Projet final

Objectif

Vous devez concevoir et développer une application web et son back-end en appliquant les bonnes pratiques de lisibilité et de maintenabilité et en utilisant les principes SOLID, le DDD et l'architecture Hexagonale.

Vous serez évalués sur :

- le fiabilité de vos fonctionnalités
- le nombre de fonctionnalités réalisées
- la qualité de votre nommage et la lisibilité de votre code
- la qualité de vos fonctions
- la couverture de test et la pertinence de vos tests
- le respect des principes SOLID
- l'application du DDD pour vos entités métiers
- l'architecture de votre application
- la facilité d'extension de votre application (ex.: l'ajout d'un cas ou d'une fonctionnalité)

L'objectif principal est **la qualité de code**. Ainsi, un programme ne faisant que la moitié des fonctionnalités mais de très bonne qualité aura plus de points qu'un programme complet mais fait à la va-vite.

Groupe de 3.

Enoncé

Mise en situation

En tant qu'étudiant et futur professionnel de l'informatique, vous avez besoin d'être en apprentissage constant. Afin de gagner du temps et d'optimiser votre apprentissage, vous décidez de développer une application d'aide. L'application se basera sur le fonctionnement du système de Leitner en mettant en place les principes de "répétition espacée" et "d'auto-évaluation".

Comme vous êtes quelqu'un d'altruiste, vous pourrez également prévoir un système d'authentification pour en faire profiter vos camarades de promotion. Il sera donc attendu que vous **anticipiez la gestion des utilisateurs dans votre architecture (schéma et code)**. Néanmoins, il ne sera pas attendu que vous l'implémentiez.

Le fonctionnement du système de Leitner est le suivant :

1. Vous créez des fiches d'apprentissage composées de :
    - au recto : une question
    - au verso : la réponse
2. Vous positionnez ces fiches dans la catégorie 1
3. Chaque jour vous parcourez les fiches de la catégorie 1 :
    - les fiches répondues avec succès passent en catégorie 2
    - les fiches non ou mal répondues restent en catégorie 1
4. Tous les 2 jours vous parcourez les fiches de la catégorie 2 :
    - les fiches répondues avec succès passent en catégorie 3
    - les fiches non ou mal répondues restent en catégorie 1
5. Tous les 4 jours vous parcourez les fiches de la catégorie 3 :
    - les fiches répondues avec succès passent en catégorie 4
    - les fiches non ou mal répondues restent en catégorie 1
6. etc…

La répétition se fait sur 7 catégories avec un délai entre les catégories doublé à chaque fois :

Catégorie	1	2	3	4	5	6	7
Fréquence (jours)	1	2	4	8	16	32	64

A chaque mauvaise réponse, la fiche revient en catégorie 1. Si une fiche atteint la catégorie 7 et qu'elle est correctement répondue, elle sort du système et peut être considérée comme apprise définitivement.

Expression de besoin

Rappel : Prévoyez la gestion des utilisateurs dans votre architecture mais **vous pouvez faire l'impasse**. Pour cette gestion utilisateur vous ne serez évalué que sur l'aspect architectural, aucune implémentation n'est attendue. Une fois votre architecture designée (schéma + code), **vous pouvez considérez l'expression de besoin pour des utilisateurs non connectés systématiquement**.

- En tant qu'utilisateur non connecté, je souhaite pouvoir me connecter pour accéder à mes propres fiches
- En tant qu'utilisateur connecté, je souhaite pouvoir créer des fiches qui seront intégrées dans le système en catégorie 1
- En tant qu'utilisateur connecté, je souhaite pouvoir déclencher un questionnaire afin de répondre aux questions de mes fiches
- En tant qu'utilisateur connecté, je souhaite ne pouvoir faire qu'un questionnaire par jour
- En tant qu'utilisateur connecté, je souhaite pouvoir comparer une mauvaise réponse avec la réponse d'origine
- En tant qu'utilisateur connecté, je souhaite pouvoir forcer la validation d'une fiche même si ma réponse n'est pas la même afin de ne pas être impacté par une différence de formulation
- En tant qu'utilisateur connecté, je souhaite que les fiches mal répondues reviennent en catégorie 1
- En tant qu'utilisateur connecté, je souhaite que les fiches correctement répondues passent dans la catégorie supérieure
- En tant qu'utilisateur connecté, je souhaite que les fiches de catégorie 7 auquel je réponds correctement ne me soient plus proposées à l'avenir
- En tant qu'utilisateur connecté, je souhaite que les fiches qui me sont proposées correspondent à la fréquence associée à leur catégorie selon le système de Leitner
- En tant qu'utilisateur connecté, je souhaite pouvoir mettre des tags personnalisés sur mes fiches
- En tant qu'utilisateur connecté, je souhaite pouvoir consulter toutes les fiches associées à un même tag et leur catégorie
- En tant qu'utilisateur connecté, je souhaite être notifié à l'heure de mon choix pour déclencher le questionnaire

Légende :

- *besoin : expression de besoin sur laquelle seul le design architectural (schéma + code) est attendu. Aucune implémentation fonctionnelle n'est nécessaire.*

Attendu

D’abord, vous devrez mettre en évidence votre conception et l’architecture de votre application par le biais de schéma(s). Le ou les schémas doivent représenter le front et le back, leur modèle respectif et le découpage en couche de l’architecture Hexagonale. Pensez à représenter et expliciter les responsabilités de chaque brique technique et les interactions entre ces briques. Aussi, prenez le temps d’expliquer et justifier vos choix pour comprendre votre démarche.

Ensuite, vous devrez fournir le code de votre application avec un *README.md* permettant d’installer, démarrer votre application et exécuter les tests avec le calcul de la couverture. Les tests devront être développés pour le front comme pour le back.

En ce qui concerne le fonctionnement de votre application, votre front sera testé en étant connecté avec le back de l’examinateur. De même, votre back sera testé en étant connecté avec le front de l’examinateur. Ainsi, pour que les deux puissent communiquer, **votre API doit suivre le contrat décrit dans le Swagger File fourni avec l'énoncé.** Pour facilement lire et interagir avec le Swagger, vous pouvez en copier le contenu dans le Swagger editor : https://editor.swagger.io/. **Une incapacité à démarrer votre application, à la connecter aux services de l’examinateur ou un non-respect de l’API documentée dans le Swagger comptera pour 0 sur la note de fonctionnement.**

En ce qui concerne le stockage de données, **vous pouvez faire l’impasse et écrire en dur les informations**. Si vous faites ce choix, pensez bien à quand même créer une architecture simulant la lecture en base, par exemple pour la création de nouvelles fiches.

En ce qui concerne l’interface graphique, vous pouvez faire l’impasse. Cependant, une exigence plus forte sera appliquée sur la qualité de votre code back. Autrement, **vous pouvez ne pas vous préoccuper du design graphique tant que votre application est utilisable**.

Bonus 1 (1 points)

L'intérêt majeur du système de Leitner est la répétition espacée. Des études ont prouvées que la rétention d'information est significativement plus élevée lorsqu'elle a lieu au moment où l'oubli commence. Ainsi, le délai augmente entre chaque répétition pour avoir lieu au moment où la personne commence à oublier.

Le système de Leitner tel que décrit plus haut se base sur un calendrier de fréquence pour chaque catégorie :

Jour	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Cat. 1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Cat. 2		x		x		x		x		x		x		x		x	
Cat. 3				x				x				x				x	
Cat. 4								x								x	
Cat. 5																x	

Cela pose un problème de fréquence dans certains cas car le délai n'est pas respecté entre chaque catégorie.

Dans le tableau précédent, les cases en rouge représente une question qui est correctement répondue dès le premier jour. On obtient les délais suivant

# de question	1	2	3	4	5
Catégorie	1	2	3	4	5
Délai depuis dernière question (jours) avec système calendaire	0	1	2	4	8
Délai depuis dernière question (jours) théorique	0	2	4	8	16

Pour respecter le délai théorique, il faut donc se baser sur la date de dernière réponse. C'est-à-dire qu'au moment où la question rentre en catégorie 5, il faut attendre 16 jours avant qu'elle soit reposée.

Modifiez votre code pour se baser sur la date de réponse et non sur un système calendaire.

Suggestion : vous pouvez faire une branche à part bonus-1 pour ne pas impacter le reste de votre application.

Bonus 2 (jusqu’à 2 points)

Développez le scénario de test end-to-end suivant :

- En tant qu’utilisateur connecté, je souhaite pouvoir créer des fiches qui seront intégrées dans le système en catégorie 1

Afin de respecter les bonnes pratiques, il est recommandé de s’appuyer sur le formalisme Gherkin afin de reformuler le scénario et découper les étapes de votre test.

Inspirez vous de la littérature et des exemples que vous trouverez pour essayer d’avoir un test clair, lisible et explicitement les actions utilisateurs.

Le choix de la technologie est libre. Néanmoins, utilisez une technologie où vous implémenterez votre propre test. L’utilisation d'outils permettant d’enregistrer et rejouer une série de clics ne permettra pas de valider ce bonus.

Enfin, pensez à compléter votre *README.md* pour indiquer la procédure pour installer et exécuter votre test end-to-end.

Suggestion : vous pouvez faire une branche à part bonus-2 pour ne pas impacter le reste de votre application

Suggestion : vous pouvez utiliser Playwright qui est un des leaders du test end-to-end. Cette librairy est light-weight, multi-langage et vous offre de grandes possibilités de personnalisation.