# **Mise en situation**

* En arrivant à votre bureau un matin, vous recevez le mail suivant de Sayf Bejaoui :

**A : vous**

**Sujet : FW : Construction de QA v0.9.1**

*Je viens de recevoir le rapport de QA pour la phase 1 du* [*projet*](https://1drv.ms/u/s!AmJGbSlW18YGsdcoJgUMHsqLyIEMqQ?e=JKfAa3)*. Il y a plusieurs bugs qui font planter la WebApp* (*maintenant de plus en plus populaire*) : *plateforme numérique pour coordonner les compétitions* [*deadlifting*](https://en.wikipedia.org/wiki/Deadlift)*,* [*strongman*](https://sport.gentside.com/musculation/strongman-les-athletes-les-plus-impressionnants-de-2020_art64158.html) *en Amérique du Nord et en Australie ! Malheureusement, je ne serai pas au bureau pour les prochains jours* (*un de mes enfants est tombé malade*)*. Voici le travail demandé* :

1. *Vous installeriez* [*pytest*](https://docs.pytest.org/en/6.2.x/)(*notre framework de test préféré*) *et* [*Locust*](https://locust.io/)(*tests de performance*)*. Je vous encourage à adopter une approche de TDD, car cela vous aidera à rationaliser votre travail.*
2. *J'ai déjà listé les bugs de la phase* 1 *que vous devriez régler et mettre en œuvre les éléments de la phase* 2 (*voir ci-dessus*). Très important : *Essayez de reproduire les bugs sur votre machine pour mieux les comprendre et ainsi plus facilement les résoudre.*
3. *Vous testeriez de manière approfondie les résultats requis pour toutes les fonctionnalités* (*de la WebApp*)*. A noter que pour éviter d'utiliser une DB, nous passons par un fichier JSON.*
4. *Vous prépareriez un rapport de test et un rapport de performances, conformément au* [*Guide de dév*](https://1drv.ms/w/s!AmJGbSlW18YGscZuleOcFb2LoA6OZg) *de l’entreprise car le QA nous reproche de ne pas respecter les normes.*

*Par ailleurs, je vous préviens : l'équipe commerciale n'était pas sûre du nombre de points à attribuer par concours, vous devrez donc certainement ajuster cela plus tard. Pour l'instant, il suffit de s'en tenir à 1 point = 1 concours. Je vais demander à Zayn, le Product Owner, de vous le confirmer* ([*email de Zayn*](https://1drv.ms/b/s!AmJGbSlW18YGscZxUNF2xKa2TkHlzg?e=SNCkR3) : *vous demandant de modifier plus tard cette fonctionnalité*)*.*

*Une fois que vous aurez terminé, nous ferons un examen de ce que vous avez dans la branche QA du code.*

*Nous examinerons les rapports et la manière dont vous avez résolu les problèmes, nous examinerons votre code et nous testerons la couverture de la nouvelle fonctionnalité.*

*Merci !*

***Sayf Bejaoui,***

# **Missions :**

##### Mise en place de l’env de test ?

**Pour vous aider :**

Problèmes et erreurs courants à éviter (liste non exhaustive) :

**Env de test** (*Install locale de l’app*) est **différent** del’**Env de Dév** (*GitHub*)**!**

* Ne pas passer par un env virtuel et du *requirements.txt* (*pour bien respecter les versions des pkgs*)
* Suite à l’install locale, le requirements.txt est à renouveler (pip freeze > requirements.txt) et à augmenter en ajoutant pytest

Oublier de mettre en place des **branches Git**.

Plutôt que de se lancer dans la recherche des erreurs du code et à les corriger, il est important de comprendre : *i*) le fonctionnement de l'application, *ii*) **d'écrire d'abord des tests**,*iii*) **à les passer au fur et à mesure de votre travail** (*en suivant une approche de TDD*). En d’autres termes, il n’y a pas d’« étape finale dite de test ». Cela vous permettra de ne pas prendre plus de temps que nécessaire ; Écrire des tests après avoir écrit du code est une stratégie perdante !

##### roote index :

##### Ecrire un test unitaire pour s’assurer que le serveur se lance correctement et que la page *index* s’affiche proprement ?

**Pour vous aider :**

* La webApp ne se lance pas parce qu’il manque à la fin du fichier *server.py* la condition **if \_\_name\_\_ == ‘\_\_main\_\_’: …** .
* Une autre méthode pour lancer l’app consiste à déclarer une var d’env **FLASK\_APP** (voir la [doc offic](https://flask.palletsprojects.com/en/2.0.x/quickstart/#a-minimal-application)).
* **Indice :** status\_code = 200, b’<form’ in response.data, b’<input’ in response.data

##### roote showSummary

##### Un test unitaire pour s’assurer qu’un Summary est affiché lorsqu'un *secretary email* (d’un *club*) est saisi dans le formulaire de la *page index* ?

##### Un test unitaire pour s'assurer qu’un message d’erreur est affiché lorsque le secrétaire du club saisi un email incorrect (par exemple : “Désolé, cet email n'a pas été trouvé”) ?

**Pour vous aider (les 2 dernières questions ensemble) :**

* La saisie d'un email inconnu entraîne le blocage de l'application (email qui ne se trouve pas dans le fichier *clubs.json*).
* Le bug se trouve à la ligne 29 du *server.py*.
* On s’intéresse aux fonctions***:*** loadClubs(), loadCompetitions(), index(), showSummary(). Vous devriez déboguer le code défectueux en mettant en place un message d’erreur lorsque le secrétaire du club saisit un email incorrect.

##### roote booking/competition/club

##### Un test pour s'assurer qu'il affiche une page *booking.html* pour un club ou une compétition valide ?

##### Un test pour s'assurer que l'interface utilisateur empêche la réservation d'un club ou d'une compétition non valide ?

##### roote PurshasePlaces

##### Un test pour s'assurer que lorsque l’User achète avec succès, les places sont correctement déduites du concours et les points déduits d'un club ?

##### Un test pour s'assurer qu'il affiche le message d'erreur souhaité lorsqu'il y a plus de 12 places ?

##### Un test pour s'assurer qu'il affiche le message d'erreur souhaité lorsque l'utilisateur s'engage à acheter plus de places que les points disponibles ?

**Pour vous aider (les 3 dernières questions ensemble) :**

* Il n'est pas vérifié que le nombre de places demandées est inférieur ou égal au nombre de places disponibles. En outre, les clubs ne devraient ni réserver plus de 12 places par compétition ni utiliser plus de points que ceux autorisés ! Enfin, les mises à jour des points ne sont pas prises en compte. Il y’a du code manquant sur les lignes 48-49 du server.py (bug à corriger).
* Corriger un bug (ligne 30 du serveur.py) : ‘’*Réservation de places dans les concours précédents*’’ ?! Les clubs ne peuvent utiliser les points que pour réserver des places pour des compétitions futures. La liste complète des concours est en effet passée dans le modèle. On pourrait filtrer uniquement les concours à venir dans le fichier welcome.html (ligne 27). Ajouter un conditionnel pour n'autoriser la réservation que si la date est dans le futur.

##### Mettre en œuvre la fonctionnalité

##### Livrable : Une page séparée qui comprends un tableau montrant la liste des clubs et les soldes de points respectifs actualisés. Aussi, les utilisateurs peuvent voir le tableau sans se connecter.

##### Modifier la valeur des points (selon le mail de Zayn) : L'échange actuel de 1 point = 1 place de compétition a été mis à jour de sorte que 3 points = 1 place de compétition.

##### Test unitaire pour logout ?

##### Tests d’intégration (fichier *server.py*) :

Un test qui exécute successivement les fonctions “loadClubs, loadCompetitions, index, showSummary, book, purchasePlaces, logout.”

##### Vérifier la couverture des tests. Livrable : rapport de couverture achevés.

##### Effectuer des tests de performance.

**Pour vous aider :**

Le rapport de performance Locust est **pertinent** quand :

* Le rapport de performance Locust indique le temps de rendu des fonctionnalités du site web.
* Le rapport indique ce qui suit :

Il ne faut pas plus de cinq secondes pour récupérer la liste des concours.

Il ne faut pas plus de deux secondes pour mettre à jour le total des points.

Il ne faut pas plus de deux secondes pour rendre la page du solde de points.

# **Livrables attendus et critères d’évaluation**

Un repo GitHub (ou équivalent) accessible au public contenant :

* Un fichier README
* Tous les tests sont contenus dans un dossier nommé de manière appropriée (par exemple, un dossier appelé tests).

Il existe un dossier séparé pour les tests unitaires et les tests d'intégration. Le rapport entre les tests unitaires et les tests fonctionnels doit être de 2:1.

* Le rapport de couverture montre que les tests couvrent au moins 60 % du code.
* des branches séparées pour chaque correctif et chaque fonctionnalité ; une branche d'assurance qualité qui combine tout cela.
* Le Code avec tous les bugs des onglets problèmes résolus, le code pour la nouvelle fonctionnalité et tous les tests réussis.
* Captures d'écran du rapport de tests (avant et après). Captures d'écran du rapport de performances Locust (avant et après).
* Un support de présentation à destination de votre manager (environ 10 slides) et une note méthodologique.

# **Modalités de présentation du travail**

Votre présentation pourra prendre cette forme :

|  |  |
| --- | --- |
| 5 min. | Rappel de la problématique et présentation du jeu de données |
| 15 min | Explication de l’approche pour mettre en place votre solution (méthodo, code, interprétation des résultats, etc ? ) |
| 10 min | Séance de questions-réponses |