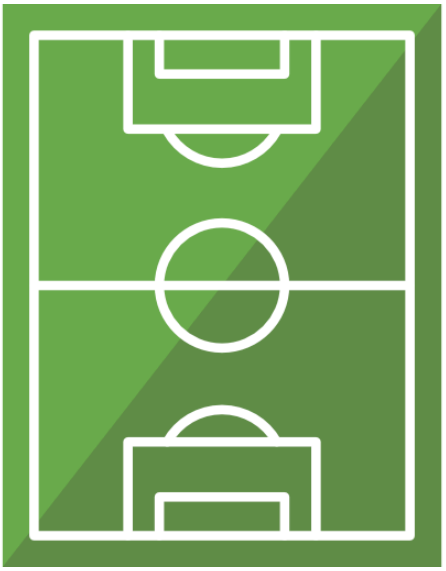
Projet - The Real Deal

Architecture Orientée Services



Sommaire

Sujet

La société ***TRD*** *(The Real Deal),* spécialisée dans les jeux d'argent en ligne (hippisme, casino…), vous contacte dans le cadre de son projet de lancement d'une plateforme de paris sportifs en ligne.

TRD souhaite avancer progressivement dans son projet et compte sur vous pour concevoir et réaliser une solution logicielle flexible et robuste, capable de s'adapter à la charge et à des évolutions futures.

En effet, l'aspect événementiel des rencontres sportives associées aux paris génère d'importantes fluctuations du nombre de requêtes simultanés, avec des pics lors de soirées de match.

D'autre part, ***TRD*** souhaite proposer à ses utilisateurs de parier sur les rencontres de *Champions League* (compétition phare de football en Europe). Par la suite, si le projet s'avère fonctionnel et rentable, ***TRD*** souhaite s'étendre à d'autres compétitions, voire à d'autres sports (Tennis, Basket…).

L'application est destinée aux parieurs disposant d'un compte ***TRD*** et aux bookmakers de ***TRD*** (salariés de l'entreprise chargés de fixer les côtes).

Votre mission consiste à recueillir et analyser les besoins en vous imprégnant du domaine métier, à concevoir une solution sur-mesure et à réaliser une application reposant sur une architecture distribuée (à base de micro services)***.***

La portée du projet est autant **méthodologique, conceptuelle** que **technique**. Vous serez évalués sur ces 3 aspects.

Spécifications fonctionnelles générales

L'application permettra aux utilisateurs anonymes de créer un compte parieur TRD, afin de déposer une somme d'argent sous forme de cagnotte, utilisable pour effectuer des paris.

L'application permettra d'une part, aux bookmakers de TRD de sélectionner des matchs, de saisir et mettre à jour des côtes associées aux matchs, et d'autre part, aux parieurs de réaliser des paris sur les matchs proposés (paris simples et paris combinés).

Pour un même match, la valeur des côtes pourra évoluer au fil du temps, en fonction de l'actualité sportive.
TRD pourra également influencer les parieurs en proposant des côtes attractives ou en mettant en avant certaines rencontres.

L'application permettra de suivre le résultat des matchs.

Les parieurs dont les pronostics se révèleront exacts recevront automatiquement leur gains, dont la valeur sera calculée au moment de la validation du paris.

Modalités

- Travail en équipe de 2 à 3 personnes (formation des groupes librement mais uniquement au sein de votre promo respective ACSI ou SID).
- Travail personnel uniquement : pas de copier / coller, usage nul ou modéré de l'IA générative (ChatGPT, Github Copilot, Gemini…).
- Délai de rendu : **20/12/2024** (pénalités en cas de retard).
- Livrables attendus :
    - **Dossier de conception :** documents d'analyse du Domaine (identification des différents services et processus métier) et de conception de l'Architecture sous forme de diagrammes UML et d'une note de synthèse rédigée.
    - **Application** rendue sous forme de dépôt Git distant (Github ou Gitlab).

      Si votre dépôt Git est privé, ajouter ***shrp777*** (alex@shrp.dev) en tant que collaborateur. Votre projet contiendra un fichier README.md où seront indiqués l'identité et le rôle joué par chaque membre de l'équipe.
- **Réalisation du travail pendant les séances de TD et sur votre temps libre.**
- Critères d'évaluation :
    - Respect des consignes (spécifications techniques et fonctionnelles, délais…),
    - Pertinence de la solution technique et architecturale implémentée (adaptation aux besoins fonctionnels et techniques, créativité, force de proposition…),
    - Qualité du code,
    - Comportement en cours (participation, assiduité, respect, efforts fournis,…).

Méthodologie

Mise en application d'une démarche d'analyse / conception inspirée de **Domain-Driven Design**.

Les activités détaillées ci-après, pourront être effectuées en suivant une approche itérative (cf. Méthodes Agiles).

Problem Space

Objectifs : découverte du domaine, recueil du besoin, élaboration de la solution, définition de l'architecture logicielle…

- Chaque groupe nommera une personne chargée de recueillir les spécifications fonctionnelles détaillées auprès de la Maîtrise d'Ouvrage (rôle joué par l'enseignant). Cette personne sera l'interlocuteur privilégié de la Maîtrise d'Ouvrage au cours de la réalisation du projet (suivi, points d'étape, proposition…).
- Recherche et documentation à propos du *Domaine Métier* (vocabulaire, logique métier, usages…).
- Identification du *Domaine* et des *Sous-Domaines* (*Core, Generic, Supporting*),
- Identification des fonctionnalités clés, des entités et leur relations, des acteurs et leur rôles,
- Mise en place d'un atelier d'*Event Storming* afin d'identifier les événements du *Domaine* (soit, les différents processus métier qui produisent des changements d'état dans le système).

Solution Space

Objectifs : implémentation de la solution logicielle définie à l'étape précédente.

- Sélection d'un *Bounded Context* parmi ceux identifiés lors de l'Étape 1, afin de mettre en place un template de micro service, réplicable pour les autres *Bounded Contexts*.
- Pour le *Bounded Context* sélectionné, mise en place de l'architecture de l'application en respectant les principes de découplage logiciel (ex: *Architecture Hexagonale*),

  Conception et implémentation des *Entities, Value Objects, Aggregates, Domain Events, Repositories…*
- Certaines fonctionnalités pourront être rendues sous forme de *"Preuve de Concept"* (en concertation avec l'enseignant).
- Sélection d'une ou plusieurs stacks technologiques.
- Conception d'une base de données par *Bounded Context*. Gestion de la réplication et du partage de données.
- Mise en place d'une *API* par *Service (Bounded Context)*.
- Mise en place d'une *API Gateway* en frontal de l'application back end.
- Mise en place d'une *Message Queue* avec *Rabbit MQ* pour la gestion des messages asynchrones entre services au sein de l'application.
- Chaque micro service devra fonctionner au sein d'un container *Docker* orchestré au sein d'un *Network* avec *Docker Compose*.
- Organisation de la collaboration entre services à travers différents protocoles de communication synchrones et asynchrones, tels que HTTP (*API REST*), *AMQP* (*Message Broker RabbitMQ*), *WebSocket*…
- Sécurisation des services.
- Monitoring des services.
- Mise en place de données de tests.

- Fourniture d'une documentation "interactive" des API
  (ex: Bruno, Postman…).
- Programmation de tests unitaires.
- **Les étudiants de la promo SID réaliseront un service de recommandation dans le cadre de leur cours avec Sylvain Castagnos. Cette fonctionnalité sera associée au projet sous la forme d'un micro service complémentaire.**