Modèle rétrospectif

Sprint 1 – mediscreen-patient

Ce qui s’est bien passé :

- J'ai utilisé le modèle de spécification REST API pour créer une implémentation REST pour accéder aux patients.

- J'ai réussi à communiquer avec la base de données SQL.

- J'ai testé mon implémentation en utilisant des outils comme Postman ou des commandes cURL ainsi qu’avec des test d’intégration et unitaires.

- J'ai réussi à mettre en place l'infrastructure des microservices Docker conformément aux exigences.

- L'interface utilisateur a été créée, permettant la saisie et la modification des informations de base du patient.

- Les tests d’intégration lancent un conteneur docker contenant une base de donnée dédiée.

Ce qui aurait pu aller mieux :

- J'aurais pu mieux estimer le temps nécessaire pour terminer certaines tâches, ce qui aurait permis une meilleure planification et gestion du temps.

- Utiliser, pour la ressource Patient, des noms de variables plus adaptés, plus éxplicites. (given, dob → lastName
)

Ce que j’aimerais faire différemment :

- Établir des estimations de temps plus précises pour les prochaines itérations afin d'améliorer la planification des sprint.

- Revoir la structure des URI des points de terminaison. (Singulier/pluriel, pas de mot CRUD…)

Sprint 2 – mediscreen-notes

Ce qui s’est bien passé :

- J'ai utilisé le modèle de spécification REST API pour créer une implémentation REST pour accéder aux notes.

- J'ai réussi à communiquer avec la base de données NoSQL MongoDB.

- J'ai testé mon implémentation en utilisant des outils comme Postman ou des commandes cURL ainsi qu’avec des test d’intégration et unitaires.

- J'ai réussi à mettre en place l'infrastructure des microservices Docker conformément aux exigences.

- Les tests d’intégration lancent un conteneur docker contenant une base de donnée dédiée.

Ce qui aurait pu aller mieux :

- J'aurais pu mieux estimer le temps nécessaire pour terminer certaines tâches, ce qui aurait permis une meilleure planification et gestion du temps.

Ce que j’aimerais faire différemment :

- Établir des estimations de temps plus précises pour les prochaines itérations afin d'améliorer la planification des sprint.

- Faire le choix d’utiliser d’autres objets JAVA, plus récents et plus faciles à intégrer. (Date → LocalDate)

Sprint 3 – mediscreen-report

Ce qui s’est bien passé :

- J'ai utilisé le modèle de spécification REST API pour créer une implémentation REST pour générer un rapport sur le risque de diabète.

- J'ai testé mon implémentation en utilisant des outils comme Postman ou des commandes cURL ainsi qu’avec des test d’intégration et unitaires.

- J'ai mis en place l'infrastructure des microservices Docker conformément aux exigences.

- J’ai utilisé Docker Compose afin d’automatiser le déploiement de tous les conteneurs Docker du projet.

- J'ai mis en place les règles de détermination des niveaux de risque en fonction des données démographiques du patient et des déclencheurs présents dans les notes du praticien.

- L'interface utilisateur a été créée, permettant la génération et l’affichage du rapport de diabète d’un patient.

Ce qui aurait pu aller mieux :

- J'aurais pu mieux estimer le temps nécessaire pour terminer certaines tâches, ce qui aurait permis une meilleure planification et gestion du temps.

- Utilisation tardive de variables d’environnement pour pouvoir passer dynamiquement de la configuration docker a la configuration localhost.

- Le manque de précision quand aux règles de détermination des niveaux de risque m’a amené à prendre des décisions arbitraires. (30 ans c’est 30+ ou 30- , etc...)

Ce que j’aimerais faire différemment :

- Établir des estimations de temps plus précises pour les prochaines itérations afin d'améliorer la planification des sprint.

- J’aurais aimé avoir une approche différente concernant l’extraction des mots de la liste de notes. J’ai utilisé une REGEX, ce qui m’a paru plutôt complexe.