******

***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**

[**1.** **Introduction :** 3](#_Toc155751542)

[**2.** **Description du projet :** 3](#_Toc155751543)

[**3.** **Technologies utilisées :** 4](#_Toc155751544)

[**a.** **Backend : Spring Boot** 5](#_Toc155751545)

[**b.** **Eureka, Gateway et LoadBalancer** 5](#_Toc155751546)

[**c.** **Frontend : Angular avec Material** 5](#_Toc155751547)

[**d.** **Base de données : MySQL** 6](#_Toc155751548)

[**4.** **Architecture du système :** 6](#_Toc155751549)

[**a.** **Microservice : SeasonService** 6](#_Toc155751550)

[**b.** **Microservice : MatchesService** 7](#_Toc155751551)

[**5.** **Conception du Système :** 8](#_Toc155751552)

[**a.** **Conception de la base de données** 8](#_Toc155751553)

[**b.** **Diagramme de classe** 9](#_Toc155751554)

[**6.** **Maquettes d’interfaces :** 9](#_Toc155751555)

[**7.** **Cahier des charges :** 10](#_Toc155751556)

[**8.** **Implémentation :** 11](#_Toc155751557)

[**9.** **Conclusion :** 12](#_Toc155751558)

[**10.** **Références :** 13](#_Toc155751559)

# **Introduction :**

Ce rapport présente le projet "Victory Vault Tunisia", une initiative ambitieuse visant à révolutionner la manière dont les résultats des matchs de football sont gérés en Tunisie. Réalisé par une équipe de trois collègues dévoués, ce projet vise à créer une plateforme robuste et efficace pour la gestion des résultats de la Ligue Pro1 du championnat de Tunisie de Football.

Le projet est construit 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. Il comprend deux microservices principaux : SeasonService et MatchesService, qui sont orchestrés à l'aide de Eureka, Gateway et LoadBalancer.

L'objectif principal de ce projet est de fournir une plateforme fiable et conviviale pour la saisie et la gestion des informations relatives aux matchs de football, y compris les résultats des matchs, les détails des matchs, le classement des équipes, et la composition des équipes.

Ce rapport détaille le processus de réalisation de ce projet, de la compréhension du cahier des charges à l'implémentation et aux tests du système. Il met en évidence les défis rencontrés, les solutions adoptées, et les résultats obtenus, offrant ainsi un aperçu complet du travail accompli.

# **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.

# **Architecture du système :**

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

## **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.

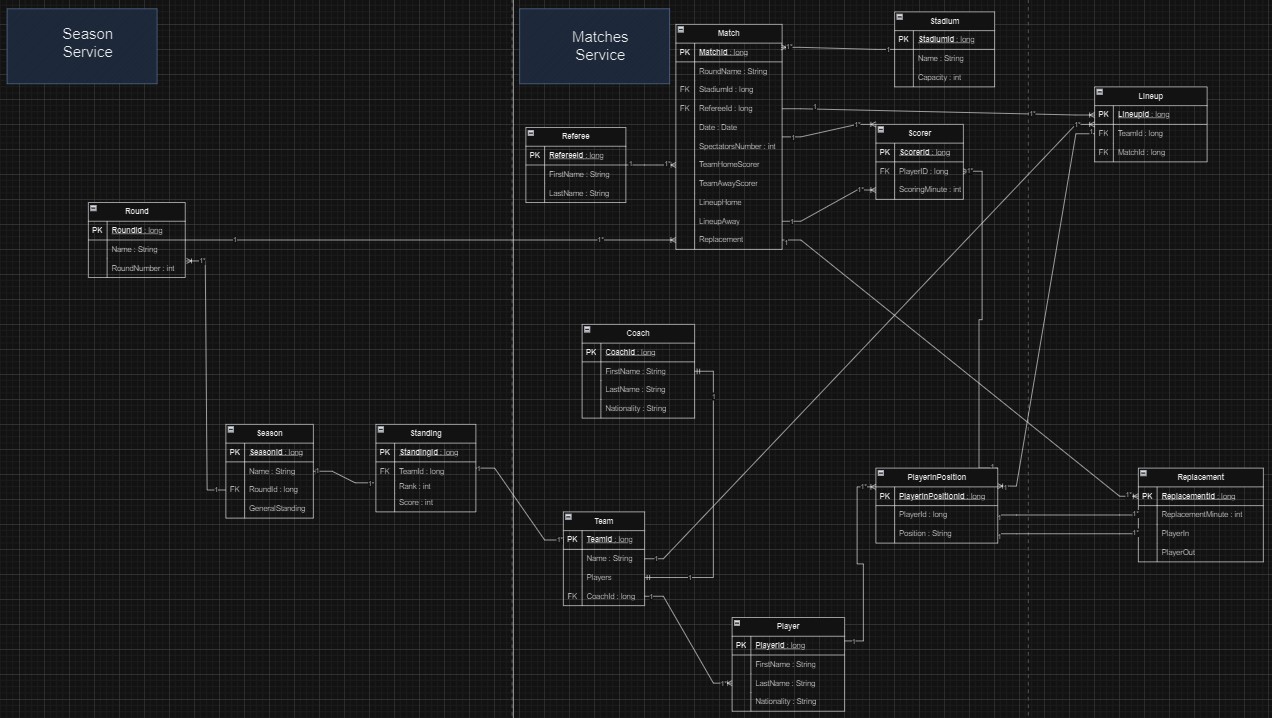
Enfin, nous utilisons MySQL pour la persistance des données. Chaque microservice a sa propre base de données pour assurer l'indépendance des données.

# **Conception du Système :**

Cette section décrit la conception détaillée de notre système, y compris la conception de la base de données et la conception des classes.

## **Conception de la base de données**

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



## **Diagramme de classe**

# **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 du classement : Cette page affiche le classement des équipes après chaque journée. Les utilisateurs peuvent voir le nombre de points de chaque équipe, leur position dans le classement, 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 statistiques des joueurs, etc.

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

# **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.

# **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 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.

# **Références :**

Dans le cadre de ce projet, nous avons utilisé une variété de ressources pour nous aider à comprendre et à mettre en œuvre les différentes technologies et concepts. Voici quelques-unes des principales références que nous avons utilisées :

- Documentation officielle de Spring Boot : Nous avons utilisé la documentation officielle de Spring Boot pour comprendre comment configurer et utiliser Spring Boot pour développer nos microservices.

- Documentation officielle d'Angular : La documentation officielle d'Angular a été une ressource précieuse pour comprendre comment utiliser Angular pour développer notre frontend.

- Tutoriels et articles de blog : Nous avons consulté divers tutoriels et articles de blog pour comprendre comment mettre en œuvre des fonctionnalités spécifiques, résoudre des problèmes et optimiser notre code.

- Forums de discussion et plateformes Q&A : Des plateformes comme Stack Overflow ont été très utiles pour trouver des solutions à des problèmes spécifiques que nous avons rencontrés lors du développement.

- Livres et cours en ligne : Nous avons également utilisé divers livres et cours en ligne pour approfondir notre compréhension des technologies et des concepts que nous avons utilisés.