

## Pré-rapport de l'analyse des besoins

# Jeu mobile en Flutter : Chrono Chroma

Antoine Chevaleyre - Adrien Schloesser - Justine Pruliere - Julien Sacquard - Lucas Cordurié

Tuteur : Alexandre Leroux

Année : 2022 - 2023

[MoSCoW](#)

[Présentation du projet](#)

[Test de l'ascenseur](#)

[Diagramme de navigation](#)

[Diagramme d'architecture logicielle](#)

[Ébauche de base de données](#)

[Cas d'utilisation](#)

[Exemple de diagramme de séquences sur la génération de la carte](#)

[Étude technique](#)

[Étude de l'existant](#)

[Maquettes](#)

[Glossaire](#)

[Bibliographie](#)

# MoSCoW

## Must have

- S'inscrire
- Se connecter au même compte que sur son téléphone
- Menu d'accueil
- Déplacements et actions du joueur
- Interface de jeu, boutons à l'écran
- Génération ~aléatoire~ de l'environnement de jeu
- Graphisme minimalistes, plus fonctionnel qu'esthétique
- Pas de fichier de sauvegarde, stockage en base de données pour cross-progression
- Accéder aux statistiques de son profile

## Should have

- Amélioration du personnage permanente et temporaire
- Système de points ou de timer
- Ennemis et/ou obstacles
- Écran récapitulatif de fin de partie
- Enregistrement des scores en base de données et classement de joueur
- Système de seed
- Possibilité de publier sa seed et sa partie pour challenge ouvert
- Fantômes d'un joueur en confrontation
- Système d'amis

## Could have

- Connexion via portail compte Google
- Multijoueur (limité à 2 joueurs, style co-op)
- QR code de partage de compte ou de partie (portail multijoueur)
- Niveaux (toujours générés aléatoirement) séparés par des salles fixes (checkpoint)
- Mécanique autour de la couleur (lien entre mécanique de jeu et visuel)
- Mettre en pause
- Des sprites personnalisées
- Des éléments sonores

## Won't have

- Confrontations de joueurs en simultanée (envisageable après un multijoueur bien fonctionnel à la limite)
- Nouvelles armes changeant la façon d'interagir (projectiles, dégâts passifs...)
- Compétences particulières de mouvement (dash, glissade, grimper...)
- Des boss (IA, patterns...)

"M" « Doit être effectué »

"S" « Devrait être accompli autant que possible »

"C" « Pourrait être réalisé dès lors où cela n'a pas d'impact sur les autres tâches ».

"W" « Ne sera pas effectué cette fois, mais sera fait ultérieurement »

Ces éléments peuvent être repris dans le même ordre d'apparition pour créer la feuille de route du projet.

# Présentation du projet

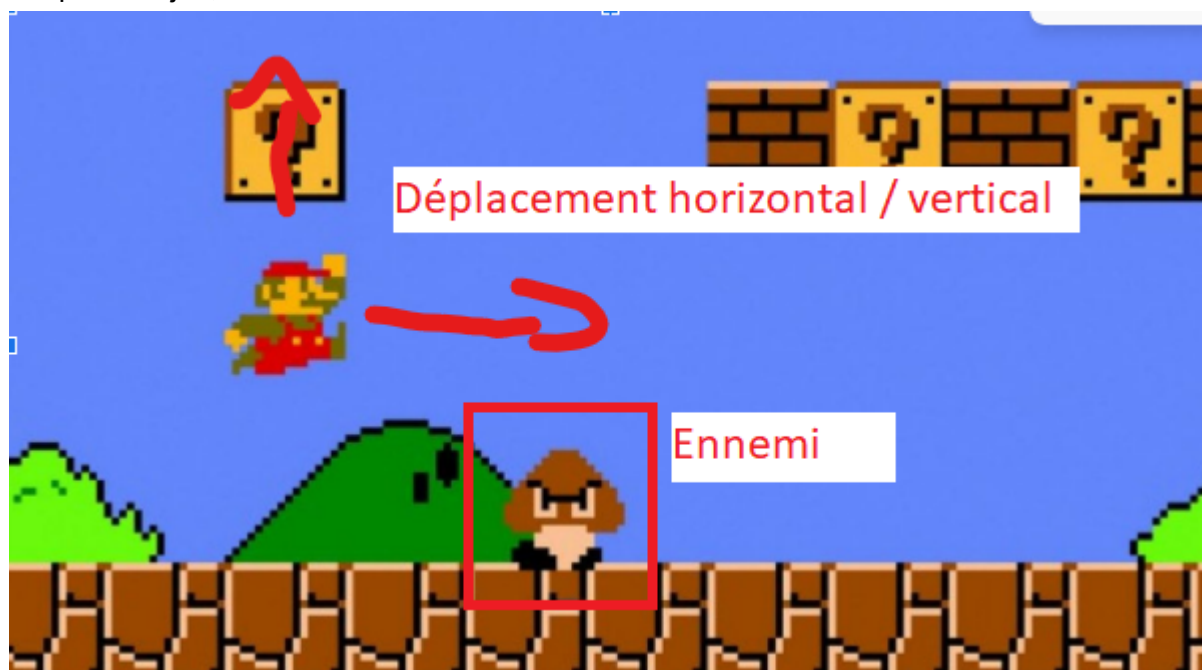
L'objectif de ce projet est de créer un plateformer de type Rogue Lite, pour y parvenir, nous utiliserons le langage Dart couplé à son Framework Flutter.

Il sera entouré de fonctionnalités communautaires permises par des menus en dehors du cœur de jeu et par une base de données.

Ainsi, on peut séparer en 2 catégories les éléments à produire : éléments de "gameplay" et éléments de "structure et social".

Concernant le gameplay :

L'idée est d'avoir quelques écrans fixes de jeu correspondant à des menus (début de partie, pause, fin de partie) et un environnement en 2D à défilement horizontal servant d'espace de jeu, à la manière d'un "Mario Bros".



Un personnage contrôlé par le joueur peut se mouvoir dans cet espace (déplacements horizontaux et verticaux) à travers des couloirs générés à l'aide d'aléatoire (avec murs, sol, plafond, obstacles...) et interagir avec certains éléments.

L'adversité à laquelle le joueur sera confronté se retrouve en quelques ennemis, des éléments statiques ou éventuellement mobiles pouvant blesser le joueur et être détruits.

Cependant, le principal ennemi, le challenge, réside dans une course contre-la-montre où le temps est une ressource vitale pour parvenir à une victoire.

Suivant une logique de jeu se rapprochant du "Rogue Lite", seront incluses des améliorations facilitant la progression (certaines permanentes, d'autres se limitant à l'essai en cours).

Des mécaniques multijoueurs sont à prévoir. Idéalement, de la coopération pourra être incorporée, laissant 2 joueurs agir en simultanée dans une même partie.

Le style visuel se rapprochera de ce que l'on peut qualifier de "pixel art", une esthétique minimaliste restant à notre portée. Une mécanique de couleurs liées aux éléments de gameplay pourra donner une thématique visuelle au projet (jouant sur la présence et l'absence de couleurs) et apporter en clarté sur les éléments de jeu.

Pour ce qui est de la structure et du social :

On s'intéresse plus à ce qui entoure le jeu, que ce soit les menus hors du jeu ou des éléments se greffant sur le gameplay.

Le multijoueur évoqué plus tôt, impliquant a minima l'enregistrement de données de jeux en base de données et la présentation de celles-ci dans certaines pages comme un tableau des scores. L'enregistrement des actions en jeu permettrait également de revoir une partie jouée à la manière d'un "replay" dans lequel il serait possible de se mouvoir.

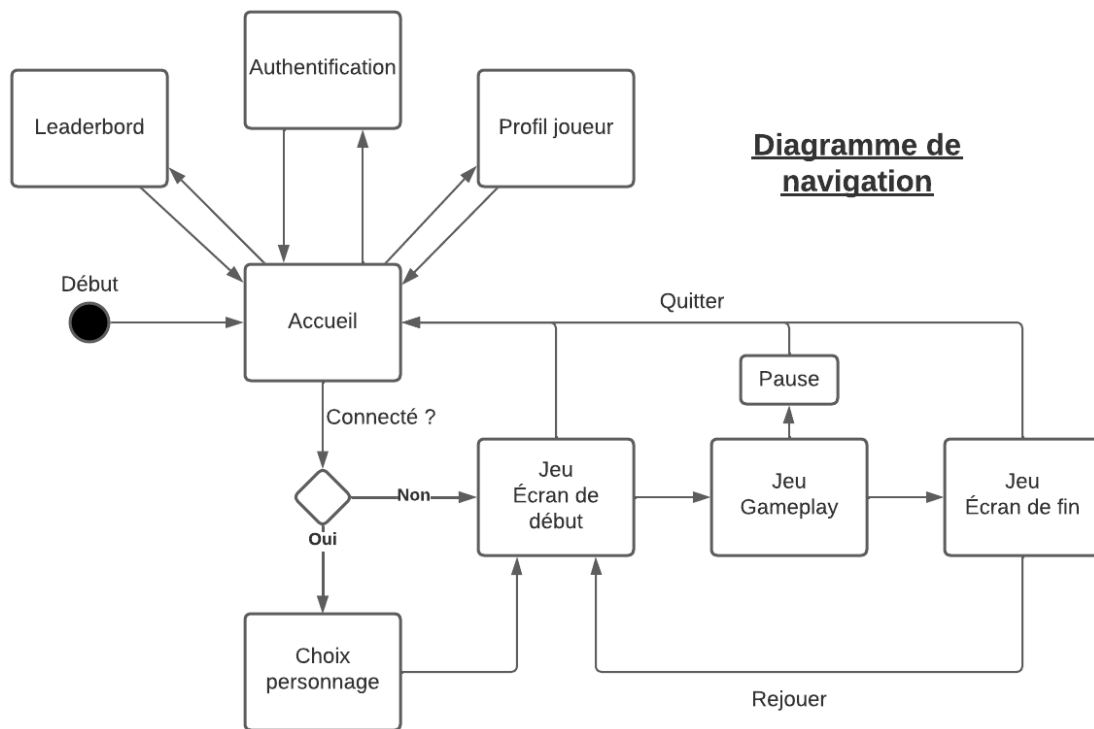
La logique sociale peut être retranscrite dans un partage de ces parties à l'aide de "seed", permettant de bloquer l'aléatoire de génération du jeu et laissant plusieurs joueurs faire une tentative dans le même environnement.

Tout cela implique la création de comptes avec profils consultables, possiblement via un code QR.

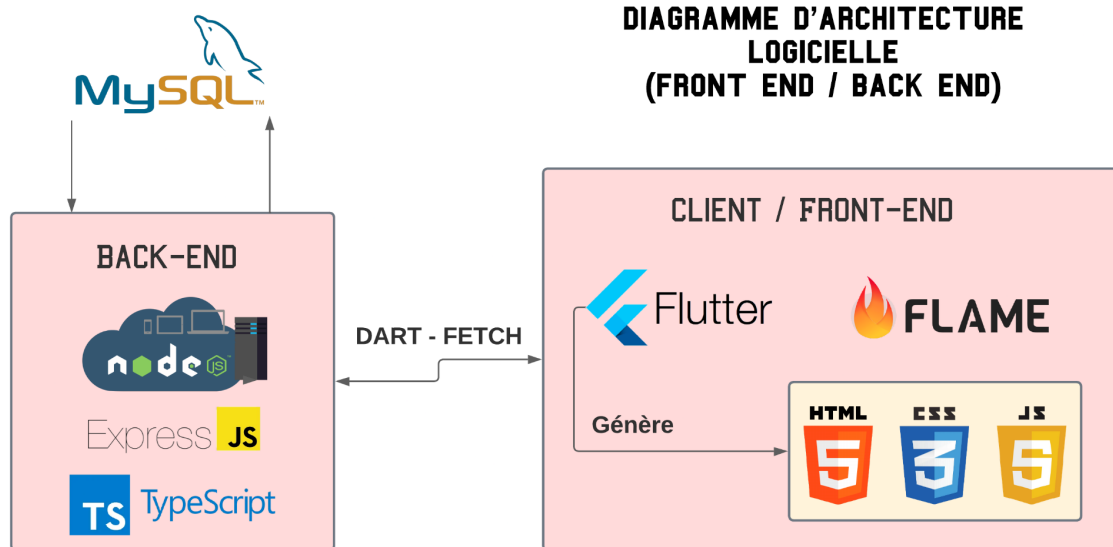
## Test de l'ascenseur

C'est un jeu à défilement horizontal dont le but est de gagner une course contre-la-montre qui devient plus simple au fil des parties, le tout dans un environnement généré avec une part d'aléatoire. Des tableaux de scores, des affrontements, de la coopération et du partage de jeu seront idéalement inclus afin de sortir du pur jeu et proposer un service.

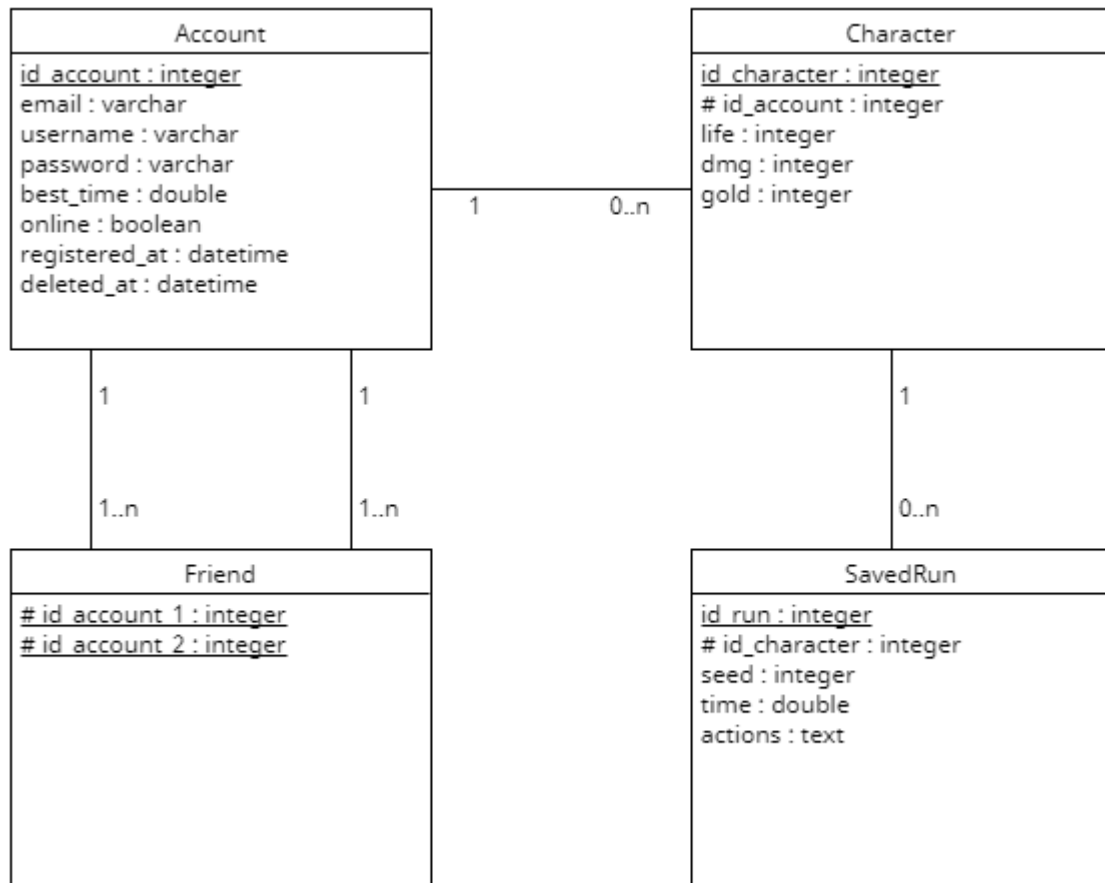
# Diagramme de navigation



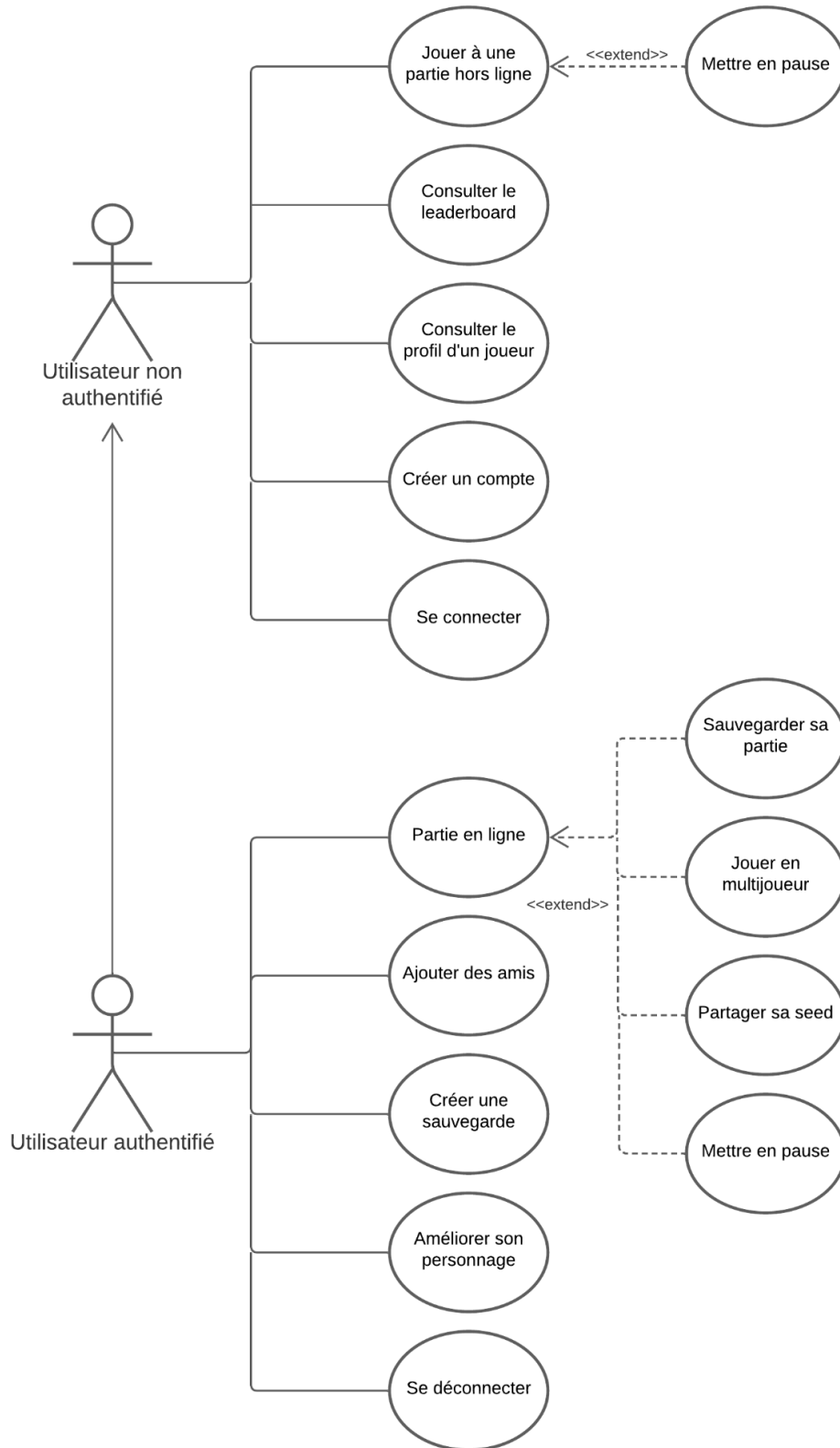
# Diagramme d'architecture logicielle



## Ébauche de base de données

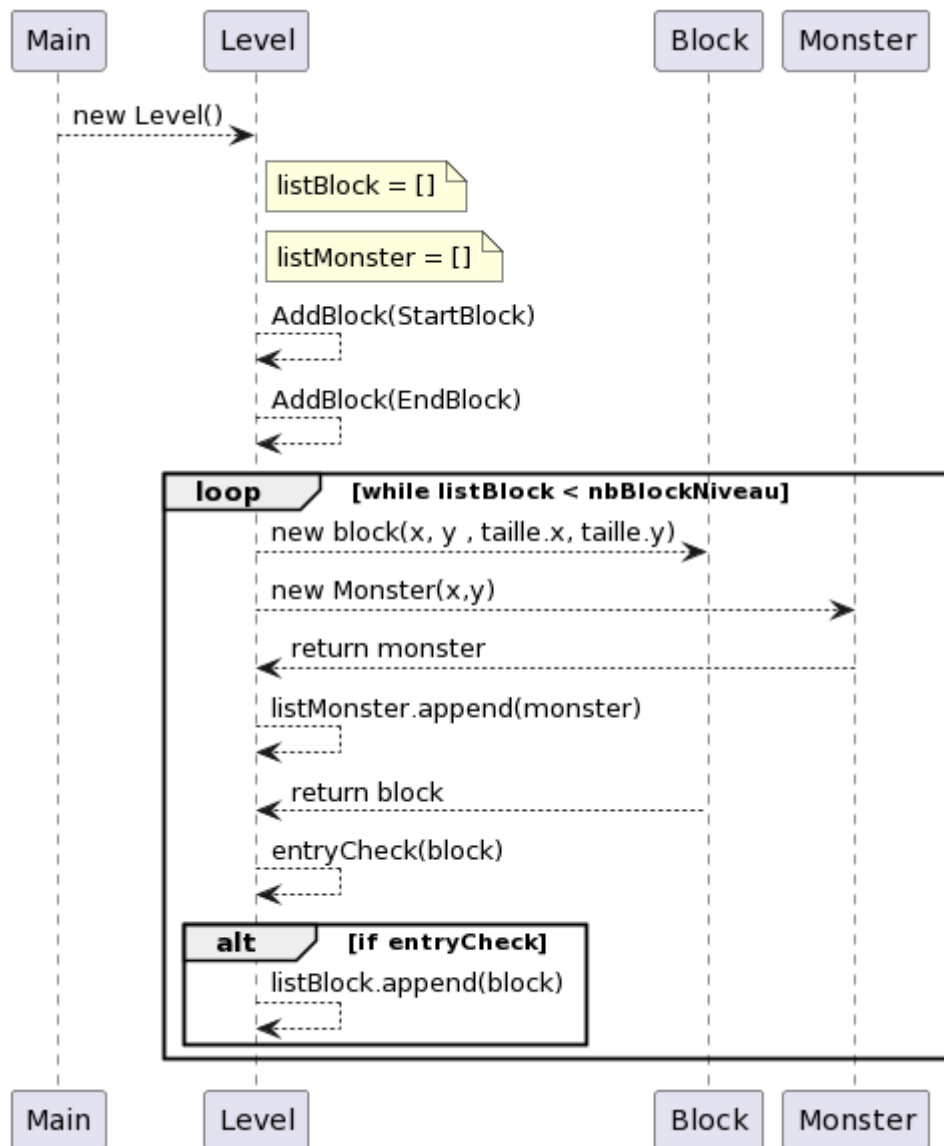


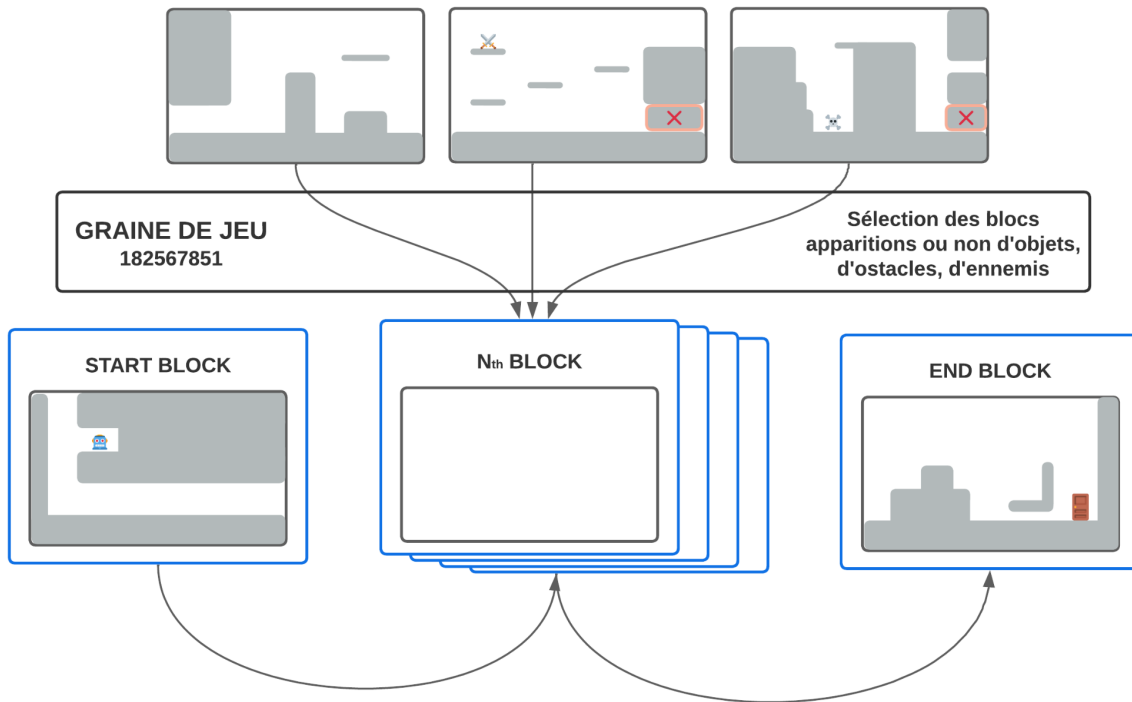
# Cas d'utilisation





## Exemple de diagramme de séquences sur la génération de la carte





# Étude technique

## *Accessible via diverses plateformes ?*

Flutter permet de mettre en place l'application sur toutes les plateformes : Windows, Mac, iOS, Android, et web nativement. Les échelles de résolutions et l'affichage aux différents formats est bien entendu facilité par Flutter.

## *Génération de carte aléatoire et seeds ?*

La librairie Flame de Flutter offre des possibilités en ce sens, comme dans [cet article](#) qui fait partie d'une série expliquant comment réaliser un jeu en flutter, une source de valeur pour nous.

Il reste de points importants à considérer dans cette génération, comme l'explique [cet article](#), il y a 3 impératifs.

La génération doit permettre un parcours faisable (pas de zones totalement infranchissables) mais offrant du défi (des zones non franchissables et des obstacles) sans que la répétition se fasse trop sentir.

En ayant créé des schémas de cartes, des obstacles, des ennemis et des récompenses pour changer le comportement du joueur, on aura déjà une génération pertinente.

La question de la faisabilité demeure. Soit une IA doit être créée pour vérifier ce que la génération donne, soit une fonctionnalité de jeu est implémentée afin de contourner les possibles blocages, dans le cas où ceux-ci ne sont pas trop récurrents. Sur cette dernière option, puisque le jeu est un contre-la-montre, cela pourrait être une façon de ralentir le joueur, rendu de moins en moins gênante par des améliorations permettant de détruire les obstacles plus vite, par exemple.

## *Jeu en simultané dans une même partie ?*

Une possibilité serait d'utiliser Flutter et le Flame-Engine, comme cela a été fait pour ce jeu dont le dépôt GitHub est accessible : [Multijoueur club-pingouin Nakama](#), une autre librairie, offre des possibilités similaires.

La problématique est récurrente dans les jeux et a déjà été résolue de nombreuses fois, pas de soucis à se faire en la matière.

Les modifications du jeu en comptant deux joueurs plutôt qu'un seront à prendre en compte et pourront être limitées par des choix tels que : ne sauvegarder que la progression de l'hôte, copier les statistiques de l'hôte pour l'invité, entre autres.

## *Intégrer des sprites, des textures, des sons dans le jeu ?*

Grâce à la librairie "Flame", il est possible d'intégrer des sons, des textures ainsi que des sprites.

Les composants de Flame permettent une variété d'animations pour les sprites, exécutées automatiquement à intervalles réguliers ou pendant un temps selon certaines actions.

[Exemple de gestion complète des sprites](#). Le soin de fournir les éléments graphiques nous revient, mais l'évolution à l'écran de ceux-ci est bien prise en charge par l'outil.

### ***Possibilité de charger des données dans le code depuis des fichiers de configuration ?***

Création d'un fichier Dart de configuration, car des difficultés sont présentes avec des fichiers de configuration javascript. On pourra y placer les variables de jeux concernant tout ce qui ne relève pas de l'aléatoire. Statistiques de personnages, de génération, d'ennemis...

### ***Charger des données depuis une base de données ?***

À l'aide d'un serveur node Js qui interagit avec une base de données, il est possible de faire une persistance de données sur le jeu, dans le cas où le serveur est non fonctionnel, il est toujours possible d'utiliser les "sharedpreference" intégrées à Flutter. Cette centralisation des données en base sera appréciable afin de permettre une cross-progression, à comprendre, la possibilité de ne pas perdre ses données d'un appareil à l'autre.

### ***Jouer au jeu avec un timer intégré / chronomètre ?***

Cette option ne semble pas intégrée par défaut, mais il doit être possible de l'implémenter de nous même ou de le trouver en open source.

Le besoin est surtout d'avoir le temps qui évolue sous forme de valeur pour avoir des changements à certaines valeurs du timer.

[Ces packages](#) font des timers et chronomètres en Flutter avec widget et compagnie. Pas tout à fait pertinent dans notre cas, mais utile pour comprendre et répliquer le fonctionnement de ces outils.

## Étude de l'existant

Le site du Framework Flutter propose une liste des jeux réalisés avec celui-ci. Dans cette liste, nous avons pu trouver quelques jeux ressemblant à l'idée que nous nous faisons de notre future réalisation.

### **Dino run :**

*Page de présentation* : <https://madewithflutter.net/dino-run/>

*GitHub* : [https://github.com/ufrshubham/dino\\_run](https://github.com/ufrshubham/dino_run)

Jeu en 2D à défilement horizontal avec ennemis à éviter

### **Bob box :**

*Page de présentation* : <https://madewithflutter.net/bob-box/>

*GitHub* : [https://github.com/bluefireteam/bob\\_box](https://github.com/bluefireteam/bob_box)

Jeu en 2D à déplacement vertical avec ennemis à éviter et pièces à ramasser

### **Break guns using gems! :**

*Page de présentation* : <https://madewithflutter.net/break-guns-using-gems/>

*GitHub* : <https://github.com/bluefireteam/bgug>

Jeu en 2D à défilement horizontal avec gemmes à ramasser au long du chemin

### **Darkness dungeon :**

*Page de présentation* : <https://madewithflutter.net/darkness-dungeon/>

*GitHub* : [https://github.com/RafaelBarbosatec/darkness\\_dungeon](https://github.com/RafaelBarbosatec/darkness_dungeon)

Jeu en 2D avec déplacement sur une map : ennemis à éviter, coffres à ouvrir, génération de map aléatoire

---

Nous avons également basé notre idée sur des jeux déjà existants développés sous d'autres technologies :

**Dead Cells** : référence rogue-like et base de jeu (déplacements, style)

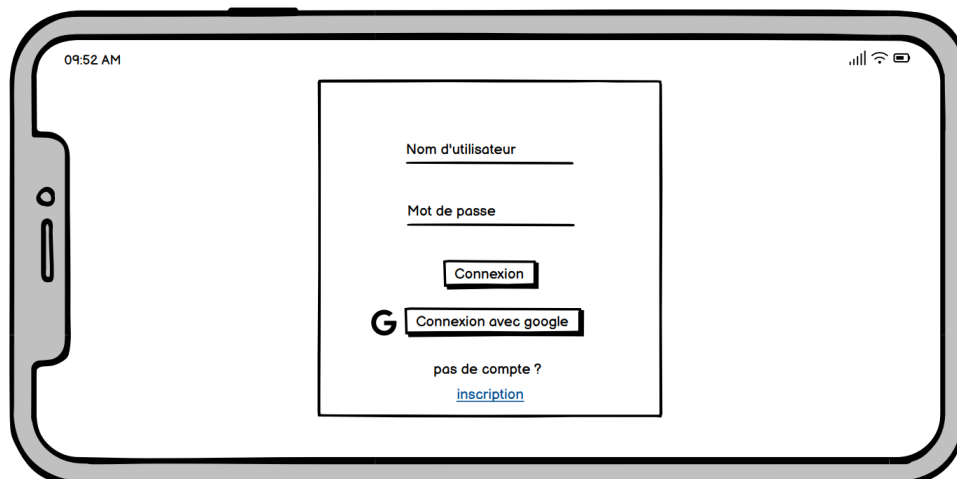
**Minit** : principe de temps restreint et pixel art aux couleurs rares

**Mario Bros** : référence du plateforme

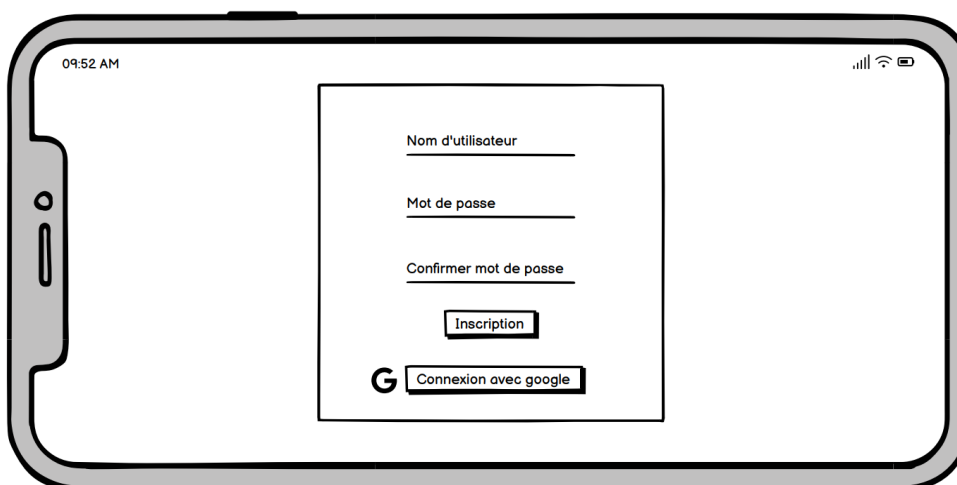
# Maquettes



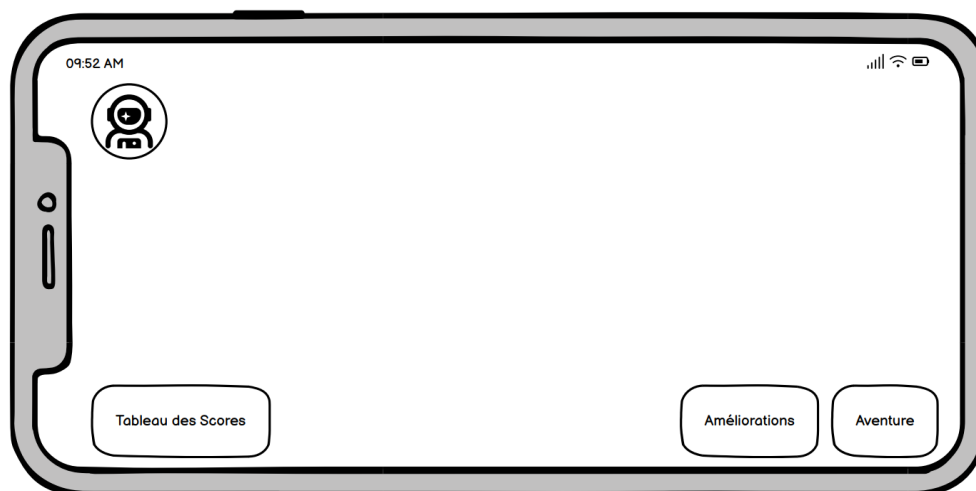
Page à l'ouverture de l'application



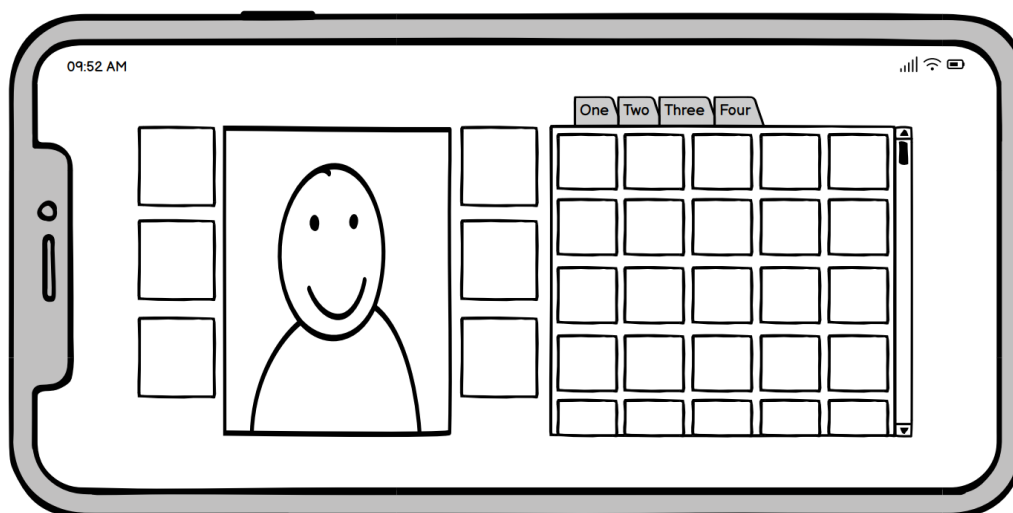
Page de connexion



Page d'inscription



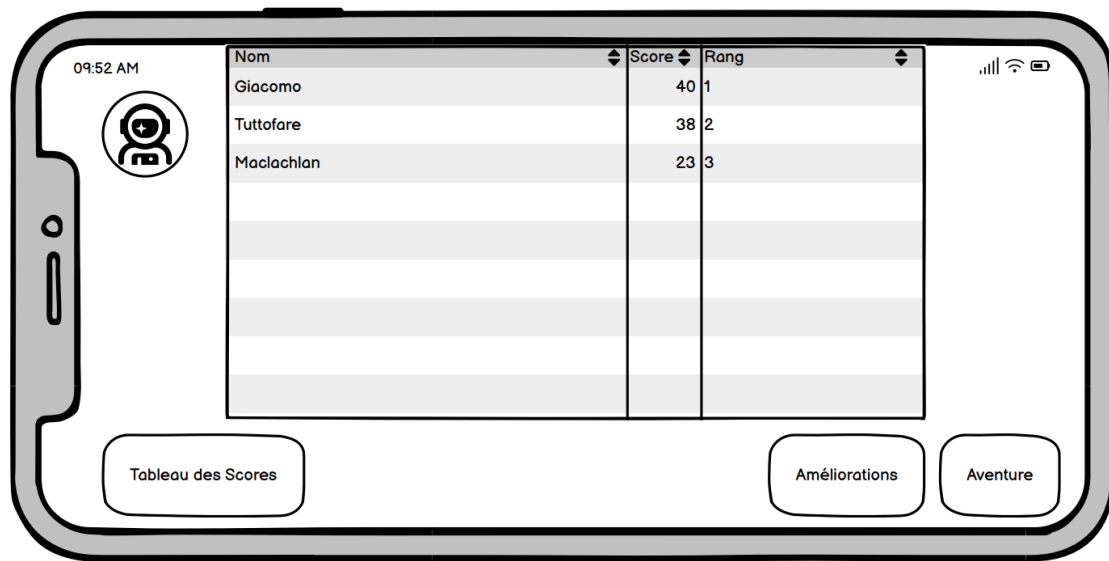
Menu dans le jeu



Page d'amélioration du personnage



Jeu



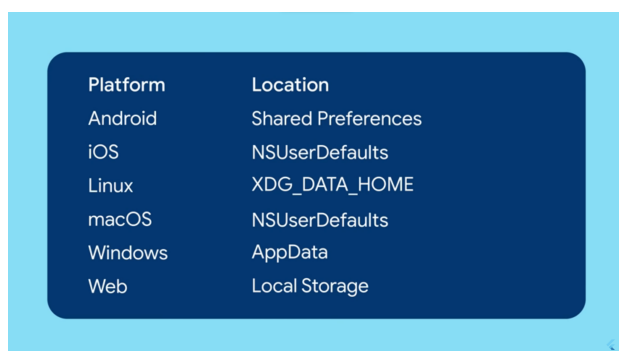
Page de classement de tous les joueurs



# Glossaire

**Roguelite:** dérivé allégé de Rogue-Like, caractérisé par certaines mécaniques : l'utilisation de niveaux générés procéduralement pour les hybrider avec d'autres genres. La notion de mort permanente est également très allégée. Mort après mort, dans un Rogue-Lite, vous allez améliorer votre personnage qui sera de plus en plus fort, partie après partie.

**Sharedpreferences:** Les préférences partagées (shared preferences) sont un système de sauvegarde clé-valeur qui est disponible sur toutes les plateformes avec les mêmes codes, elles agissent comme le LocalStorage (stockage local) par exemple.



Platform	Location
Android	Shared Preferences
iOS	NSUserDefaults
Linux	XDG_DATA_HOME
macOS	NSUserDefaults
Windows	AppData
Web	Local Storage

Un **platformer**, ou jeu de plates-formes, est un genre de jeu vidéo où le joueur incarne un avatar qui doit sauter sur des plateformes et éviter des obstacles.

**Side scroller** ou jeu à défilement horizontal est un sous-genre de jeu avec une vue de côté où un avatar se déplace de gauche à droite pour accomplir un objectif. Un exemple notoire serait le jeu Mario Bros.

**Seed** : Nombre généré aléatoirement servant à créer une carte (environnement) de jeu, lui transmettant son caractère aléatoire. Créer une carte avec une seed définie retire l'aléatoire de la génération et permet d'obtenir la même chose à chaque fois.

**Map** : Une carte de jeu, l'environnement dans lequel évolue l'avatar.

Le **pixel art** est une représentation graphique de personnages ou autres dans laquelle le dessin est effectué pixel par pixel, laissant ces derniers bien visibles, contrairement aux représentations tenant de la haute définition.

Un **Framework** est un ensemble de composants logiciels qui proposent des fonctionnalités supplémentaires à un langage.

Le **gameplay** regroupe l'intrigue et la façon de jouer dans un jeu vidéo.

# Bibliographie

## **Série très utile sur la création d'un jeu en Flutter et Flame :**

[Partie 1](#) : create your game

[Partie 2](#) : game basics

[Partie 3](#) : sprites and input

[Partie 4](#) : collision and viewport

[Partie 5](#) : level generation and camera

[Partie 6](#) : effect and sound design

[Partie 7](#) : play the game

## **Documentations des outils :**

[Documentation Flame - beaucoup d'outils pour créer le jeu](#)

[Firebase - permettant d'étendre le jeu à du multijoueur \(Par Google, tout comme Flutter\), voir SupaBase en cas de problème](#)

[Nakama semble permettre des fonctionnalités similaires, au cas où Firebase ne convient pas](#)

[Express en Node JS](#)

[Knex pour la base de données](#)

[TypeScript pour un JS plus typé](#)

## **Autres exemples de fonctionnalités envisagées :**

[Exemple de génération procédurale](#)

[Exemple de multijoueur](#)

[Ressources pour timer et chronomètre](#)

Certains des liens ci-dessus sont présents plus tôt dans le document, d'autres n'y apparaissent pas.