Rapport : Conception agile de projets informatiques

Université Lumière Lyon 2

ALFONSO OCAMPO Juan Diego

AUDI Karim

28/12/2023

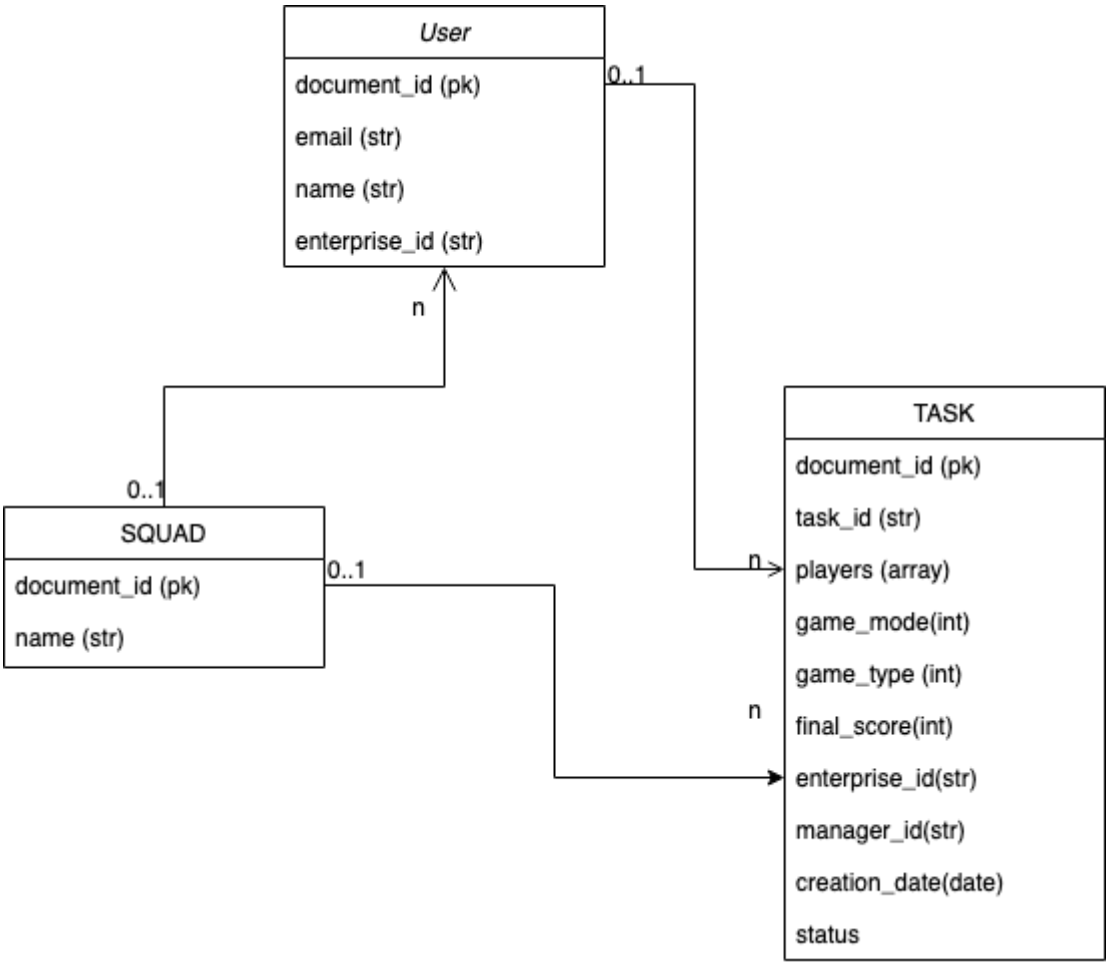
Documentation

Url Dépôt Git: https://github.com/jdalfons/ProjectAgileLyon/tree/main

Url Documentation: https://jdalfons.github.io/ProjectAgileLyon/moduleIndex.html

Url Projet déployée: https://planning-poker-jda-project-d76f7ea674c7.herokuapp.com/

UML



I. Cas d'utilisation :

Cas d'utilisation 1 : Conduite d'une session de Planning Poker (Boîte noire)

- **Portée :** Application web de Planning Poker.
- **Visibilité :** Boîte noire.
- **Niveau :** Utilisateur.
- **Déclencheur :** L'utilisateur choisit de lancer une nouvelle session de Planning Poker.
- **Acteurs :** Joueur (principal), autres joueurs (secondaires).
- **Description** : Permettre la participation des joueurs à une session de Planning Poker.
- **Garantie minimale :** Accès à l'application web de Planning Poker.
- **Critères de réussite :** Achèvement réussi de la session de Planning Poker avec qualification de la tâche.

Scénario nominal :

1. Le joueur se connecte à l'application web.
2. Le joueur démarre une nouvelle session de Planning Poker ou rejoint une session existante.
3. Le joueur créé ou rejoint une tâche
4. Le joueur attribue une note à la tâche sélectionnée.
5. Le joueur soumet sa note.
6. Le joueur attend la soumission des autres participants.
7. Le joueur examine les résultats des tâches auxquelles il a participé.

Extensions :

- **Alternative :** Les joueurs peuvent modifier leur note attribuée après la soumission, mais avant la conclusion de la session.

Cas d'utilisation 2 : Gestion des sessions de Planning Poker (Boîte Blanche)

- **Portée :** Application web de Planning Poker.
- **Visibilité** : Boîte blanche.
- **Niveau :** Application.
- **Déclencheur :** Un utilisateur se connecte sur l'application poker planning
- **Acteurs :**. Application, Utilisateur
- **Description :** Calculer et donner aux utilisateurs les qualifications des tâches et gérer efficacement les sessions de Planning Poker.
- **Garantie minimale :** Connexion Internet des utilisateurs et accès à l'application web de Planning Poker.
- **Critères de réussite :** Les Utilisateurs gère avec succès les sessions de Planning Poker.

Scénario nominal :

1. L'utilisateur se connecte à l'application web.
2. L'utilisateur accède à la page de gestion des sessions.
3. L'utilisateur peut effectuer les actions suivantes :
    a. Créer une nouvelle tâche.
    b. Qualifier une tâche existante.
    c. Fermer session
4. Une fois que tous les participants à une tâche ont évalué la tâche, celle-ci est calculée.
5. Si tout le monde choisit la même chose, le type de tâche est modifié à l'unanimité.

II. Design Patterns utilisés :

Singleton

- Objectif : S'assurer qu'une seule instance d'une classe est créée et fournir un point d'accès global pour y accéder.
- Explication : Implémenté pour contrôler une seule instance d'une connexion à la base de données, empêchant ainsi les connexions multiples. Cela est crucial pour gérer les performances, l'efficacité et éviter les problèmes liés aux connexions multiples. Il centralise le contrôle vers une seule instance globale, facilitant la gestion des erreurs et l'optimisation de la mémoire du système.

Decorator

- Objectif : Attacher dynamiquement des responsabilités supplémentaires à un objet.
- Explication : Les décorateurs Python seront utilisés pour améliorer les fonctionnalités de chaque fonction sans affecter les objets. Cela permet d'étendre les fonctionnalités sans réécrire tout le code.

Factory

- Objectif : Définir une interface pour la création d'un objet, mais laisser les sous-classes modifier le type d'objets qui seront créés.
- Explication : La classe `Task` est utilisée en tant que Factory, permettant des modifications et des créations de tâches dans la base de données. Cela offre une flexibilité dans la création de différents types de tâches.

Model-View-Controller (MVC)

- Objectif : Séparer une application en trois composants interconnectés : Modèle (données et logique métier), Vue (interface utilisateur) et Contrôleur (gestion des entrées utilisateur et mise à jour du modèle et de la vue).
- Explication : Utilisé pour contrôler l'application, où Firebase sert de modèle (base de données), le code HTML contrôlé par les modèles Jinja2 est la vue, et Flask agit comme un pont reliant la base de données et la vue, permettant le contrôle des fonctionnalités de l'application.

Gestion de Session

- Objectif : Gérer les informations de session, généralement dans les applications web.
- Explication : Flask contrôle complètement la gestion de session, la traitant comme un singleton. Il enregistre des informations pour chaque utilisateur entrant dans

l'application. Bien que les sessions soient actuellement stockées sur le serveur, l'intention est de les enregistrer dans une base de données à l'avenir.

Data Mapper

- Objectif : Maintenir l'indépendance entre les objets en mémoire et les entités de la base de données.
- Explication : Les valeurs sont d'abord intégrées en mémoire puis enregistrées ultérieurement dans la base de données, comme le montre la fonction `db_add_player_2_task`. Cela aide à séparer la représentation en mémoire de la structure de la base de données.

Post/Redirect/Get (PRG)

- Objectif : Gérer les soumissions de formulaire et éviter la re-soumission lors de l' actualisation de la page.
- Explication : Après toute modification ou mise à jour des données, une requête POST est effectuée. Ensuite, l'utilisateur est redirigé vers une autre page. Ce modèle évite les soumissions multiples et améliore l'expérience utilisateur.

III. Choix techniques :

Python

La réalisation du projet a été effectuée en utilisant les bibliothèques Jinja2 et Flask, choisies pour leur efficacité dans le développement rapide de projets web. Ces deux bibliothèques étant assez légères, simplifient la création d'applications web évolutives. De plus, l'utilisation de Python permet une organisation du code à la fois confortable pour les développeurs et propice à une compréhension aisée des structures sous-jacentes.

Bootstrap

Bootstrap a été choisi pour sa courbe d'apprentissage dans la construction d'interfaces web compatibles avec différents navigateurs. Sa vaste bibliothèque de composants et de styles pré-conçus facilite la conception rapide et réactive. Ce choix est particulièrement bénéfique pour garantir une expérience utilisateur visuellement attrayante et performante.

Firebase

Firebase a été sélectionné comme base de données en raison de sa solidité, l'implémentation des documents, les capacités de requêtage simples et la transformation transparente des résultats de requête en dictionnaires Python font de Firebase un outil idéal pour un jeu comme le Poker Planning avec une integration online. Son intégration avec le code simplifie

la gestion des données, ce qui est crucial pour gérer les tâches et les interactions dans un scénario de jeu de planification.

IV. Motivation

Pour la version 1, le projet a été conçu pour répondre au besoin d'un jeu Planning Poker pouvant être jouée à distance, Nous avons priorisée les fonctionnalitées du jeu de cette manière, d'après notre expérience de travail en entreprises qui travaillent sûr scrum, un fois que nous avons un premier livrable fonctionnel et qui permettre au développeurs et clients d'avoir un idée clair de la vision du projet, avec un méthodologie Agile permettre de mieux façonner le produit, c'est pourquoi nous avons choisi les fonctionnalitées ci-dessus pour ce première version.

- Fonctionnalité en ligne.
- Possibilité de télécharger un fichier JSON avec des tâches.
- Plusieurs tâches à travailler en même temps.
- Plusieurs sessions pour chaque utilisateur.
- Possibilité de se joindre à une tâche.

Advantages

L'idée de ce projet est que l'application ait la possibilité d'être trouvée via Internet afin que plusieurs utilisateurs, quel que soit le pays dans lequel ils se trouvent, puissent contribuer à la notation des tâches. Les utilisateurs pourront entrer, créer des tâches , les partager avec d'autres utilisateurs et les évaluer ensemble.

Défis

Comme défis, la connexion à plusieurs comptes à travers des sessions en plus de la qualification, en évitant le chevauchement dans chaque tâche qualifiée, en garantissant l'intégrité du vote et en atteignant le consensus des votes.

Utilisation

Sûr l'écran de sign in nous avons implémenté un méthode de registrer un utilisateur qui n'est pas enregistré sur l'application ainsi que de trouver un ancien session. les utilisateurs avec un ancien session déjà crée et vous pouvez utiliser sur différents navigateurs sont:

Admin

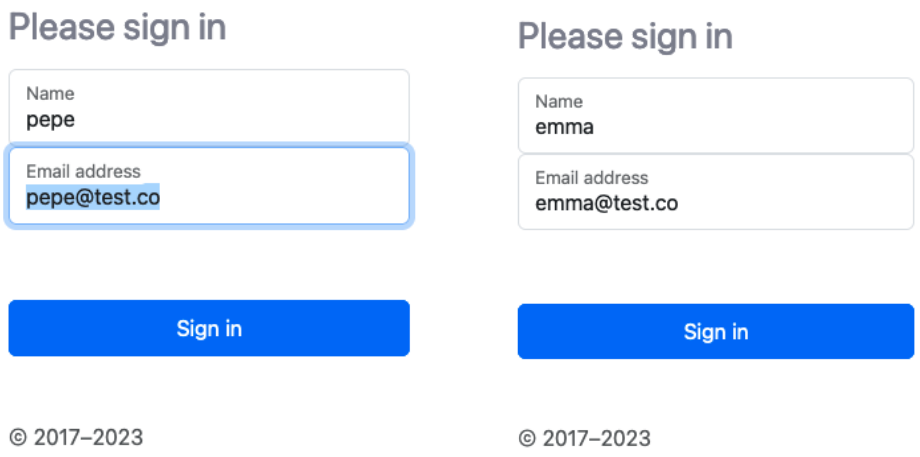
Nom: pepe

Email: pepe@test.co

Utilisateur

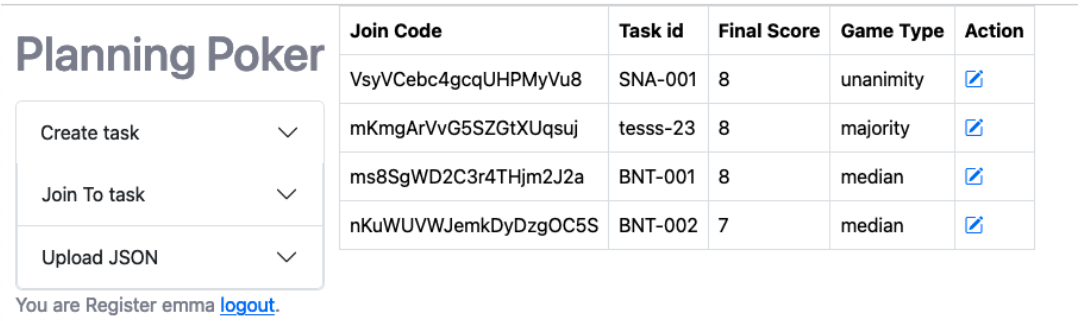
Nom: emma

Email: emma@test.co

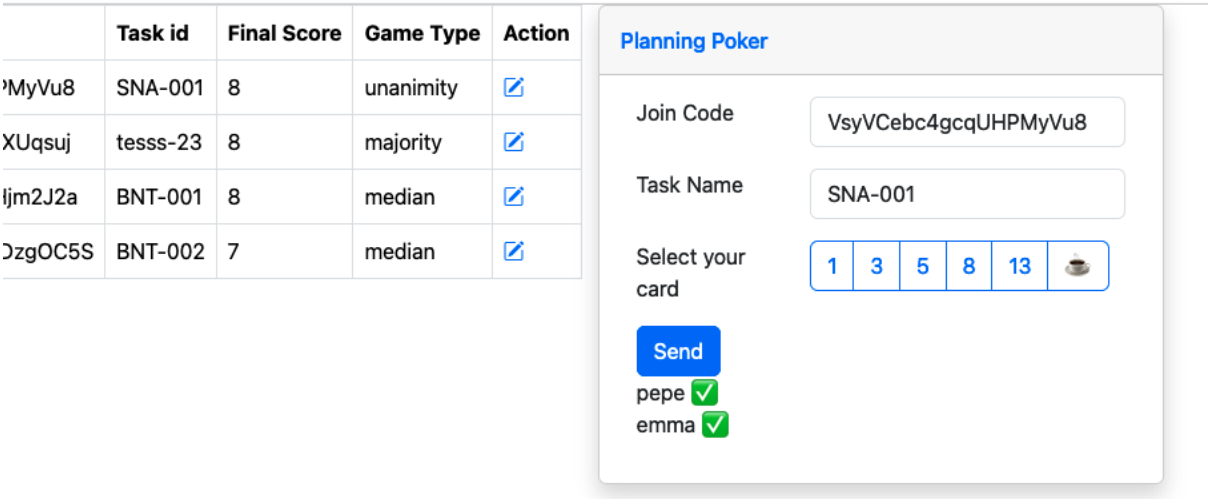


Sur l'écran d'accueil, vous verrez les tâches, au centre, à gauche, un menu qui vous permet de créer, rejoindre ou télécharger un fichier JSON avec des tâches.

Pour rejoindre des tâches, utilisez uniquement le code de jointure (Join code) qui apparaît dans la première colonne du tableau des tâches, et pour noter une tâche le bouton d'édition qui apparaît dans la dernière colonne, les autres colonnes font référence au nom de la tâche et au score final au cas où vous avez un score final.

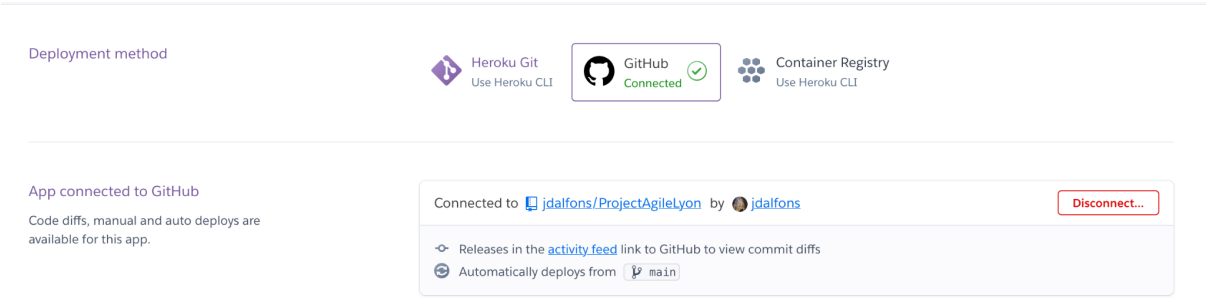


Finalement, dans la section de notation, nous aurons les informations sur la tâche, join code et nom (elle ne peut pas être modifiée) et le score que nous pouvons sélectionner et envoyer une seule fois (la tâche ne sera pas évaluée s'il n'y a pas plus de deux utilisateurs et pour à l'unanimité, tous les utilisateurs doivent avoir le même score et l'application changera le type de tâche).



V. Déploiement Continu

Pour cette première version nous avons choisi Heroku comme fournisseur cloud car il propose en déploiement continu une connexion directe au référentiel Github et à la branche à utiliser pour identifier les modifications et les extraire.



Sur chaque changement sur la branche **Main** Heroku sera capable de faire l'actualisation automatique du déploiement.

VI. Q&A

Comment est-il possible de calculer la note d'une tâche?

Les tâches seront calculées une fois que tous les scores seront envoyés et qu'il sera identifié que toutes les personnes qui ont rejoint la tâche envoient leurs scores pour la tâche.

Comment les utilisateurs s'inscrivent-ils? Pour cette première version, les utilisateurs s'inscrivent automatiquement lorsqu'ils identifient que l'email saisi n'existait pas auparavant.

Pour qui est pensé ce projet? Le projet a été pensé comme partie d'un projet interne d'une compagnie avec l'objectif de contrôler les tâches développées pour une entreprise sur chaque squad de développement.

Comment me connecter à la base des données? Le développement utilise Firebase ça veut dire que pour le bon fonctionnement est nécessaire suivre les instructions du README.md, les variables d'environnement seront envoyées par courrier.

Pourquoi il y existe l'Id-tâche et l'Id joint?: le projet à été pensé pour enregistrer le poker planning de chaque tâche avec un nom de nomenclature Jira ex DEV-0001 c'est pour cela que chaque tâche à un Id-tâche qui fait référence au nom de la tâche.

VII. Test

Les tests ont été configurés avec la bibliothèque Unittest. Des tests ont été écrits dans les classes Player et Task pour vérifier le comportement des entités principales de l'application. Ainsi, les

fonctionnalités liées à ces entités ont également été testées, le pre-commit est utilisé pour évaluer les tests, lisez README.md pour voir comment ils fonctionnent.

VIII. Version 1 de l'Application et Perspectives pour la Version 2

Le travail accompli du projet représente la version 1 de l'application. Celle-ci pouvant être amenée à évoluer de manière simple et efficace grâce au choix technique.
Voici une liste non exhaustive des fonctionnalités qui pourront être intégrées sur une version 2 de l'application suivant un développement agile itératif :

- Implémentation des changements de squad.
- Choix des quantité des cartes, types de jeu.
- Implementation d'interface responsive.
- Implémentation des contrôles au panés pour quelques actions.
- Edition des tâches (Id tâche) après la création.
- Elimination des tâches.
- Implémentation des plus types des cartes.