SOKOBAN

Nous avons conçu trois solutions pour mener à bien ce projet.

DARLOUB
GAMES

I/ API en NodeJS

Lien API : https://sokoban-api.onrender.com

Justification de la technologie :
Le choix de NodeJS a été fait dans le but de conserver le langage JavaScript pour l'ensemble du projet. De plus, cela a permis à Théo de mettre en pratique ses acquis en entreprise. Paul, voulez essayer de gagner en compétence dans cette technologie.

Justification des dépendances :

express : Pour la création du serveur HTTP.
sequelize : Pour la gestion de la base de données SQL. (ORM)
jsonwebtoken : Pour la création et vérification des jetons JWT, nécessaire pour l'authentification.
bcrypt : Pour le hachage des mots de passe, afin de ne pas les stocker en clair dans notre base de données.
cors : Pour gérer les requêtes Cross-Origin, nécessaire pour que notre API puisse répondre aux requêtes de nos deux autres projets.
swagger-jsdoc et swagger-ui-express : Pour la création de la documentation de l'API de manière automatique et lisible.

Infos complémentaires :

l'endpoint "/boards" : renvoie une liste de plateau de jeu ordonnée par difficultée puis par date de création

l'endpoint "/boards/ : id" : renvoie les infos du plateau demandé ainsi que la représentation du plateau et l'identifiant du plateau suivant selon l'ordre établie précédemment afin de savoir quel plateau appeler lorsque le joueur termine son niveau

Difficultés rencontrées :

Implémentation de Sequelize : vu que c'était l'une des premières fois qu'on utilisait l'ORM sequelize pour gérer les données de la base MySQL, nous avons eu un peu de mal au début. Notamment la gestion des relations entre les modèles de données. Mais il s'est avéré très utile pour persister et récupérer facilement nos données de plateaux de jeu.

II/Créateur de niveaux en VueJS

Lien : https://sokoban-admin.onrender.com/

Justification de la technologie :

L'idée était de simplifier le processus de création des plateaux de jeu pour éviter de devoir les créer manuellement. De plus, l'utilisation de VueJS a permis à Théo de mettre en pratique ses acquis en entreprise, et à Paul de renforcer ses compétences acquises lors des cours.

Justification des dépendances :

vue, vue-router, vuex : Les trois principaux packages nécessaires pour développer une application en Vue.js, y compris la gestion des routes et des états.
axios : Pour faire des requêtes HTTP à notre API.
jwt-decode : Pour déchiffrer les jetons JWT renvoyés par notre API, permettant ainsi de gérer l'authentification.

Difficultés rencontrées :

Drag and drop : Faire fonctionner correctement le drag and drop des images

III/ Application mobile en React Native

Lien GitHub :

https://github.com/PaulGub/sokoban-reactNative

Justification de la technologie :

L'usage de React Native pour l'application mobile Sokoban était une exigence du projet. Cette technologie nous permet de construire une application mobile avec JavaScript, tout en offrant des performances proches de celles d'une application native.

Justification des dépendances :

expo : Il nous permet d'exécuter du code JavaScript sur notre téléphone sans avoir à passer par une phase de build avec Xcode ou Android Studio.
react-navigation : Pour le routage entre les différents écrans de notre application mobile.
axios : Pour faire des requêtes HTTP à notre API.
expo-av et react-native-sound : Pour la gestion du son dans l'application.
expo-linear-gradient et react-native-linear-gradient : Pour créer des dégradés linéaires dans notre interface utilisateur.
react-native-confetti : pour afficher une animation de confetti lorsqu'on réussit un niveau
async-storage : stockage de donnée "clé-valeur" asynchrone non chiffré

Difficultés rencontrées :

Animation du personnage : nous avons eu du mal à faire en sorte qu'une séquence d'images s'affichent lorsqu'on choisit une direction de déplacement pour le personnage. Nous affichons donc juste une image du personnage correspondant à la direction cliqué et non une animation

Gestion des déplacements : La gestion des déplacements du personnage a été une partie assez complexe, notamment pour faire en sorte d'afficher de nouveau le point de destination une fois que le personnage soit passé dessus. Pour résoudre ce problème, quand le personnage se déplace sur une case autorisée, nous stockons dans une variable dynamique la valeur de cette case pour l'afficher de nouveau une fois qu'il l'aura quitté.

Effet sonore : Problème de dépendance, nous avons voulu utiliser des dépendances réserve à React Native ne fonctionnant pas avec Expo.