TDLog : Jelly

BATY Léo, BRUNOD-INDRIGO Luca, CAMPO Yannis,
LAURET Jéremy, LO Emmanuel

15/02/2019

Table des matières

L'objectif initial de ce projet était de créer un jeu multi-joueur en ligne, adaptation d'un jeu de plateau créé l'année dernière dans le cadre du cours de développement durable.

1 Présentation du jeu

Dans ce jeu, les joueurs incarnent chacun un pays. Ils doivent gérer le développement économique, social, et environemental du pays durant deux ères, depuis la révolution industielle jusqu'à nos jours. Chaque ère se décompose en tours appelés générations, pendant lesquels les joueurs jouent simultanément en construisant des bâtiments, recherchant des technologies et concluant des accords. Le tour suivant débute une fois que tous les joueurs ont indiqué avoir fini leur tour.

1.1 Déroulement d'une génération

Chaque génération se déroule en trois phases :

1. Phase de revenus
2. Phase principale
3. Phase évènements

1.1.1 Phase de revenus

Chaque joueur possède trois champs principaux :

— ses ressources stockables : UM (unité monétaire) et hydrocarbures.

— sa production excédentaire : UM, hydrocarbures, nourriture, pollution, déchets, électricité, régénération de l'environnement. Par exemple, une production négative de nourriture indique que le joueur doit en importer afin de satisfaire les besoins de son pays.

— l'état de son développement : social, économique, et environnemental (entier entre 0 et 100).

Durant la phase de revenus, chaque joueur reçoit son revenu en UM, doit importer nourriture et électricité s'ils sont en production négative, ou les exporter s'ils sont en production positive.
Les revenus d'hydrocarbures fonctionnent d'une manière un peu particulière. Au début de la partie, on constitue trois piles d'hydrocarbures représentant la réserve mondiale.

Lorsque l'on produit des hydrocarbures, on en ajoute aux ressources du joueur de la manière suivante :

— Tant qu'une pile n'est pas vide on prend les ressources toujours dans la même pile en commençant par la première.

— Si au début de la phase revenus il y avait des ressources dans la pile 1, on prend 3 hydrocarbures par point de production que l'on possède et on les place dans notre réserve.
— Si au début de la phase revenus la pile 1 était vide et la pile 2 non vide, chaque point de production rapporte 2 hydrocarbures.
— Si au début de la phase revenus les pile 1 et 2 étaient vides, chaque point de production rapporte 1 hydrocarbure.
— S'il n'y a plus de ressources dans la réserve, on n'en prend pas.

1.1.2 Phase principale

Durant la phase principale, les joueurs effectuent des actions de manière simultanée. Voici la liste des actions possibles :

— Construire un bâtiment débloqué. Chaque bâtiment a un coût en UM, des modificateurs au niveau de la production du joueur et de son développement, et un éventuel effet (ponctuel ou permanent).
— Acheter une technologie débloquable. Chaque technologie débloque une ou plusieurs technologies, un ou plusieurs bâtiments, et possède un éventuel effet (ponctuel ou permanent).
— Conclure des accords d'importation ou d'exportation. Chaque accord est proposé ou non de manière aléatoire au joueur, et réduit le coût des importations ou permet des exportations d'une ou plusieurs ressources pendant un certain nombre de générations.

Chaque joueur peut effectuer autant d'actions qu'il veut, dans la limite des fonds disponibles. Cependant, il ne peut plus construire de bâtiments dès qu'il a acquis une technologie, ceci simulant que la recherche met du temps à porter ses fruits.

1.1.3 Phase évènements

A la fin de chaque génération, on tire aléatoirement un évènement correspondant à l'ère en cours, et on l'applique à tous les joueurs. Les évènements ont des effets très variés dépendants de l'état de développement des joueurs. Certains s'appliquent de la même manière à chaque joueur, et donc regardent la moyenne (ou le minimum/maximum) de chaque état, d'autres s'appliquent individuellement. La fin de chaque ère est déclenchée par des évènements particuliers appelés évènements finaux. Ces évènements ont des effets plus radicaux que les évènements classiques, et déclenchent le début de l'ère suivante. A la dernière ère, l'évènement final déclenche un dernier tour sans évènements avant la fin de la partie.

1.2 Décompte des points

A la fin de la partie on décompte les points pour chaque joueur. Chaque joueur classe les trois échelles de développement dans l'ordre décroissant de sa position sur ces dernières,

et calcule son score de la manière suivante :

1. Chaque niveau atteint sur l'échelle de développement la plus haute rapporte 1 point.
2. Chaque niveau atteint sur l'échelle intermédiaire rapporte 2 points.
3. Chaque niveau atteint sur l'échelle la plus basse rapporte 3 points.

Le joueur possédant le plus de points remporte la partie.

2 Objectifs initiaux

Voici la liste des objectifs principaux fixés au début du projet :

1. Coder le jeu ainsi que le plus de fonctionalités possibles parmi celles décrites dans la section précédente.
2. Y ajouter un système qui permet de créer des parties, d'en rejoindre, d'en lancer, et d'en sauvegarder.
3. Créer un site internet hébergeant le jeu.

3 Organisation/architecture du code

Nous avons utilisé un backend en Python avec le framework Django, et un frontend Javascript avec la librairie React.

3.1 Backend Django

Nous avons codé le backend en Python à l'aide du framework Django. Le backend est composé de quatre applications. L'application principale est l'application *game*, dans laquelle se trouvent les modèles représentant les différents composants du jeu, les fonctions associées aux règles du jeu. La majeure partie de ces modèles peut être communiquée au frontend grâce à l'extension Django REST Framework, qui permet de servir ces modèles par le biais d'une API et de requêtes HTTP.

Voici le diagramme UML représentant la majeure partie des modèles du projet ainsi que leur agencement :

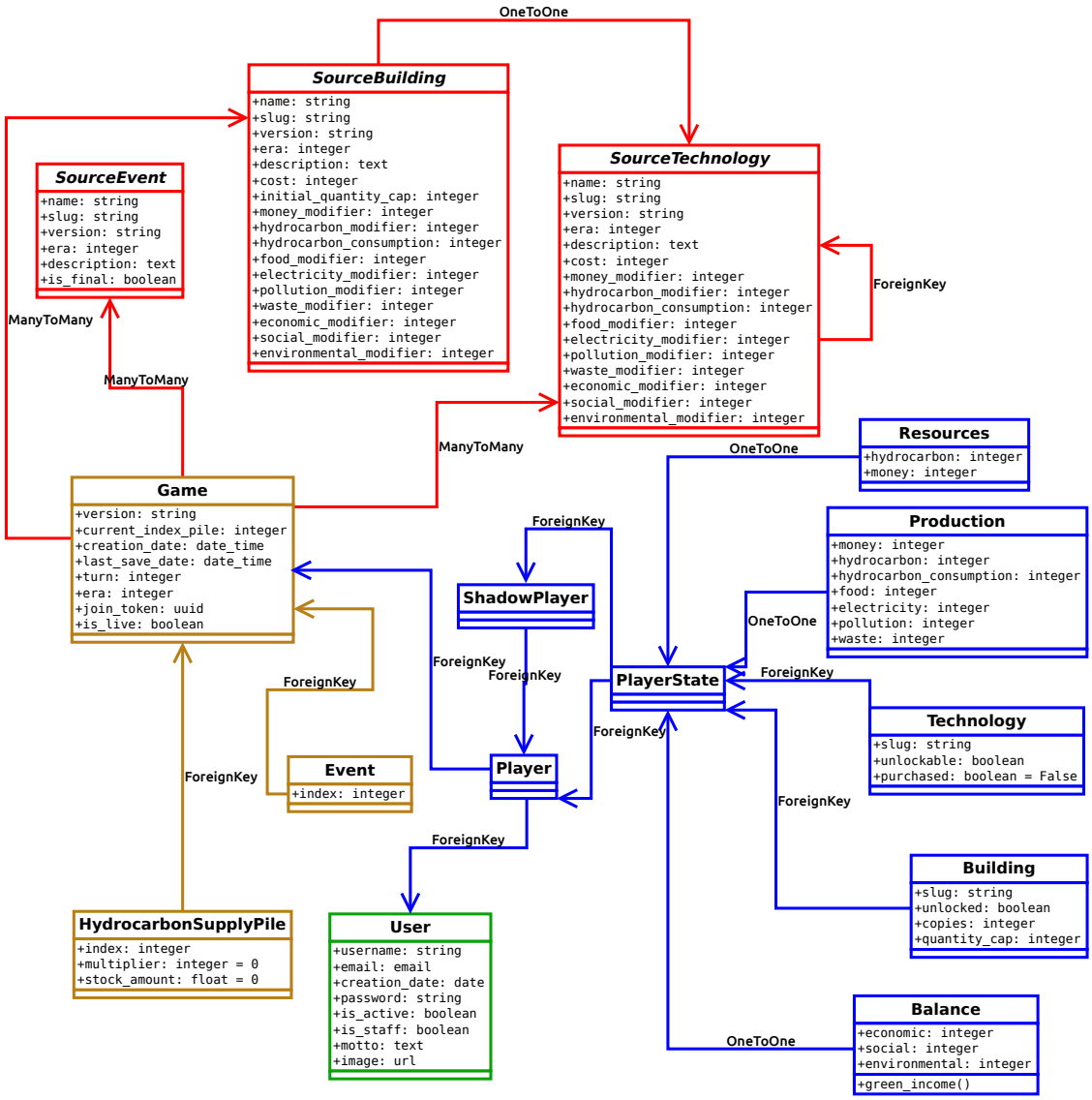

FIGURE 1 – Diagramme UML des modèles

Comme indiqué sur le diagramme, on a tout d'abord une classe centrale **Game**, à laquelle sont reliées les fixtures (écrites dans des fichiers .json puis chargées en tant qu'instances de modèles) qui décrivent les instances de **SourceBuilding** (bâtiments), **SourceTechnology** (technologies), et **SourceEvent** (évènements). Puis, à **Game** sont reliés avec une relation ForeignKey (many-to-one) les modèles **HydrocarbonSupplyPile** (piles d'hydrocarbures, qui sont ordonnées par leur index) et **Event** ("deck" d'évènemements pour cette partie, ordonnés par leur index, et liés en OneToOne aux **SourceEvent**). Le deuxième modèle central de l'architecture du backend est le modèle **Player** qui est relié à **Game** par une ForeignKey. Chaque **Player** possède un **PlayerState**, auquel sont reliés les modèles **Resources**, **Production**, **Balance**, **Technology**, et **Building**. De plus,

chaque **Player** est lié à un **ShadowPlayer**, qui est lui même lié à un **PlayerState**. Cela permet d'implémenter la pile d'action : au début de la phase principale de chaque joueur, on copie le Player dans le ShadowPlayer. Puis, pendant toute la phase les actions effectuée ne modifient que le ShadowPlayer. En cas d'annulation de la pile d'actions, le Player est de nouveau copié dans le ShadowPlayer. Enfin, le Player est mis à jour à la fin du tour en lui attribuant les nouvelles propriétés du ShadowPlayer.

3.2 Frontend React

3.2.1 Généralités

Pour programmer le Front-End de notre projet, nous avons choisi d'utiliser la bibliothèque de JavaScript React. React est fondé sur l'usage d'objets appelés composants qui héritent d'une classe « Component » et qui s'organisent selon une structure arborescente. Chaque composant dispose d'un attribut state, qui peut stocker des données nécessaires au composant lui-même et à ses fils, et d'une méthode render, qui retourne un ensemble d'objets HTML et de composants React générés dynamiquement en utilisant des syntaxes JSX à chaque fois que le composant est mis à jour. Pour les requêtes HTTP assurant la communication entre Front-End et Back-End, nous avons fait usage de la librairie Axios. Le composant principal qui englobe la totalité des autres s'appelle App et se situe à part dans le fichier du même nom. La méthode render de App peut retourner deux composants différents, Game ou WelcomePage, selon que le joueur est dans une partie ou sur les pages d'accueil. WelcomePage retourne l'ensemble des composants nécessaires pour se connecter, créer un compte, créer ou rejoindre une partie et qui effectuent les requêtes correspondantes auprès du Back-End. Ces composants sont codés dans le fichier WelcomePage.js auquel est associé le fichier de styles WelcomePage.css. Game, quant à lui, retourne l'ensemble des composants correspondant à l'interface du jeu en lui-même. Le state de Game contient l'ensemble des données du jeu susceptibles d'évoluer en cours de partie. A quelques rares exceptions près, les calculs concernant la faisabilité des actions par le joueur et la mise à jour des données de la partie sont effectués par le Back-End. Des requêtes HTTP sont déclenchées lors des interactions du joueur avec certains boutons pour garantir les échanges nécessaires avec la base de données. Les composants descendant de Game sont codés dans le fichier Game.js auquel est associé le fichier game.css pour les styles.

3.2.2 Structure de la page d'accueil

WelcomePage peut retourner différents composants correspondant aux différentes étapes entre la connexion et le lancement d'une partie. En toute logique, le composant retourné en premier est Login qui affiche un formulaire permettant à un joueur ayant déjà un compte de se connecter avec son nom d'utilisateur et son mot de passe. Un second composant Signup est accessible depuis Login et affiche un autre formulaire permettant à un joueur n'ayant pas de compte de s'en créer un. Si l'un des deux questionnaires est rempli correctement et envoyé, l'utilisateur accède au composant MainMenu qui propose de rejoindre

une partie au moyen d'un code ou d'en créer une. Finalement, si le joueur choisit de créer une partie, un quatrième composant CreateGame est affiché qui permet au joueur de créer une partie et de se procurer le code de partie correspondant communiqué par le Back-End. Ce code de partie que nous avons déjà évoqué plus haut est nécessaire à d'autres joueurs pour se connecter à la même partie. Que ce soit par création d'une nouvelle partie ou en en rejoignant une déjà existante, il s'opère un changement dans le composant retourné par App qui devient celui de l'interface du jeu : Game.

3.2.3 Structure de l'interface du jeu

Le composant Game peut retourner trois sous-composants différents, chacun correspondant à un onglet de l'interface. MenuGrid correspond à l'onglet « Menu », sa méthode render retourne des composants affichant les statistiques et ressources du joueur, le menu des actions pouvant être achetées et la file des actions effectuées pendant le tour. TechGrid correspond à l'onglet « Technologies » et permet d'afficher le menu des technologies, toujours avec la file des actions du tour. Finalement, GameInfoPanel devrait, à terme, afficher des statistiques globales du jeu, des évènements et d'autres informations utiles telles que le code de la partie, mais cette page est pour l'instant vide.

4 Eléments réalisés

(il est usuel d'avoir un delta avec les objectifs initiaux)

5 Problèmes rencontrés

(afin que nous capitalisions de l'expérience en vue des prochaines promos)