# Rapport de notre projet Sokoban

Corentin BACQUÉ-CAZENAVE et Hector BIDAN

## Contexte et but du projet

Dans le cadre de notre cours de programmation mobile, nous avons dû réaliser un projet de Sokoban en React-Native avec une partie back end et une partie front end.

L'objectif du projet était donc de réaliser un jeu de Sokoban jouable aussi bien sur Android que sur iOS en s'appuyant sur la puissance de React-Native pour assurer la compatibilité multi-plateformes.

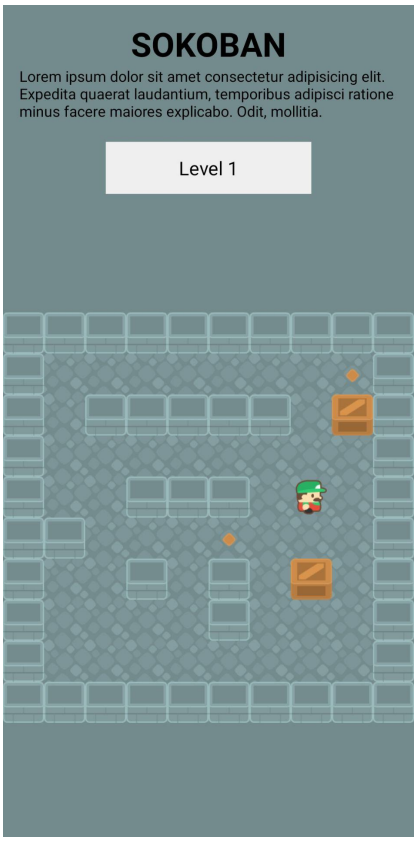
L'application devait contenir une API afin de récupérer les plateaux et les niveaux, eux-mêmes stockés dans une base de données. Ensuite, la partie front end via l'interface utilisateur était chargée d'assurer l'affichage du plateau et le mouvement des joueurs.

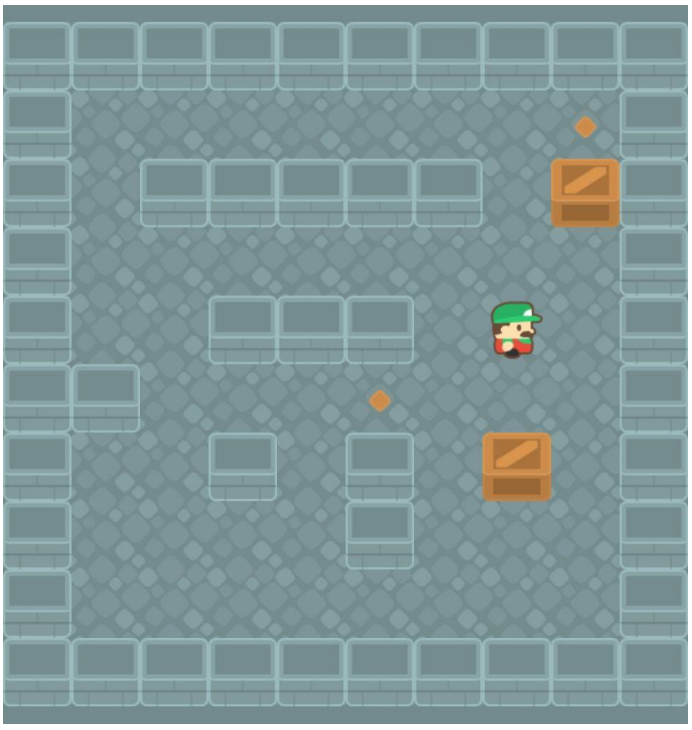
## Répartition des tâches

Dans notre petite équipe, il a fallu se répartir les tâches, deux personnes, deux parties, rien de plus simple. Corentin a donc réalisé la partie Back end, pendant que Hector s'est occupé du front.

## Disponibilité du code

Notre code est disponible sur notre [dépôt GitHub]



# Réalisation

## Back End

Nous avons choisi de réaliser le back end en Symfony d'abord parce que nous avions déjà l'habitude de cette technologie, mais aussi parce que cela nous a semblé simple et cohérent avec notre front.

En ce qui concerne la base de données, nous avons utilisé phpmyadmin, là encore pour des questions d'habitude.

Enfin pour la partie back, l'API a été réalisée avec API-Platform, ici c'est la simplicité et la facilité qui ont orienté ce choix. Avec de simples annotations dans le code, nous avons pu réaliser grâce à API-Platform une API qui contient tout ce dont nous avons besoin.

Pendant la réalisation du back end à proprement parler, il n'y a eu quasiment aucun problème à signaler, la mise en place de la base de données a été simple et rapide grâce aux automatisations proposées par Symfony.

Ensuite, cette fois grâce à la simplicité de mise en œuvre d'API-Platform, l'API s'est elle aussi réalisée très simplement.

C'est au moment de la connexion de l'API avec le FrontEnd que les premiers problèmes sont apparus à cause de problèmes d'ouverture de ports et donc d'accès, y compris en local. Quelques ajustements à l'API ont donc dû être réalisés afin que tout puisse s'interconnecter et fonctionner correctement.

Pour des raisons de temps, nous n'avons pas mis en place de tests unitaires, nous avons manuellement testé notre code au fur et à mesure de sa réalisation. Nous avons également choisi de réaliser les deux parties indépendamment l'une de l'autre dans un premier temps sur deux branches Git distinctes afin de ne pas se marcher dessus et d'avoir le moins de conflits Git à résoudre possible. Lorsqu'une partie était assez avancée pour pouvoir être testée dans de bonnes conditions avec l'autre partie, un merge des deux branches a été faite.

## Front End

Au début du projet, je me suis confronté à l'incertitude quant à la façon de gérer le jeu. J'ai commencé à avancer instinctivement jusqu'à obtenir un jeu jouable, incluant l'affichage de la carte, les déplacements du joueur et le mécanisme de poussée des caisses. Cependant, lorsque j'ai repris le projet quelques jours plus tard, je me suis rapidement rendu compte que je me perdais dans tous les tests réalisés à la hâte. C'est pourquoi j'ai décidé de repartir de zéro et de mieux segmenter mon projet en utilisant les contextes React, tout en important les parties pertinentes de ma version précédente. Cette approche a permis d'améliorer considérablement la structure du code et de faciliter son évolution.

Malgré ces avancées, j'ai été confronté à un problème récurrent dès le premier cours, pour lequel je n'ai malheureusement pas trouvé de solution satisfaisante malgré de nombreuses heures de travail. Ce problème était lié au temps de réaction entre le mouvement du joueur et le glissement effectué sur l'écran tactile. J'ai exploré différentes solutions telles que le changement de bibliothèques, l'optimisation des performances, la réduction de l'utilisation des useEffect, ainsi que la refonte de la partie gestion des mouvements. Malgré tous mes efforts, le délai imparti pour la remise du projet se rapprochant, j'ai préféré me concentrer sur les autres aspects du jeu, même si cela a altéré l'expérience de jeu.

Une fois que le jeu était fonctionnel, j'ai intégré la partie développée par Corentin pour gérer les appels à l'API. Cependant, j'ai été confronté à une erreur étrange liée à Expo, qui m'a causé de nombreux problèmes. Je ne comprenais pas pourquoi mes requêtes ne fonctionnaient pas, alors qu'elles étaient opérationnelles sur le web. Comme beaucoup d'autres étudiants de notre classe, il a été nécessaire de remplacer "localhost" par l'adresse IP de l'hôte. Cependant, cela ne fonctionnait pas dans mon cas, sans raison apparente. J'ai failli abandonner, mais quelques jours plus tard, en essayant sur une autre connexion Internet, j'ai finalement réussi. Je ne comprends toujours pas pourquoi cela ne fonctionnait pas avec mon réseau WiFi, mais seulement avec le partage de connexion. Malheureusement, même après avoir obtenu les réponses de l'API, j'ai constaté que la structure de mon application et celle fournie par l'API n'étaient pas parfaitement compatibles, ce qui a nécessité quelques ajustements au niveau de l'API. De plus, alors que j'avais directement accès à l'objet contenant les données en utilisant des données en dur, l'appel à l'API entraînait un délai de réponse du serveur, perturbant ainsi le fonctionnement général de l'application. Étant donné que la date limite de rendu approchait, cela a eu un impact négatif sur la qualité du code.

# Fonctionnalités

En plus des fonctionnalités de base du jeu (déplacements, chargement des niveaux…), nous avons ajouté quelques petits détails pour améliorer l’expérience utilisateur.

Le personnage, les boîtes et les cibles sont définis en dehors de la carte, ce qui permet de les déplacer de manière fluide et non case par case. Cette approche apporte une sensation de fluidité et de réalisme lors des déplacements des éléments du jeu et cela m’a permis d’essayer les animations dans React native..

Le joueur se tourne dans la direction dans laquelle il se déplace pour un peu plus d’expérience utilisateur et pour essayer de cacher le bug des déplacements.

La carte prend automatiquement la largeur de l’écran, on calcule la taille de la tuile en fonction du nombre de colonnes et la taille de l’écran.

Le jeu utilise des assets gratuits pour dessiner la carte et les objets du jeu.

# Conclusion

En conclusion, ce projet de développement d'un jeu Sokoban en React Native avec Expo a été un véritable défi. Nous avons dû faire face à plusieurs problèmes techniques, notamment la gestion du temps de réaction entre les gestes de l'utilisateur et les mouvements du joueur. Au niveau du backend, l'intégration de l'API s'est bien passée et il n'y a pas eu de complications hormis les difficultés liées à l'appel réseau. Malgré ces difficultés, ce projet nous a permis de monter en compétences sur React et de découvrir les spécificités de React Native et d'Expo.

# Perspectives d'amélioration

Pour améliorer ce projet à l'avenir, plusieurs aspects pourraient être pris en compte. Tout d'abord, il serait important de trouver une solution au problème du temps de réaction pour garantir une expérience de jeu fluide. Ensuite, une attention particulière pourrait être portée à l'optimisation des appels à l'API afin d'améliorer les performances globales de l'application. De plus, il serait intéressant de se pencher sur l'amélioration de la structure du code pour faciliter l'intégration de futures fonctionnalités et maintenir une base solide pour le projet.