Rapport - DiceGame

Nathanaël MIESCH / Reika JACQUOT

M2 MIAGE SID
2025-2026

Table des matières :

Présentation du sujet

Dans le cadre du cours de patrons, il a été demandé la conception et l'implémentation d'un jeu, Dicegame.

Nous rappelons le cahier des charges :

- Le joueur lance 10 x 2 dés.
- Si le total fait 7, il marque 10 points à son score.
- En fin de partie, son score est inscrit dans le tableau des scores.

Ainsi, ce rapport a pour objectif de montrer les travaux effectués pour les deux étapes (conception et implémentation).

Conception

La conception suit les différents travaux effectués en cours d'analyse, conception et mise en oeœuvre de SI à base de patrons.
La conception a également pour rôle de proposer une représentation implémentable par la suite.

La conception est décrite ci-dessous :

Diagramme de classe

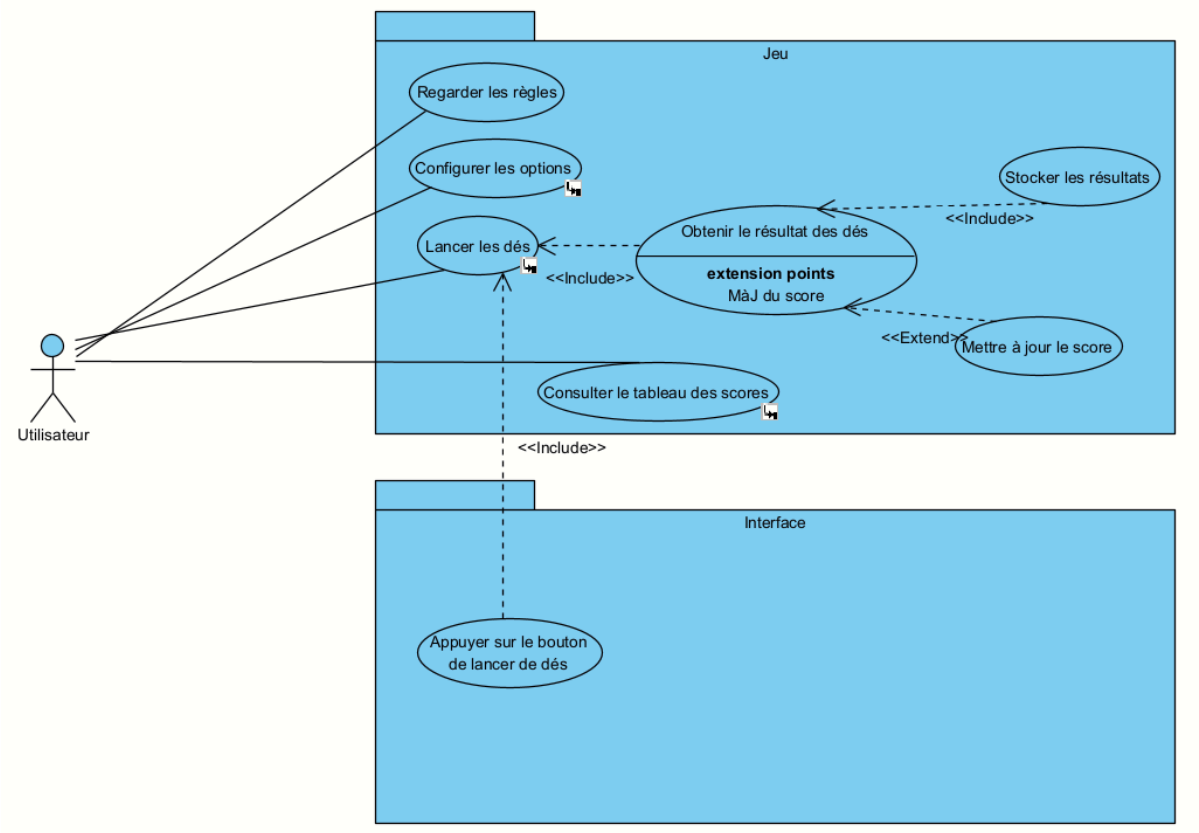


Diagramme de séquence

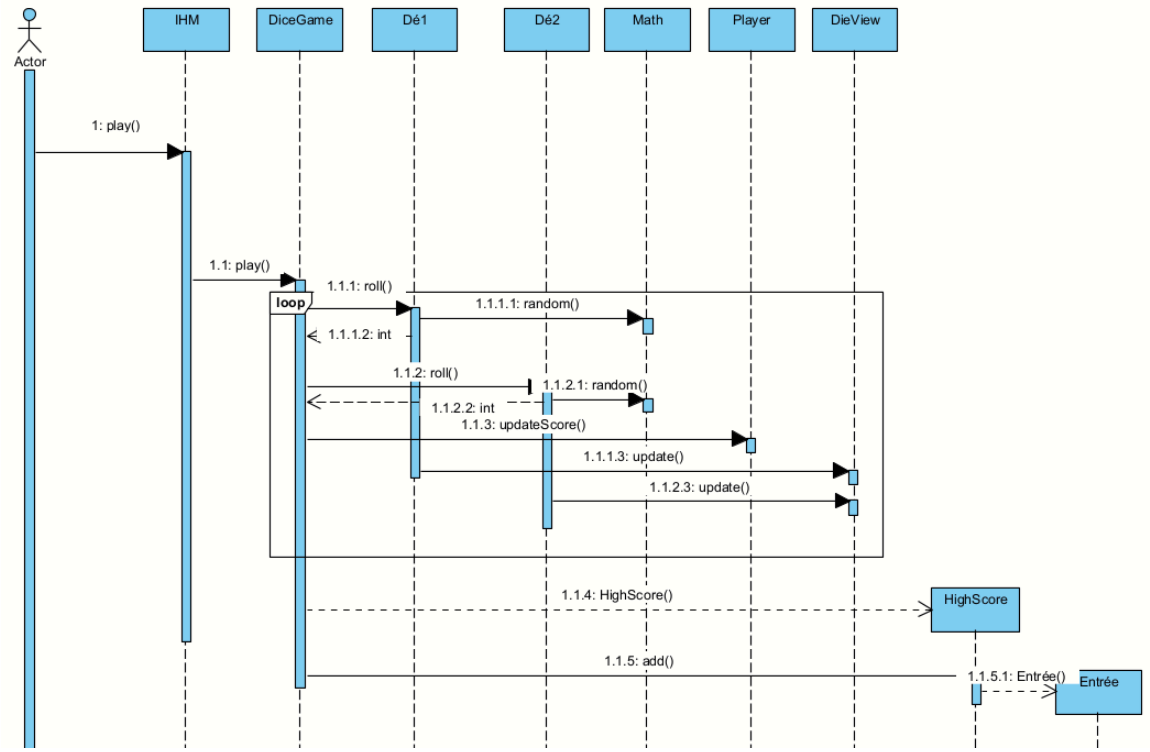


Diagramme d’activité : Lancer les dés

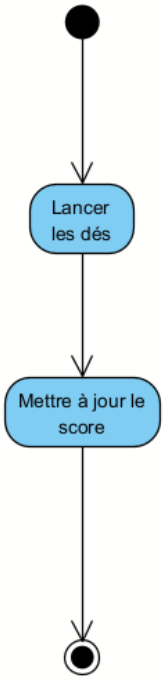
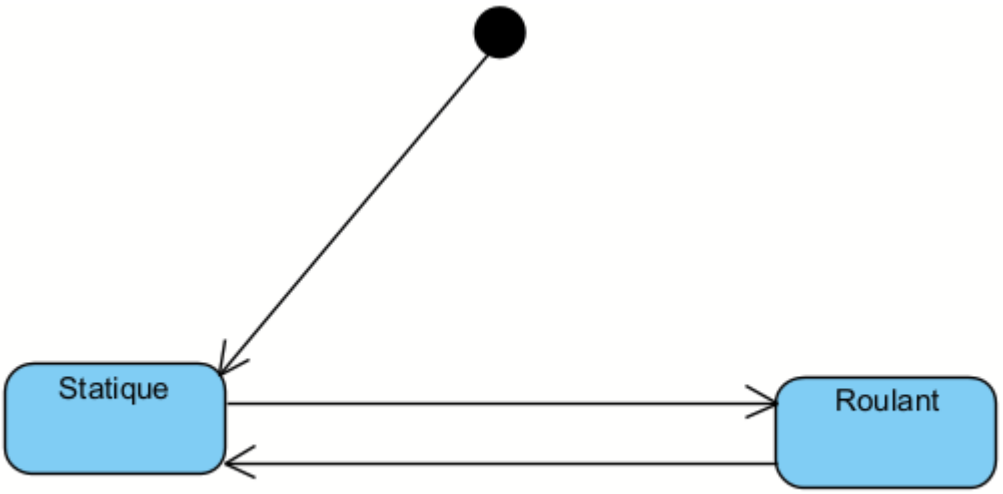


Diagramme d’états : Dé



Implémentation

L’implémentation suit le modèle de données fait sur la rubrique précédente.
L’implémentation est séparée en trois parties.

Base de données

La base de données est une instance Postgres créée avec une image Docker. Elle permet d’enregistrer les scores.

Pour démarrer la base de données :

cd db && docker compose up -d

API

Une API est présente pour gérer la persistance entre les données et la base de données. Elle transmet les scores entre la base de données et l’application web.

L’API est implémentée en Spring.

L’architecture est simple : un contrôleur, un service, un répertoire et une entité aux noms de ScoreController, ScoreService, ScoreRepository et Score respectivement.

Usuellement, le contrôleur reçoit l’appel API, transmet à ScoreService qui demande à ScoreRepository d’enregistrer sur la base de données ou de retourner des informations. L’application permet de récupérer les 10 meilleurs scores, d'enregistrer un score, de récupérer tous les scores et de récupérer les scores d’un joueur.

L’entité, Score, décrit ses attributs (ici, un ID, un id_joueur en String et un score en int).

Pour démarrer l’API :

cd API && mvn clean install && mvn package && java -jar target/api.jar

Application web

L’application web est disponible en Javascript.

L’application contient plusieurs fichiers :

- de.js
- highScore.js
- rules.js
- options.js
- Joueur.js
- DiceGame.js

Tous les fichiers sauf highScore.js et DiceGame.js sont des modèles et contiennent des informations qui vont rester au joueur courant. Seul le score est persistant et highScore fait appel à l’API, qui appele base de données pour tous les usages reliés au score.
DiceGame gère l’ensemble de ces objets ainsi que l’affichage des différents éléments à l’écran. Il permet de dérouler une partie de jeu.
Le serveur Python permet de rapidement lancer le serveur.

Pour démarrer l’application web :
cd web && python -m http.server
Le serveur est accessible en localhost:8000/