Rapport de Projet

SpeedRunners

Outil communautaire pour le speedrun

Réalisé par

Hugo Fazio

Encadré par

Anne Laurent et Arnaud Castelltort

INTRODUCTION

Ce projet est réalisé dans le cadre de mon cursus d'études Informatique et Gestion à Polytech Montpellier. Il a pour but le développement d'une application web complète.

Cette application sera axée sur le **speedrun**, une discipline e-sport qui consiste à terminer un jeu vidéo le plus rapidement possible. Un essai chronométré est appelé une **run**. A cela peuvent s'ajouter des contraintes, telles que la nécessité de battre tous les boss du jeu, ou encore de finir le jeu à 100%.

Ce sujet m'intéresse particulièrement car j'ai débuté le speedrun sur quelques jeux récemment, et j'apprécie cette nouvelle approche du jeu vidéo. De plus, cela me permet de me positionner en également en tant qu'utilisateur, et ainsi mieux cerner leurs besoins. Enfin, cet un outil dont je peux me servir directement, et donc songer à des développements futurs ainsi qu'une maintenance.

SpeedRunners est donc une application web permettant le **chronométrage** et le traitement des résultats des runs, mais se voulant également **communautaire**, avec un axe de développement autour du partage de ressources et conseils entre les membres.

Sommaire

Cahier des Charges

ANALYSE DU CONTEXTE

Le speedrun est une discipline consistant originellement à déclencher les crédits de fin d'un jeu vidéo. Il est possible d'y ajouter des règles supplémentaires, créant différentes catégories de runs par jeu.

Il existe déjà de nombreuses applications locales de chronométrage disponibles, ainsi que communautés de runners sur des sites web. Le coeur de ce projet est donc la conception d'une application web reliant ces deux aspects du speedrun.

De plus, la plupart des échanges autour du sujet se faisant en ligne, l'aspect communautaire pourrait servir de partage de connaissances, et non seulement de données.

Les besoins en fonctionnalités sont donc multiples et très variées, du module de chronométrage aux partages des données de runs, ils nécessitent une conception minutieuse.

BESOINS FONCTIONNELS

Tout d'abord, pour assurer un suivi complet des statistiques de chaque utilisateur, une authentification est requise. Ainsi, quelque soit le lieu de connexion, le runner a accès à ses données de manière sécurisée, peut réaliser un temps, et le sauvegarder. Un module d'inscription est évidemment nécessaire pour les nouveaux utilisateurs.

Le module essentiel de cette application sera évidemment le chronométrage, qui se doit donc d'être intuitif et facile d'utilisation, tout en assurant la fiabilité des temps réalisés.

Un essai requiert l'utilisation de splits (créneaux de découpage de la run) qui servent de points de repères temporels intermédiaires pour le runner. Les classements sont donc organisés par categorie, mais chacun est libre d'utiliser les repères qu'il souhaite. L'utilisateur doit également pouvoir sélectionner les jeux et les catégories associées qu'il souhaite runner, et les ajouter s'ils ne sont pas encore présents dans l'appli web.

Il semble intéressant de proposer à l'utilisateur des outils d'analyse de ses statistiques, l'incitant à la progression. Ces développements pourront donc aussi se pencher sur la représentation graphique de ses résultats, et la possibilité d'y fixer des objectifs.

Dossier d'Architecture Technique

ARCHITECTURE GLOBALE & TECHNOLOGIES

SpeedRunners doit donc être une application particulièrement stable et *responsive*, avec un contrôle important des informations pour le client. Souhaitant des échanges avec le serveur limités, l'application sera donc principalement orientée autour de JavaScript pour le client et son navigateur web, mais aussi pour le serveur ! En effet, de nombreuses technologies utilisant Javascript pour la création apparaissent, et elles permettent une gestion très dynamique des sites web.

L'API Serveur sera donc réalisée à l'aide de NodeJs, qui se base sur le V8 Engine de Google Chrome. Ce moteur d'exécution permet un traitement extrêmement rapide du JavaScript, notamment grâce à une compilation Just In Time. De plus, l'exécution dite non-bloquante de NodeJS permet une programmation asynchrone et une nouvelle optimisation des temps de traitement. Ce serveur sera réalisé à l'aide du framework Express développé pour NodeJS, qui assure des facilités de programmation et qui est particulièrement bien documenté.

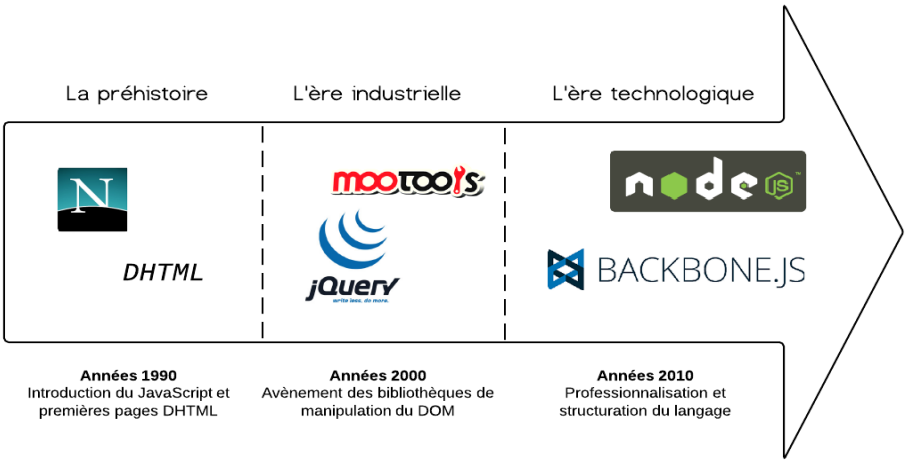


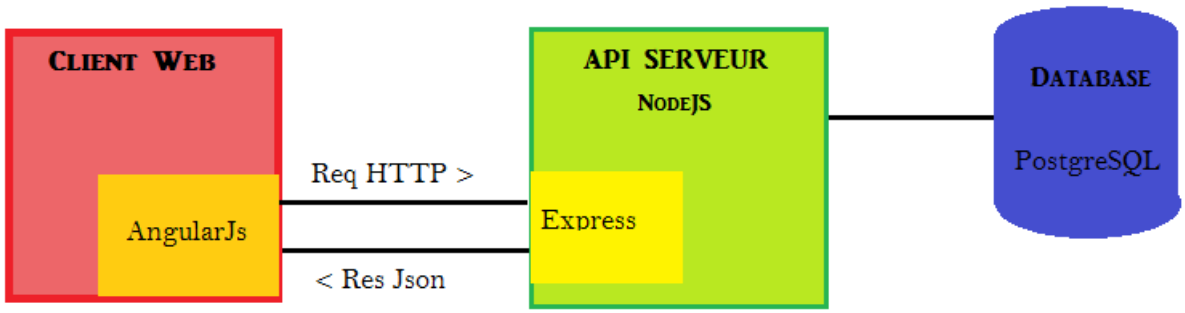
Figure 1 : Évolutions de l'utilisation de JavaScript

source : www.openclassrooms.com

Concernant le frontend, la navigation et le contrôle des données client seront assurés par le framework AngularJS v1.6.4. Initialement développé dans le but de l'inclure dans une architecture MVC (Model View Controller), il permet en réalité une conception MVW (Model View Whatever).
En effet, il peut dépasser son rôle de contrôleur et assurer une communication avec n'importe quel motif d'architecture MV.

Enfin, le serveur communiquera avec le SGBD Relationnel PostgreSQL v9.6, car ce dernier assure un respect fiable des contraintes d'intégrités, et permet le traitement de très grands nombres de données, ce qui est bon d'envisager lors du traitement statistique des résultats.

Ces technologies de base nous permettent de dégager l'architecture suivante.



API SERVEUR

Le développement de l'API se réalise donc grâce à NodeJs, via le framework Express.

La réception d'une requête HTTP (GET,POST,UPDATE...) entraîne son routage dans le serveur qui abouti à la transmission d'une requête SQL vers PostgreSQL. Ces étapes sont factorisées pour permettre une fluidité de suivi de la demande. Les données récupérées (ou non) sont alors traitées et retournées au format Json, qui garanti un volume de données réduit et un retour rapide.

En cas de succès (code retour HTTP 20x), les informations sont retournées au client. Sinon, le code de retour 40x ou 50x est envoyé avec le message d'erreur associé.

Le serveur assure également la limitation à l'accès aux données lorsque celles ci requièrent une connexion (ou inscription) du membre. Ce maintient de l'authentification est géré par le module JWT-simple (JsonWebToken) de Node, qui délivre puis vérifie à chaque accès la présence et la validité du token envoyé par l'utilisateur.

La connexion à PostgreSQL est quand à elle assurée par le module pg, qui apporte une simplicité d'accès au SGBDR. Concernant le cryptage de données sensibles, le module crypt-js gère codage et encodage à la réception d'une requête.

AngularJS assure donc la navigation du client à travers le vues avec la directive ng-Route, mais nous permet surtout l'utilisation d'un pattern d'architecture plus abouti que le MVC, car il se base particulièrement sur la gestion des événements qui l'amène à modifier les données de la vue ou à communiquer avec le serveur.

Ainsi, il limite grandement les accès au serveur, permettant un maximum de traitement d'informations délivré rapidement à l'utilisateur.

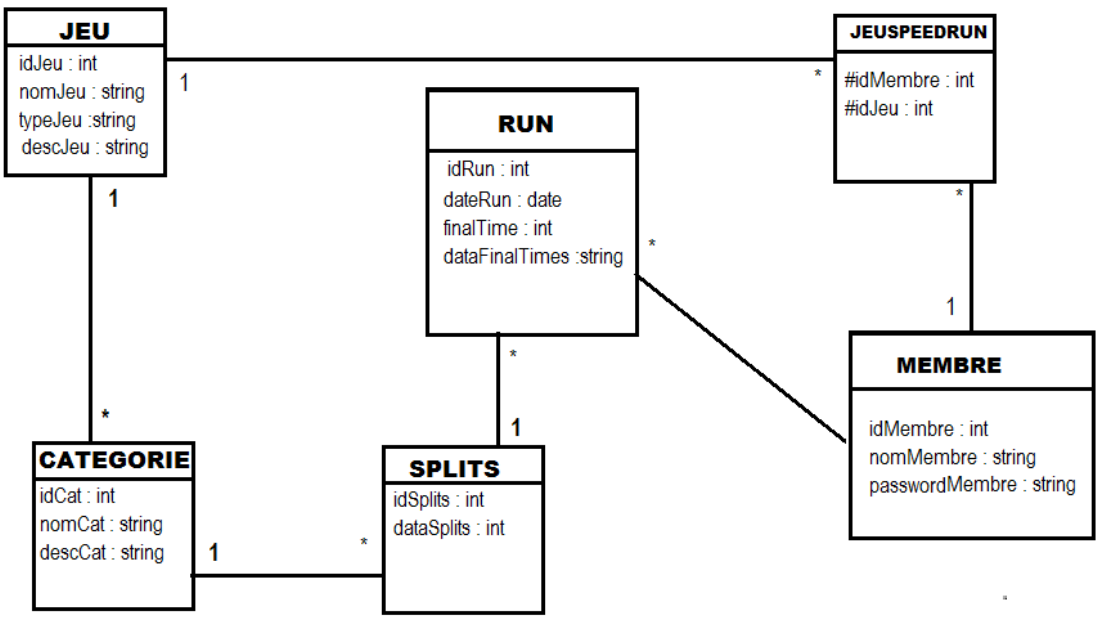
Le web design, réalisé en CSS, a principalement été réalisé directement par des feuilles de style sans framework CSS, pour conserver une séparation optimale entre CSS et HTML. Cependant, quelques détails, les boutons par exemple, sont obtenus grâce au framework Materialize.

Le SGBD PostgreSQL a donc été préféré à MySQL pour sa capacité de traitement de grandes plages d'information et sa vérification constante des contraintes d'intégrité, qui le rende plus abordable.

La création de la base de donnée est partie de l'analyse des besoins en données pour pouvoir correctement traiter les résultats par la suite.

Ainsi, le résultat d'une run doit être relié à l'utilisateur qui l'a réalisé, au jeu, à la catégorie de speedrun sur ce jeu, à un découpage temporel (splits) et l'heure à laquelle est réalisée ce speedrun. Pour lui associer automatiquement un découpage « basique » (d'une seule partie), un trigger a été créé à la création d'une nouvelle catégorie.

J'ai alors abouti au diagramme de classes suivant (voir Scripts BD sur le git hub)



DÉPLOIEMENT WEB

La mise en ligne de l'application web est effectuée par Heroku, un service de cloud-computing en PaaS (Platform as a Service). Il est donc responsable de la maintenance des outils de déploiement et de gestion des serveurs physiques via AWS (Amazon Web Service), les applications sont quand à elles bien sur gérées par les utilisateurs d'Heroku.

Ce PaaS est, en plus de pouvoir déployer gratuitement ses applications (sous limites), l'un des plus intuitifs et simple d'utilisation. De plus, il permet la création et le contrôle (relatif) d'une base de données postgres directement liée à l'application depuis Heroku.

Cette database est également hébergé via les services d'Amazon AWS, aux Etats Unis.

Fonctionnalités

FONCTIONNALITÉS DÉVELOPPÉES

L'application est disponible ici : https://w3bproject.herokuapp.com/

Pour un utilisateur lambda :

Inscription (Vous êtes invités à vous inscrire)

Pour un membre :

Identification et maintient de la connexion

Accès aux jeux disponibles ou sélectionnes

Ajout d'un jeu
Ajout d'une catégorie de speedrun
Ajout de splits pour une catégorie

Sélection d'une catégorie pour speedrun

Chronométrage d'une run et des intermédiaires
Sauvegarde du temps final et des temps intemédiares
Récupération du record du joueur pour une catégorie

DÉVELOPPEMENTS FUTURS

Ayant réalisé la plupart des fonctionnalités que j'espérais présentes, je pense que l'évolution de cette application passera par le développement d'outils de traitement statistiques.

Les membres pourraient donc comparer leurs temps, obtenir des graphiques sur leurs résultats ou encore fixer des objectifs de temps à atteindre.

Rapport d’activité

GESTION DE PROJET

J’ai voulu définir rapidement les technologies à utiliser, pour pouvoir les préparer et les faire fonctionner entre elles en amont. Je me suis donc basé sur des technologies que j’avais déjà abordé (PostgreSQL, NodeJS) pour y ajouter des nouveautés (AngularJS, JWT-simple).

Puis il a fallu établir clairement et de manière cohérente les modules à développer, en conservant des limites liées aux contraintes de temps de réalisation.

Je me suis concentré en premier lieu sur la réalisation de l’authentification, pour m’assurer rapidement d’avoir une application utilisable et stable. Puis le routage a permis de délimiter des contours clairs à l’application.

J’ai réussi à conserver une bonne stabilité du site lors du développement en réalisant les modules un par un, en m’assurant constamment que cela ne pouvait influencer les réalisations précédentes.

OUTILS UTILISÉS & CRÉDITS :

Pour m'assister dans la réalisation de ce projet, j'ai été amené à utiliser plusieurs outils de développement, ainsi que des ressources open source que j'aimerais citer.

Postman: Application permettant d'envoyer des requêtes à un serveur en paramétrant facilement les headers et les données de la requête, et de voir ce que le serveur retourne.

pgAdmin : Interface d'administration et de développement open source pour les bases de données postgreSQL.

www.flaticon.com : Banque de données d'icônes
by Dave Gandy

Conclusion

Je pense avoir tirer profit du fait que le sujet m'intéressait et qu'il pouvait m'apporter une utilité directe.

La bonne gestion de mon temps dans mon projet m'a permis de prévoir et anticiper les échéances de l'avancée de mon projet. Ainsi, la plupart des fonctionnalités désirées ont vu le jour, permettant à l'application d'être fonctionnelle et réellement utilisable.