Plan de tests

Équipe Dédale (Kigmou Bilal, Williams Simon, Choukroun–Balzan Lilou, Couton Alexandre, Magzoumov Nikita, Fletschinger Valentin)

1. Quelle est l'application que vous testez ? La décrire.

Dédale est un projet divisé en une application mobile et une application lourde qui peuvent s'échanger des données.

L'application mobile permet aux utilisateurs d'ajouter des points d'intérêts sur une carte de la ville de Strasbourg. Ces points peuvent être mis à l'emplacement actuel de l'utilisateur ou à un autre endroit sur la carte. Quand l'utilisateur enregistre ses points il peut y ajouter d'autres informations : commentaires, photos, et les types d'infrastructures à y placer et leurs nombre. L'utilisateur peut ensuite accéder à une liste de tous ses points enregistrés, il peut les modifier, les supprimer ou leur ajouter des informations supplémentaires. À l'aide de ces points il peut ensuite créer un chemin de route dans l'ordre qu'il veut et l'application lui permet de suivre ce chemin et de passer automatiquement au précédent une fois l'actuel atteint.

L'application lourde permet aux utilisateurs de centraliser les données, visualiser de manière globale les informations sur les événements ainsi que gérer les équipes et les plannings. En effet, sur cette dernière, l'utilisateur à un visuel sur les informations d'un événement à travers une carte sur laquelle il peut créer, modifier et supprimer des points, zones, parcours, points d'intérêts et équipements. L'utilisateur peut également gérer son personnel et ses équipes ainsi qu'exporter des données au format excel et pdf.

Pour assurer la communication de la base de données entre l'application lourde sur ordinateur et l'application mobile, un QR code créé par l'application lourde permet, une fois scannée par l'application mobile, l'envoi d'informations sélectionnées au préalable entre les deux plateformes.

2. Quelles fonctionnalités seront testées et en quelle mesure ? Pourquoi ?

Les fonctionnalités sont classées par priorité afin d'optimiser la couverture de test.

Fonctionnalités critiques (priorité haute) :

Application desktop

- Ajout de points
- Modification de points
- Suppression de points
- Réception des données depuis le mobile
- Gestion de la base de données

Application mobile

- Ajout de points
- Modification de points
- Suppression de points
- Création d'un itinéraire

Ces fonctionnalités sont critiques car elles conditionnent l'intégrité des données et la communication entre les deux applications.

Fonctionnalités importantes (priorité moyenne) :

Application desktop

- Recherche d'un lieu
- Placement et modification des obstacles
- Affichage des listes et détails de points

Application mobile

- Ajout de photos
- Ajout de commentaires
- Ajout d'obstacles

Ces fonctionnalités sont importantes car elles jouent un rôle primordial dans l'expérience utilisateur et l'intérêt de celui-ci à utiliser notre application.

Fonctionnalités secondaires (priorité basse) :

- Export PDF / Excel
- Affichage avancé des statistiques
- Gestion avancée des événements

Ces fonctionnalités sont secondaires car elles constituent des ajouts de fonctionnalités individuelles qui ajoutent un aspect pratique à notre application.

3. Quelles fonctionnalités ne seront pas testées ? Pourquoi ?

Les fonctionnalités suivantes ne sont pas testées dans ce plan :

- Interface graphique détaillée (ergonomie, UX, design)
- Performances graphiques de la carte
- Tests de charge et de montée en charge
- Sécurité avancée (attaques, injections, pentests)
- Compatibilité multi-appareils et multi-résolutions

Raison :

- Manque de temps
- Priorité donnée aux fonctionnalités cœur
- Peu de pertinence au vu de la demande

4. Quels risques votre plan de test comporte-t-il ?

- Les tests étant majoritairement unitaires, les interactions entre modules ne sont pas validées
- L'absence de tests d'intégration peut masquer des erreurs lors de l'assemblage des composants
- Une régression fonctionnelle peut passer inaperçue si elle n'impacte pas directement une fonction unitaire
- La validation du bon fonctionnement global repose sur des tests manuels

5. Quels outils de tests sont utilisés ?

- Le framework de tests natif de Rust
- Utilisation de l'annotation `#[test]`
- Tests unitaires intégrés directement au code source (dossier tests)
- Exécution des tests via `cargo test`
- Vérification des résultats par assertions Rust (`assert!`, `assert_eq!`)

Ces outils permettent une validation rapide et automatisée des fonctions techniques du backend, mais ne couvrent pas les scénarios fonctionnels complets.

6. Quel est le planning de la mise en place de vos tests ? Qui, quoi, quand ?

Pour la V0 (premier sprint) : Aucun test mis en place afin de donner la priorité au développement pur de fonctionnalité au vu du délais imposé

Pour la V1 (deuxième sprint) : 1 personne commence à mettre en place des tests afin de commencer à assurer la qualité de notre code ainsi qu'éviter la régression.

Pour la V2+ (reste des sprints) : Les tests sont réalisés par les différents membres de l'équipe sur les fonctionnalités qui lui ont été confiées avant la fin de l'échéance du sprint.