MPS – MANDAT DE PROJET

Faculté des sciences – Département d'informatique

NOM DE L'ÉQUIPE DE PROJET : WS

CONTEXTE

Bien qu'il existe beaucoup de sites web de vente de billet sur internet, il peut être difficile de se trouver un billet bien placé dans la salle lorsque le volume de demande est important. Cela est dû au fait que certains utilisateurs bien préparés sont toujours en attente d'évènements et achètent les billets en masse et le plus tôt possible. Il s'agit, entre autres, des scalpeurs.

OBJECTIFS

Donner les objectifs du projet en se rappelant qu'un objectif doit être mesurable (il y a moyen de savoir si l'objectif a été atteint à la fin du projet)

- Avoir des pipelines CI/CD pour vérifier nos Build
- Chaque fonctionnalité doit avoir ses tests unitaires
- Utiliser la méthodologie Agile SCRUM
- Utiliser des lints pour le clean code (vérification de syntaxe)

LIVRABLES

Préciser la liste des livrables à la fin du projet :

manuel d'utilisateur, manuel technique, système, application, article scientifique, preuve de concept, prototype, etc

- Envois de notifications pour les événements réservés pour les membres premium
- Authentification des utilisateurs
- Système de compte premium
- Liste de recommandations
- Liste d'événements
- Choix des places
- Ajout de préférences
- Ajout de billets, salles, événements, etc.
- Recevoir les billets par courriel
- Page de profil
- Acheter des billets
- Interface d'administration
- Gestion des annulations et remboursements

MÉTHODOLOGIES

NOM DE L'ÉQUIPE DE PROJET : WS

Préciser les technologies qui seront utilisées pour réaliser le projet
Exemple : diagramme de classe, java, C++, Framework Kohana, …

Backend : .NET avec Entity Framework
Frontend : Angular
Base de données : SQL
Méthodologie : Agile SCRUM
Gestion des sources : GitLab
Système d'exploitation : Windows
IDE : Visual Studio pour backend, IntelliJ ou Visual Studio Code pour le frontend
Communications : Teams

HYPOTHÈSES

Préciser les hypothèses sur lesquelles le projet repose :
concept, prototype, technologies, tendances, marché,….

- Il n'y aura pas une quantité excessive d'utilisateurs premium. Si c'est le cas, pour certains évènements en forte demande, s'il y a une grande quantité de membres ayant les mêmes préférences, il sera impossible de les respecter pour tous les utilisateurs.
- Il est aussi assumé que les utilisateurs seront prêts à payer le coût d'un abonnement premium afin d'augmenter leurs chances d'avoir des bons billets.
- On sera capable de se procurer une quantité de billets assez importante pour satisfaire tous nos utilisateurs et ces billets proviennent de diffuseurs ou de promoteurs évènementiels.
- L'API Spotify permettra d'envoyer assez de requêtes pour tous nos utilisateurs

EXIGENCES

À préciser :

1. Une liste de caractéristiques que les biens livrables doivent satisfaire

   - Exigences fonctionnelles : quelles fonction le système doit accomplir
   - Exigences non-fonctionnelles (comment le système doit accomplir ces fonctions (ex : temps, maintenabilité, langage de programmation, librairies à utiliser, etc.)),

2. Une liste de caractéristiques que le processus du projet doit satisfaire

   - Utiliser la méthode de développement XYX
   - Utiliser les techniques suivantes X, Y, Z
   - Etc.

Exigences fonctionnelles :

- Authentification des utilisateurs
- Avoir une liste d'événements

- Avoir des recommandations d'événements
- Avoir des notifications pour les billets réservés aux évènements intéressants
- Pouvoir acheter des billets
- Pouvoir devenir membre premium
- Pouvoir choisir ses sièges
- Pouvoir configurer des préférences de places, budget, etc.
- Recevoir les billets par courriel
- Liste de billets achetés (page de profil)
- Interface d'administration
- Gestion des annulations et remboursements
- Gestion des codes promotionnels ou réductions

Exigences non-fonctionnelles :

- Authentification sécuriatire
- Respect des normes PCI DSS pour les paiements
- Utilisation de HTTPS
- Capable de gérer au minimum 15 requêtes simultanées
- Rapidité raisonnable
- .NET Entity Framework (C#) pour le backend
- Angular (TypeScript) pour le frontend
- SQL pour la base de données
- Fonctionne au minimum sur Chrome et Firefox
- Utilisable au minimum sur desktop
- Utilisation de l'API Spotify
- Respect des normes du clean code
- Utilisation des tests unitaires
- Répond aux normes OWASP

Liste de caractéristiques :

- Branche master pour la production
- Branche develop pour les tests et le développement
- Branches commençant par feature/[id-ticket]-[Description courte] pour les fonctionnalités
- Impossible de push dans master directement
- Utiliser des branches de fonctionnalité dès que possible
- Toujours pull la branche master avant de créer une feature branch
- Supprimer les feature branch lorsque la PR est acceptée en gardant l'historique des commits sur la branche de destination
- Chaque PR doit être revue par un autre membre que son auteur
- Utiliser Agile SCRUM
- Kanban colonnes :
    - Open : Fonctionnalités prêtent à être développer
    - In progress : Fonctionnalités en cours de développement

    - To verify : Fonctionnalités en train d'être testé
    - Done : Fonctionnalités terminées
- Définition de terminé :
    - Un code review/pair-programming a été effectué
    - La fonctionnalité a au minimum deux tests unitaires (si possible plus que 2)
    - La fonctionnalité a été QA par l'autre, ensemble si possible
    - La fonctionnalité répond aux critères d'acceptation

INCLUSIONS

Préciser ce que la solution doit nécessairement inclure

- Vente de billets pour les événements
- Paiement par carte de crédit Visa et Mastercard
- Authentification des utilisateurs
- Liste d'événements
- Envois des billets
- Interface d'administration pour ajouter des événements
- Support français et anglais

EXCLUSIONS

Préciser ce qui ne sera pas inclut dans la solution et qui est acceptable pour le demandeur.

- Paiement débit, carte cadeaux et PayPal
- Revente de billets
- Intégration avec des billetteries externes comme Ticketmaster
- Application mobile
- Système de fidelités comme Scène+, Air Miles, etc.

NOM DE L'ÉQUIPE DE PROJET : WS

CRITÈRES D'ÉVALUATION DU PROF

Préciser avec le prof les critères qu'il utilisera pour évaluer votre solution finale et le poids de chaque critère :

		Préciser le poids des critères d'évaluation
Outils et techniques utilisés		
	Programmation	
	Architecture	
Respect des exigences		
	Exigences utilisateur	
	Exigences fonctionnelles	
	Exigences non fonctionnelles	
	Exigences de projet	
Qualité des livrables		
Évolutivité	Adaptable	
	Maintenable	
	Testable	
Exploitation	Robustesse	
	Efficacité	
	Intégrité	
Transport	Portable	
	Interoperable	
	Réutilisable	
Facteur techno.	Productif	
	Interface	
BONUS (PROJET INNOVANT)		

Il faut déposer le mandat approuvé par le prof sur l'outil de gestion de projet, section Document et demander à celui-ci d'approuver le document en venant verrouiller le document.