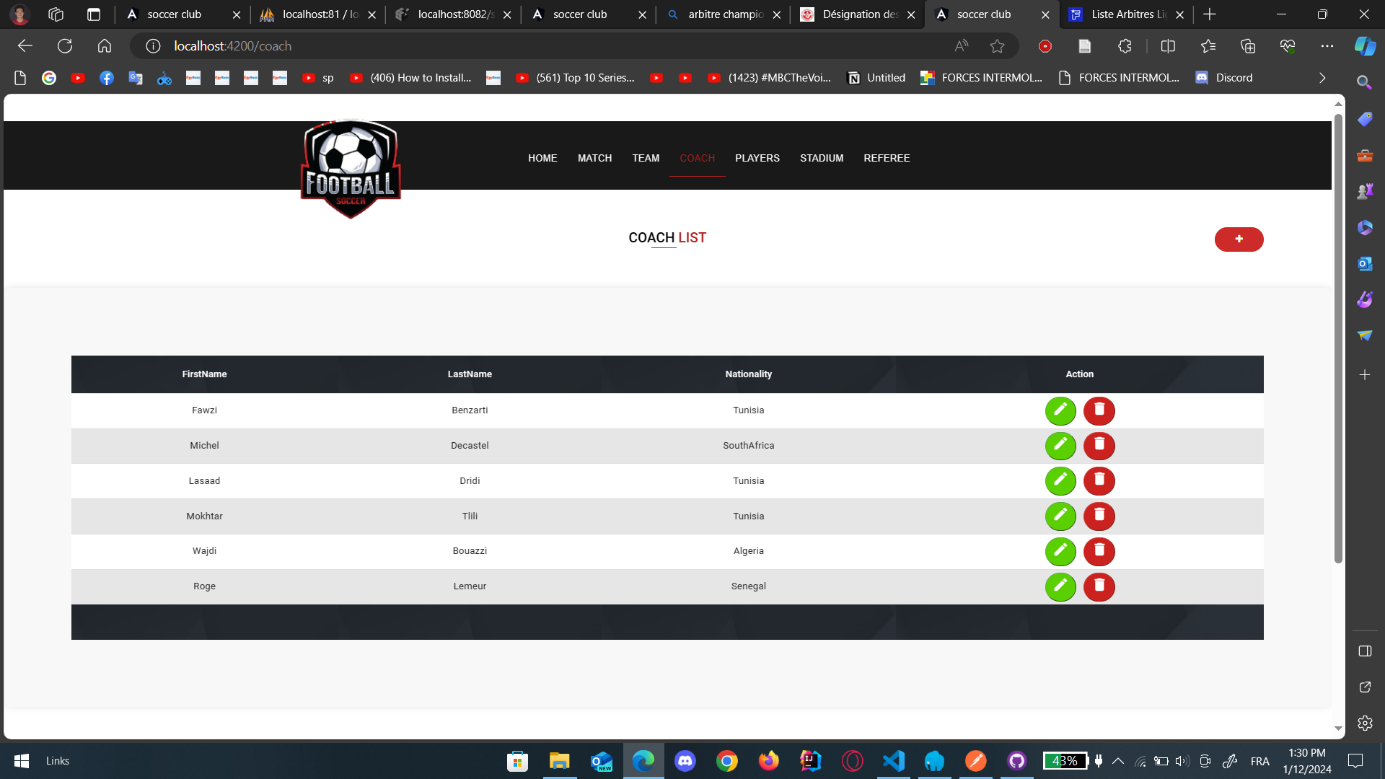
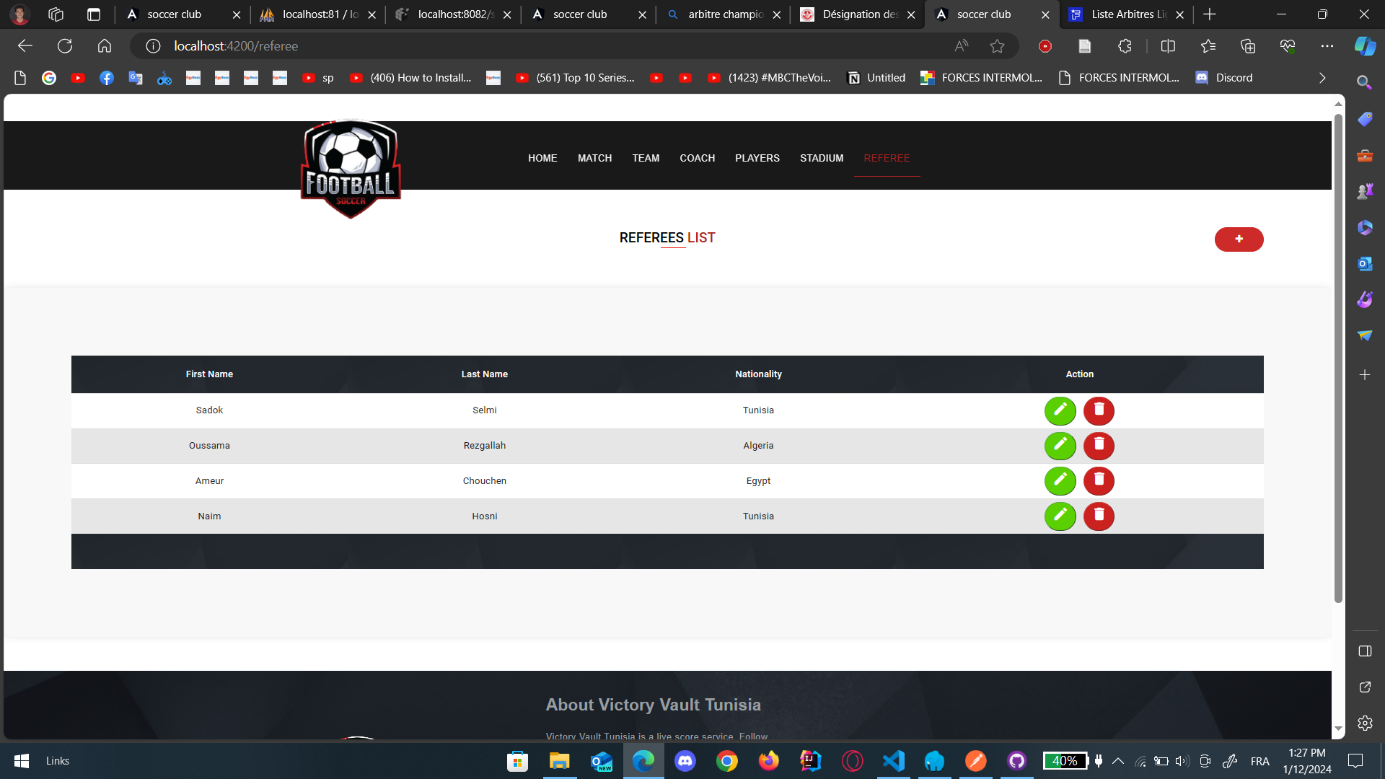
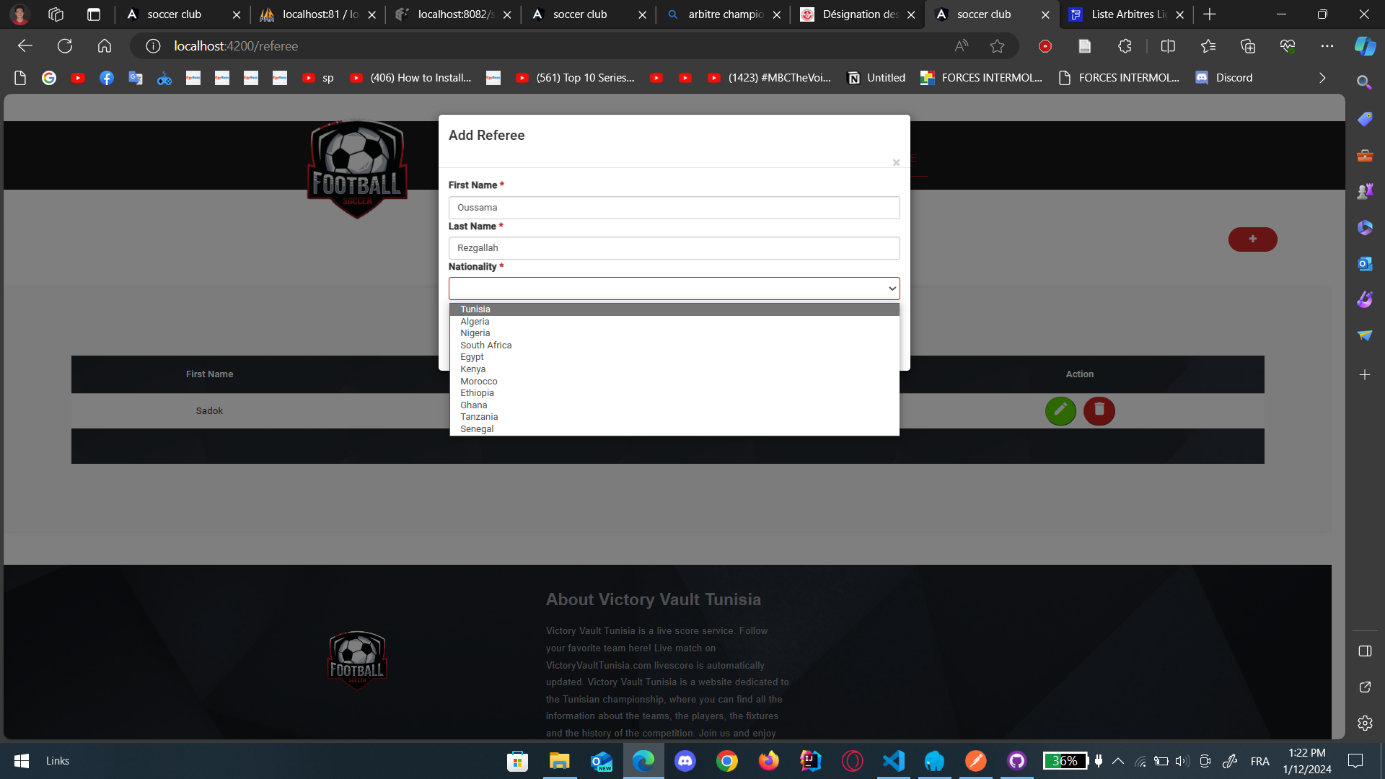
Dans cette section, nous présentons les maquettes d'interfaces pour notre système "Victory Vault Tunisia". Ces maquettes donnent une idée de l'apparence de notre système et de la manière dont les utilisateurs interagiront avec lui.

- Page d'accueil : La page d'accueil présente une vue d'ensemble du système. Elle comprend des liens vers les différentes fonctionnalités du système, comme la visualisation des matchs, la consultation du classement, etc.

- Page des matchs : Cette page affiche la liste des matchs de football. Les utilisateurs peuvent sélectionner un match pour voir plus de détails, comme le score, les buteurs, etc.

- Page des équipes : Cette page affiche la liste des équipes. Les utilisateurs peuvent sélectionner une équipe pour voir plus de détails, comme la composition de l'équipe, les joueurs, etc.

Ces maquettes ont été conçues pour être intuitives et faciles à utiliser, afin de fournir une expérience utilisateur optimale.



# **Implémentation :**

L'implémentation de notre système a été réalisée en suivant une approche modulaire, en se concentrant sur chaque microservice et chaque contrôleur individuellement.

Chaque contrôleur dans les microservices a été développé pour gérer des opérations CRUD spécifiques liées à son domaine. Par exemple, dans le microservice SeasonService, le SeasonController gère les opérations CRUD pour les saisons de football, le RoundController pour les journées de chaque saison, et le StandingController pour le classement des équipes après chaque journée.

De même, dans le microservice MatchesService, chaque contrôleur a une responsabilité spécifique liée à la gestion des matchs de football. Par exemple, le MatchController gère les opérations CRUD pour les matchs, le PlayerController pour les joueurs, et ainsi de suite.

Chaque contrôleur a été testé individuellement pour s'assurer qu'il fonctionne correctement et qu'il répond aux exigences du cahier des charges. De plus, nous avons effectué des tests d'intégration pour vérifier que les microservices interagissent correctement entre eux.

Nous avons également utilisé des patrons de conception pour structurer notre code de manière efficace et maintenable.

* Patron de conception Adapter :

Nous avons utilisé le patron de conception Adapter pour transformer les entités en DTOs (Data Transfer Objects) et vice versa. Cela a permis d'isoler la logique de transformation des données de la logique métier principale, rendant notre code plus propre et plus facile à maintenir.

* Patron de conception Builder :

Nous avons utilisé le patron de conception Builder pour construire des objets complexes étape par étape. Cela nous a permis de créer des objets avec une grande variété de configurations possibles tout en gardant la logique de construction séparée de la logique métier principale.

# **Conclusion :**

En conclusion, le projet "Victory Vault Tunisia" a été une expérience enrichissante et instructive. Nous avons réussi à développer un système robuste et efficace pour la gestion des résultats des matchs de football pour le championnat de la Tunisie de Football (Ligue Pro1). Notre site web est une opportunité pour informatiser les données de la ligue Tunisienne. Notre application peut être améliorer à travers l’intégration du web scrapping et machine Learning pour donner des prédictions sur la journée de la ligue.

Notre approche basée sur les microservices nous a permis de développer un système modulaire et évolutif. Chaque microservice a été développé et testé individuellement, ce qui a facilité la maintenance et l'évolution du système.

Nous sommes impatients de voir notre système en action et nous sommes convaincus qu'il apportera une valeur significative à la gestion des résultats des matchs de football en Tunisie. Notre application représente une vraie solution pour donner une vision sur les statistiques de la ligue Tunisienne.

# **Références :**

<https://www.redhat.com/fr/topics/microservices?fbclid=IwAR2wgApHuZy-vg67YEIsdOo0KrjKXR4fhjpCZsCi54vsEUhKkMZfWAICKVg>

<https://www.softfluent.fr/blog/architecture-logicielle-pour-application/>

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

<https://springjava.com/spring-boot/use-lombok-in-spring-boot>

<https://www.codeflow.site/fr/article/spring-vs-spring-boot>

<https://stackoverflow.com>

<https://www.geeksforgeeks.org/difference-between-json-and-xml/>

<https://angular.io/docs>

<https://openclassrooms.com/fr/courses/7471261-debutez-avec-angular?archived-source=4668271>

<https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp>

<https://fr.wikipedia.org/wiki/Portail:Accueil>

<https://medium.com>