Documentation Technique

Galaxy Task

Résumé

Ce manuel constitue une ressource incontournable pour les développeurs de Galaxy Task, qu'ils débutent ou possèdent une expérience au sein de notre collectif. Il vise à offrir une perspective limpide et exhaustive sur les technologies de pointe déployées durant le développement du projet.

À l'intérieur, les développeurs découvriront des renseignements essentiels qui les orienteront à travers les étapes variées du développement. Ce document présente un aperçu précis, soulignant l'architecture intégrale du projet ainsi que les subtilités techniques qui le composent. Des explications sur les outils et bibliothèques essentiels sont fournies, avec des conseils pour leur exploitation efficace.

La partie du guide dédiée à la communication interne est également fondamentale. Saisir la manière dont l'équipe Galaxy Task échange, via des moyens formels ou informels, est crucial pour assurer une coopération harmonieuse et fructueuse. Ce guide est conçu pour être un soutien robuste pour nos développeurs, facilitant leur intégration et leur participation active à la progression de Galaxy Task.

Description du document

Titre	Documentation technique
Objet	Explication du fonctionnement technique de la solution a un utilisateur voulant contribuer au projet.
Auteur	Groupe DevBro's
Chef de projet	Tristan Mey
Courriel	Tristan.mey@epitech.eu
Promotion	2026
Date de mise à jour	3/11/2024
Version du modèle	1.0

Tableau des révisions

Date	Version	Auteur	Section(s)	Commentaires
3/11/2024	1.0	Groupe DevBro's	Toutes	Première version

Contact

Si vous avez des interrogations ou si vous avez besoin d'aide, n'hésitez pas à nous envoyer un courriel directement à contact@devbros.com pour toute assistance.

Auteurs

Tristan Mey
Léo Stephan
Matthieu Roess

Présentation du projet

Galaxy Task est un projet ayant été donné par EPITECH Strasbourg, développé sur une durée de 4 semaines.

Galaxy Task est une application mobile qui a été spécialement conçue pour répondre aux besoins des utilisateurs du site web Trello. Sa principale fonction est de faciliter la gestion de ses espaces de travail à partir de son téléphone de manière intuitive et efficace.

Galaxy Task prend en considération les contraintes courantes rencontrées lorsqu'il s'agit d'utiliser un site web initialement conçu pour un usage sur ordinateur. Effectivement, l'interface de notre application est optimisée pour tous les téléphones, en mettant l'accent sur l'ergonomie et la performance. Galaxy Task a pour objectif d'optimiser et de rendre votre utilisation de Trello plus performante.

General

Structure du projet

L'équipe du projet Galaxy Task est composée de 3 personnes :

- Tristan Mey, développeur frontend & backend,
- Matthieu Roess, développeur frontend & backend
- Léo Stephan, développeur backend & DevOps

Notre équipe de développement fonctionne avec l'organisme de gestion SCRUM, en effet chaque jour à 9h45 nous effectuons une réunion sur Teams basé sur l'exemple du Daily SCRUM pour discuter de nos avancés, problèmes et tâches à venir, ce qui nous permet de garder une vision claire de nos objectifs.

Outils utilisés

Communication

Discord
Discord est l'outil principal de communication au sein de notre équipe de développement.

Nous avons créé notre propre serveur Discord pour faciliter la communication au sein de notre équipe de développement. Ce serveur comporte différents channels de discussion, tels que :

- général : pour partager des informations avec toute l'équipe
- documents: pour stocker et accéder aux documents liés au projet.
- ressources : pour partager des ressources utiles, comme des outils ou des références.

Teams
Nous utilisons également **Microsoft Teams** pour effectuer les réunions entre nos membres.

Contribution

Github

Github est la plateforme essentielle où le code source de notre projet est hébergé. Chaque push initie une pull request, qui doit être validée par les membres du même pôle pour garantir la fonctionnalité et la lisibilité du code.

Nous utilisons également Github Project, intégré à Github, pour suivre les tâches au sein des différents pôles.

Bonnes pratiques Git
Pour maintenir une cohérence dans nos projets, nous avons établi des règles spécifiques pour l'utilisation de Git

- En bref
    - Chaque collaborateur doit créer une branche depuis develop nommées : dev/<prénom>.
    - Par la suite chaque branches features doivent partir de la branche personnelle du collaborateur et doivent être nommés : feature/\<numéro de l'issue\>_\<description\>.

- Les Branches
    - master :
      La branche master représente la dernière version packagée et livrée au client ou déployée. Elle ne doit avancer que via des merges de branches release/* ou hotfix/.

    - develop :
      La branche develop représente la dernière version de l'application, avec toutes fonctionnalités validées. Elle ne doit avancer que via des commits de merge réalisés via des PR de Github. Tout le code qu'elle contient est donc testé et validé. Elle doit toujours compiler.

    - feature/** :
      Les branches feature/... sont les branches contenant le code associé à une fonctionnalité.
      Elles doivent respecter la nomenclature : feature/\<numéro de l'issue\>_\<description\>
      - numéro de l'issue : La référence à l'issue de la fonctionnalité qu'on implémente.
      - description : Une courte description.
      Le code présent sur cette branche doit toujours être à jour entre local et remote. Ceci afin de limiter les pertes de code, en cas de destruction du PC.

En conséquence, il n'est pas obligatoire que la branche ne build pas, tant qu'aucune PR n'a été faite. Il est possible de merge develop. Il est strictement interdit de merge une autre branche feature/.

- Commits

Pour les messages de commit, il est important de suivre la norme de conventional commits dont les règles sont expliquées ici :
https://www.conventionalcommits.org/en/v1.0.0/

- Quelques règles en plus

Ces règles vont permettre de conserver un historique facile à lire, où les commits vont pouvoir documenter l'évolution du projet.
Il ne faut jamais faire de rebase des branches partagées (develop, branches feature en commun).

  - Pendant le développement :
    Dans une branche de développement personnelle (non partagée) on préfèrera utiliser rebase à merge pour se mettre à jour avec develop. On garde ainsi une historique linéaire de ses commits. Push sa branche ne pose pas de problèmes.

  - Au moment de faire une Pull Request :
    Avant de faire une PR, on peut nettoyer sa branche : squash (grouper) les commits, enlever les WIP, écrire des commentaires de commit pertinents pour ne conserver que des commits pertinents aux relecteurs. Les corrections effectuées après relecture sont faire sous forme de nouveaux commits (pas de rebase à ce stade !)

  - Au moment de merge
    Au moment de merger sur develop on pourra squash les commits pour garder la branche develop propre, et ne conserver que des commits pertinents au suivi du projet (quelles features / correctifs contient develop à une date donnée ?).
    Cela facilite également les reverts si nécessaires.

Application

Répertoire de travail

Le projet est hébergé sur GitHub au sein de l'équipe DevBro's, qui est une équipe privée. Pour toute collaboration ou demande d'accès, il faut se référer aux administrateurs de l'équipe Github qui sont les membres créateurs du projet.

Technologies et librairies utilisées

Node (version : >=21.11.1) : Node permet aux développeurs d'écrire du code JavaScript qui s'exécute directement dans le processus informatique lui-même, et non dans un navigateur. Node peut donc être utilisé pour écrire des applications côté serveur avec un accès au système d'exploitation, au système de fichiers, et à tout ce qui est nécessaire pour construire des applications pleinement fonctionnelles.

React-native (version : >=0.73.4) : Node permet aux développeurs d'écrire du code JavaScript qui s'exécute directement dans le processus informatique lui-même, et non dans un navigateur. Node peut donc être utilisé pour écrire des applications côté serveur avec un accès au système d'exploitation, au système de fichiers, et à tout ce qui est nécessaire pour construire des applications pleinement fonctionnelles.

Expo (version : >=50.0.11) : Expo est une plateforme open-source pour créer des applications universelles pour Android, iOS et le web avec JavaScript et React. Elle offre un ensemble d'outils et de services qui simplifient le développement et le déploiement d'applications React Native. Avec Expo, les développeurs peuvent accéder à une large gamme de fonctionnalités natives du téléphone sans avoir à écrire du code spécifique à chaque plateforme.

Jest (version : >=29.3.1) : Jest est un framework de test JavaScript populaire qui est utilisé pour tester des applications écrites en TypeScript, NodeJS, React, Angular, Vue, et bien d'autres.

Environnement de Travail

Visual Studio Code : est un éditeur de code simplifié qui prend en charge les opérations de développement telles que le débogage, l'exécution de tâches et le contrôle de version. Il offre les outils essentiels pour un cycle de développement rapide, en laissant les flux de travail plus complexes aux IDE plus complets.

WebStorm : est un IDE spécialement conçu pour le développement en JavaScript et ses technologies, y compris React, Node.js, HTML et CSS. Il offre des fonctionnalités avancées de codage et de débogage.

Postman ou Insomnia : Selon les préférences du développeur, Postman ou Insomnia peut être utilisé comme client API pour créer, partager, tester et documenter facilement les API.

Extensions : Les extensions Prettier et Git Graph sont demandé pour pouvoir assurer l'application des normes Airbnb et la lisibilité des commits.

Architecture

Dans le souci de maintenir une organisation claire et une évolution fluide de notre projet, nous avons soigneusement défini une architecture divisée en cinq grandes parties dès le début. Cette structure, détaillée ci-dessous, assure la cohérence et la maintenabilité du code, favorisant une collaboration efficace au sein de l'équipe.