Rapport du projet web

Réalisation d'une application de découverte de la théorie des automates pour les plus jeunes

Thomas Meyer*- El Mehdi Kossir†- Romain Beuzelin‡

Année 2019-2020
Semestre 2
UE PROJ

Mai 2020

Table des matières

*thomasmeyer@outlook.fr
†m.kossir@outlook.fr
‡romain.beuzelin@ensiie.fr

1 Introduction

L'idée directrice de notre projet Web est de proposer une plateforme en ligne qui permet aux plus jeunes - et moins jeunes - de découvrir une facette de l'informatique intéressante et utile dans nos vies : la théorie des automates.

Remarque : Nous avons choisi le JavaScript pour développer les challenges..

2 La base de données

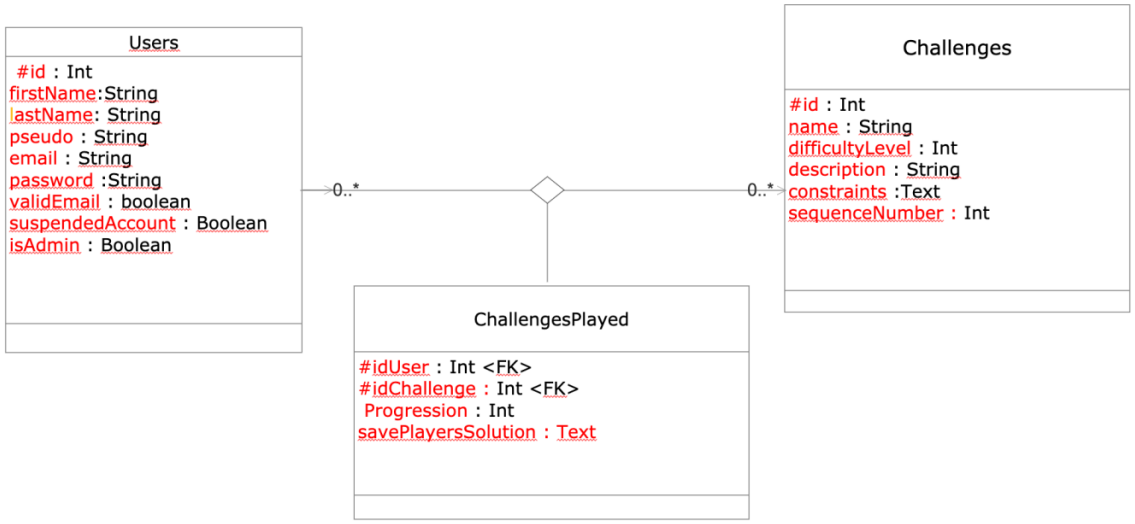

FIGURE 1 – Schéma UML de la base de donnée du site

Une telle base de données permet d'associer rapidement un utilisateur aux challenges qu'il a joué. Les tables sont déclarées dans PostgreSQL à partir du fichier `data/init.sql`.

La table `Users` contient toutes les informations liées aux utilisateurs. La table `Challenges` contient les challenges, leurs difficultés, ainsi que leurs identifiants. Enfin, la table `ChallengesPlayed` met en relation les deux tables ci-dessus en associant l'identifiant d'un utilisateur à l'identifiant d'un challenge, une fois que le niveau en question a été joué. C'est une table de type `n..n`.

3 L'organisation du code

On adopte l'architecture « MVC » dans ce projet. Les « Contrôleurs » sont les fichiers `.php` dans `public/` ; les « Vues » sont dans `src/View/` ; les « Modèles » sont dans `src/Entity` avec les « Entités » et dans `src/Repository` avec les systèmes de transactions entre la base de données et l'application.

- `/src` contient le code source principal.
  - `/View` contient les vues de l'application : `/layout.php` contient le cadre HTML des pages et les autres fichiers correspondent aux différentes vues.
  - `/Service` contient des services comme le système de gestion de la sécurité.
  - `/Repository` contient les classes pour gérer les interactions entre l'application et les tables de la base de données.
  - `/Entity` contient les différentes entités de l'application.
  - `AbstractController.php` propose une base pour faciliter l'écriture des contrôleurs de `/public`.
- `/public` contient les ressources accessibles directement depuis l'extérieur du serveur, dont les contrôleurs qui sont en `.php`.
- `/data` contient les instructions qui initialisent la base de données.
- `/config` contient les configurations de l'application.

Une fois l'utilisateur connecté, un menu simple est en haut de la page :
- « OTOMATE » qui permet de retourner à `index.php`.
- « Progression » qui permet d'accéder aux challenges et d'afficher une vue d'ensemble de la progression de l'utilisateur - `home.php`.
- « Profil » qui permet de voir et modifier son propre profil - `profil.php`.
- « Joueurs » qui affiche la liste de tous les joueurs, ainsi que leurs scores - `users.php`.
- « Me déconnecter » qui permet à l'utilisateur de se déconnecter - `signout.php`.

4 Fonctionnalités pour les utilisateurs

On a deux types d'utilisateurs : **les joueurs** et **les administrateurs** qui sont aussi des joueurs. Les administrateurs ont accès à des fonctionnalités supplémentaires.

On propose tout d'abord à l'utilisateur de se connecter. Pour vérifier que le login et le mot de passe sont les bons, on utilise `Security.php` afin de le vérifier dans la base de données. Si c'est le cas, on stocke dans une session les informations de l'utilisateur connecté et on redirige vers `home.php`.

Si un utilisateur tente d'accéder à une des pages `home.php`, `profil.php` ou `challenge.php` sans être connecté, on affiche un message d'erreur indiquant qu'il est nécessaire de se connecter pour accéder à ces pages.

4.1 Fonctionnalités pour les joueurs

Un joueur connecté peut accéder à son profil via l'onglet « Profil », ce qui lui permet de modifier les différents champs de son compte. Il peut aussi, s'il le souhaite, supprimer son compte ou sa progression depuis le début. Le fichier `profil.php` contient le nécessaire pour permettre cela.

Les joueurs peuvent également afficher la liste de tous les membres inscrits du site via l'onglet « Joueurs ». Les différents joueurs sont classés selon leurs noms, mais on peut aussi comparer les scores.

Enfin, les joueurs peuvent accéder à l'onglet « Progression » qui affiche les différents challenges, leurs difficultés et surtout leurs progressions.

4.2 Fonctionnalités pour les administrateurs

En plus des droits des joueurs, les administrateurs disposent des fonctionnalités suivantes sur la page `profil.php` :

— « Suspendre » (pour suspendre un compte joueur).
— « Rendre administrateur » (pour rajouter un nouvel administrateur).
— « Supprimer » (pour supprimer un compte joueur).
— « Retirer la suspension » pour un compte joueur.
— « Retirer l'administration » pour un compte administrateur.

Ces actions sont gérées par le fichier `actionUsers.php`.

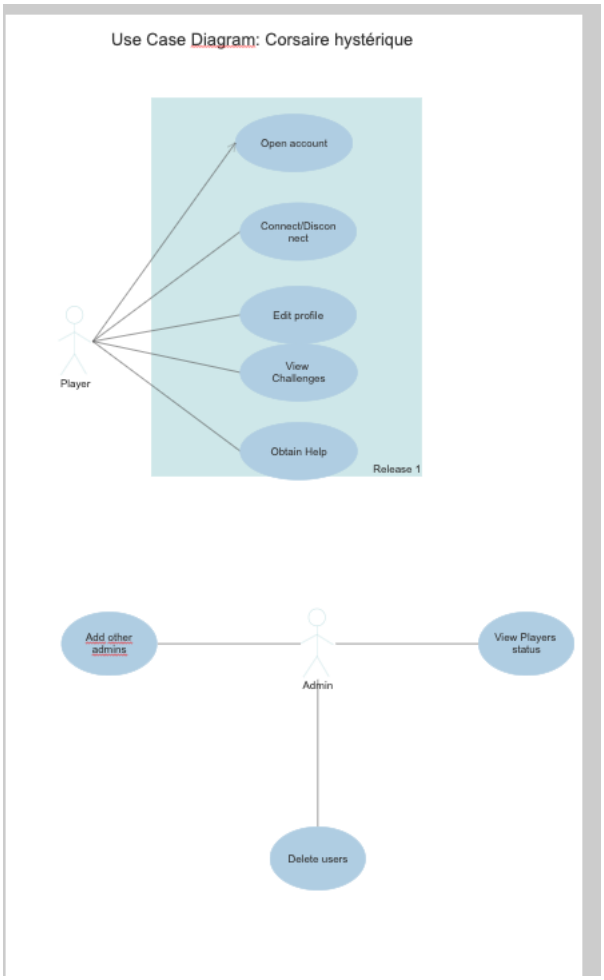

FIGURE 2 – Diagramme de cas d'utilisation pour les utilisateurs.

5 Front-end

5.1 Le design

Nous avons opté pour l'utilisation de la librairie CSS Bootstrap, ce qui nous offre un accélérateur pour le design de l'interface, ainsi qu'une occasion de prendre en main ce classique du développement Web. Le style sombre provient du site Bootswatch. Ce style donne un aspect plus moderne et attrayant. Toutes les illustrations et dénominations proviennent nous sont propres.

5.2 Les challenges

Lors d'un challenge, le joueur doit faire en sorte que son automate satisfasse plusieurs séquences de couleurs, par exemple :

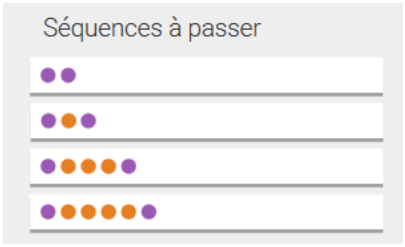

FIGURE 3 – Un exemple d'un challenge avec quatre séquences. On a choisi d'utiliser des couleurs pour rendre le jeu plus visuel et ludique.

Pour satisfaire la séquence, le joueur doit attribuer des pastilles de couleurs aux transitions entre les états.

Le jeu est entièrement implémenté en JavaScript et on utilise le tag `<canvas>` pour l'affichage. Comme prendre en main cet objet est particulièrement délicat, on utilise la bibliothèque JavaScript Konva qui apporte de nombreuses facilités et une abstraction utile : manipulation d'objets basiques (cercle ou flèche), gestion des « Drag&Drop » et des évènements sur la scène.

L'ensemble du code source qui gère le jeu se trouve dans le fichier `public/js/script.js` et est largement commenté pour en décrire le fonctionnement.

5.3 Problèmes rencontrés

Au début, il a été envisagé de proposer aux joueurs de manipuler eux-même le graphe de l'automate : ajouter ou retirer des états ou des arcs, déplacer les sommets. Cela a représenté une grande tâche pour gérer la cohésion des éléments entre eux : déplacer un sommet suppose de déplacer également les arcs qui lui sont liés. Cette gestion n'est pas assurée par Konva.

Finalement, le jeu n'était pas encore fonctionnel - pas de système de sélection des symboles sur un arc ou absence de gestion de tout les types d'arcs. Et il était particulièrement difficile de le rendre fonctionnel dans le temps imparti en poursuivant dans cette direction. Donc, il a été décidé de simplifier l'approche en fixant le graphe de base : le challenge propose un « fond » d'automate à remplir comme un « fond de carte ».

Cependant, la version antérieure est gardée comme une perspective d'évolution et est présentée sur la page `challenge.dev.php`.

6 Back-end

6.1 Contrôleurs

Les contrôleurs sont les fichiers `.php` dans `public/`. Chacun de ces fichiers contient une classe anonyme qui étend la classe abstraite `AbstractController.php` de `src/`. Cette classe abstraite facilite et centralise les actions communes et courantes :

- Gérer les différentes méthodes `GET` ou `POST`. Il faut définir les comportements du contrôleur dans une fonction `get` ou `post`. C'est la classe abstraite qui s'occupe d'appeler la bonne fonction.
- Gérer la création de l'accès à la base de données.
- Gérer le système de sécurité en instanciant la classe `Security.php`.
- Gérer le rendu des vues en spécifiant le nom de la vue qui est dans `src/View/`.
- Gérer la récupération des paramètres `GET` ou `POST`.

6.2 Security

La classe `Security.php` dans `src/Service/` permet de fournir une abstraction à la gestion de la sécurité. Elle propose de gérer la connexion et la déconnexion des utilisateurs, de savoir si et quel utilisateur est actuellement connecté, de savoir si l'utilisateur est un administrateur, et de gérer les données utilisateurs mémorisées dans la session< ;

6.3 Entities et Repositories

On trouve dans les fichiers de `src/Entity/` les classes représentant les entités du modèle. On y trouve des getters et setters pour gérer les attributs.

Dans les fichiers de `src/Repository/`, on trouve des classes qui proposents des opérations pour interagir avec la base de données : ajouter, récupérer, modifier ou supprimer une entité (CRUD) ou réaliser des opérations plus spécifiques.

7 Répartition des rôles

Le projet est versionné grâce à Git et hébergé sur GitLab. Nous avons décidé d'utiliser Discord pour organiser les tâches à effectuer et par qui elles doivent l'être.

8 Conclusion

Ce projet s'est révélé très enrichissant dans la mesure où il a consisté en une approche concrète du métier d'ingénieur. En effet, la prise d'initiative, le respect des délais et le travail en équipe seront des aspects essentiels de notre futur métier.

De plus il nous a permis de stimuler notre créativité et de trouver des idées originales que nous avons essayé de mettre en œuvre.

Le sujet que nous avions choisi était très intéressant à traiter, puisque l'informatique aujourd'hui occupe une place de plus en plus importante dans notre quotidien. Il est donc indispensable que ce domaine soit présentée de manière ludique et concrète aux enfants pour leur donner envie de s'y intéresser et de s'investir.