



DATPLAYLIST

Projet Informatique



20 DECEMBRE 2015

NOËL DO VAN

Table des matières

1.	Présentation	2
1.1.	Contexte.....	2
1.2.	Description	2
2.	Préparation du projet.....	3
2.1.	Objectifs.....	3
2.2.	Langage et environnement.....	4
3.	Le développement.....	4
3.1.	Les étapes	4
3.2.	Les défis rencontrés	5
3.3.	Résultat	6
4.	Conclusion	9

1. Présentation

1.1. Contexte

Le projet DatPlaylist a été entièrement développé par moi-même. L'idée m'était venu alors que des vidéos (musiques) hébergées sur le site Youtube aient été supprimé alors que je l'ai avait préalablement mise dans différentes playlists. Leurs suppressions avaient provoqué des éléments manquants dans mes playlists et, je devais alors chercher une autre vidéo manuellement afin de la remplacer. Cette vidéo-là était souvent identique à celle supprimée précédemment par Youtube. La différence était juste qu'elle était hébergée par un autre utilisateur.

A partir de ce point-là, j'ai remarqué que tout ce que je faisais à chaque fois était de rechercher ma vidéo avec l'outil de recherche Youtube, prendre le premier résultat et, remplacer la vidéo supprimée de ma playlist par le nouvel élément.

J'ai alors eu l'idée de créer un utilitaire qui permettrait de lire des playlists à partir de chaînes de caractères (strings) à chaque fois que l'on lancerait la playlist. Ainsi, il ne pourra plus avoir de vidéo supprimée car une recherche sera faite à chaque lecture.

Durant cette période, je voulais également apprendre le langage de programmation Ruby. J'ai alors profité de réaliser cette utilitaire avec le framework Ruby On Rails afin de rajouter un défi supplémentaire au développement de ce projet informatique.

1.2. Description

DatPlaylist est un utilitaire sous forme de site web. Sa principale fonctionnalité est de lire des playlists basées sur des strings. Voici un processus d'utilisation :

- l'utilisateur accède au site DatPlaylist
- il crée sa playlist en rentrant ses strings dans une zone de texte simpliste
- il accède à sa playlist alors hébergée sur le site
- ses vidéos sont alors jouées

Une petite parenthèse, j'ai voulu que cette utilitaire soit simple et facile d'accès. Il n'y a pas de système d'authentification et compte utilisateur, toutes les personnes entrant sur le site peuvent créer une playlist, lire une playlist et, toutes les playlists créées sur le site sont publiques. Ce point-là pourra être modifié dans le futur.

Le site web est divisé en quelques pages :

- la page d'accueil : permet de créer une playlist
- la page de listage : permet de voir toutes les playlists créées sur le site
- la page d'édition : permet d'éditer les playlists
- la page de lecture : permet de lire une playlist
- la page de F.A.Q : la page « à propos » du site

2. Préparation du projet

2.1. Objectifs

Mes objectifs au début du projet étaient de :

- réaliser un site avec un langage/framework inconnu (Ruby on Rails)
- réussir à utiliser l'api de Google afin de faire des recherches sur Youtube
- réussir à stocker les playlists créées par les utilisateurs
- réussir à lire les vidéos les unes à la suite. C'est-à-dire que lorsqu'une vidéo se termine, le site devra automatiquement charger la vidéo suivante sans que l'utilisateur ait à faire quoi que ce soit.

D'autres idées et objectifs se sont alors rajoutés au fur et à mesure que le développement avançait comme :

- permettre à l'utilisateur de se créer un compte. Cela aurait permis d'implémenter les fonctionnalités suivantes :
 - o avoir des playlists privées
 - o système de « upvote/downvote » sur les playlists publiques
 - o édition de playlist créé par l'utilisateur
- vider la liste de playlist publique périodiquement (tous les mois par exemple) afin d'éviter l'encombrement. Ceci engendrerait les fonctionnalités suivantes :
 - o sauvegarde des playlists de chaque période dans une autre page afin d'en garder une trace
 - o historique sur les statistiques des playlists publiques les plus écoutées/les plus upvotée de chaque période
 - o système de badge/achievement/récompense pour les utilisateurs
- permettre de rajouter des options à ses strings avant la création de la playlist. Par exemple, l'utilisateur pourrait vouloir que le site recherche son string sur Youtube avec l'option « Vidéo la plus vue » au lieu de « Vidéo la plus pertinente » qui est l'option par défaut.
- la suppression de playlist avec deux systèmes sont envisageables :
 - o supprimer les playlists créées non anonymement depuis la page de l'utilisateur (non implémenté car pas de compte utilisateur)
 - o supprimer les playlists créées anonymement avec un code généré aléatoirement et, donnée à l'anonyme lors de la création de sa playlist

2.2. Langage et environnement

Le côté back end du projet a été développé en Ruby à l'aide du framework Ruby on Rails. Le côté front end du projet utilise les technologies HTML/CSS/JQuery ainsi que l'api Javascript de Google.

Pour ce qui est de l'environnement de développement, j'ai utilisé une machine virtuelle sous l'OS Debian car je trouvais que c'était plus simple de faire du Ruby sur Linux que sur Windows.

Afin d'héberger le site en ligne, j'ai loué un VPS ainsi qu'un nom de domaine. Cette partie-là fut également un autre défi pour moi car c'était la première fois que je configurais un serveur de type site web de A à Z. Le site est actuellement accessible à l'adresse suivante : datplaylist.com

Les technologies ssh/ftp ont alors été utilisées afin de transférer et communiquer avec le serveur.

3. Le développement

3.1. Les étapes

Voici le déroulement de la mise en place de l'environnement :

- mise en place d'une machine virtuelle afin de développer du Ruby sur un environnement Linux
- mise en place d'un serveur VPS
- configuration du serveur (installation de ruby, rails, ssh, ftp, etc) et de son nom de domaine
- inscription du projet sur la console de développement Google afin d'avoir la clé API qui servira à faire des recherches sur Youtube

Voici le déroulement du développement en lui-même :

- j'ai premièrement essayé de faire marcher l'api de Google. Une seule page avec une zone de texte et un bouton était nécessaire. Mon objectif était de récupérer la réponse de Youtube par rapport à la requête présente dans la zone de texte quand on appuyait sur le bouton
- deuxièmement, il fallait pouvoir stocker la requête en base de données (fonctionnalité qui allait créer les playlists). J'ai donc utilisé une base de données sqlite
- ensuite, l'objectif fut de lister toutes les données dans une page (cela deviendra la page de listage)
- à partir de ce point-là, seule une requête était stockée dans les tables, hors dans une playlist, il y a plusieurs strings. J'ai donc modifié la manière dont j'envoyais la requête du front end au back end (je vais aborder ce point dans la section suivante.)
- une fois le problème réglé, il fallait travailler l'interface du site. J'ai alors implémenté Bootstrap grâce à l'API Rails
- ensuite, j'ai fait la page de lecture de playlist qui chargerait les vidéos automatiquement
- finalement, il restait la page d'édition/de suppression à implémenter

3.2. Les défis rencontrés

J'ai rencontré beaucoup de défis durant ce projet. La principale étant l'apprentissage du langage Ruby et de son framework Ruby on Rails. La seconde fut de configurer un serveur de A à Z. L'installation du Ruby on Rails sur un serveur neutre Debian ne fut pas très simple également. Cependant, après plusieurs tutoriels suivis sur internet, j'ai finalement réussi à faire marcher le tout. Voici les défis rencontrés lors de la phase de développement :

- trouver une solution à propos de, où se placerait l'API de recherche Google. En effet, je ne savais pas si le mieux était de récupérer l'api côté client ou l'api côté serveur. J'ai choisi l'API côté client. C'est-à-dire que je récupérais une clé API qui servira à faire des requêtes Youtube depuis le navigateur de l'utilisateur et non depuis le serveur
- comment j'allais transférer les données du front end à la base de données. J'ai premièrement développé le tout en utilisant la technologie Ajax qui faisait une requête sur une page côté serveur qui ajoutait les données à la base de données, mais le résultat n'était pas concluant lorsque j'avais plusieurs champs. En effet, je devais tout concaténer et tout reparser côté serveur, cela devenait trop compliqué. J'ai donc utilisé les formulaires POST afin de passer en argument plusieurs paramètres
- je me suis aperçue qu'en voulant faire un utilitaire très simpliste sans compte connecté pour l'instant, il était très facile d'ajouter des playlists. Il suffisait juste de remplir la zone de texte et d'appuyer sur le bouton. Le problème du spam m'est alors venu à l'esprit. Afin de prévenir des bots qui spammeraient la création de playlist, j'ai implémenté un système de captcha grâce à la technologie Google ReCaptcha (première fois également que je l'utilisais.). Il faut alors passer le test afin de détecter si on est un robot ou pas.
- une autre difficulté fut de faire en sorte que lorsque l'on est sur une page de lecture de playlist, la lecture se fasse normalement. Lors de mes premiers tests, lorsque l'on arrivait sur la page, une fois sur deux la vidéo ne se lançait pas car l'API Google se chargeait après le chargement de la page. Le résultat était totalement aléatoire. J'ai essayé plusieurs solutions comme réactualiser la page si le plugin ne s'était pas chargé correctement par exemple mais ça ne résolvait pas le problème. Après de longs tests, j'ai donc implémenté tout un système de vérification du plugin et, de rechargement de la page afin de contourner ce problème.
- une autre problématique fut de charger la vidéo suivante lors de la fin d'une vidéo sans déclencher le problème précédent. En effet, lorsque la vidéo se terminait, je chargeais la vidéo suivante. Cependant l'api Google ne se rechargeait pas correctement avec cette manière-là. La solution était de directement charger le lien de la vidéo suivante grâce à la technologie Javascript. En faisant ceci, le navigateur recharge complètement la page et, l'API Google se recharge correctement.
- au moment où j'implémentais la page d'édition/de suppression, un nouveau problème était survenu. Pour lire une playlist, il fallait accéder à sa page. Son lien était formaté comme suit : « play/1/0 » pour la playlist numéro 1. Cependant, après avoir supprimé une playlist, les numéros changeaient également et donc, la dernière donnait vers un lien mort, cela décalait tout. J'ai alors revu le système de linkage, le lien d'une playlist est maintenant formaté comme ceci : « play/CNPYCANKHKD/0 » afin d'éviter ce problème.

3.3. Résultat

Le site est actuellement disponible à l'adresse suivante : <http://www.datplaylist.com/>

Voici quelques captures illustrant l'idée générale du site.

Page d'accueil et création de playlist

The screenshot shows the 'Dat Playlist' website interface. At the top, there's a navigation bar with 'Dat Playlist', 'Create', 'Edit', 'View All', and 'FAQ'. The main header is blue with the title 'Dat Playlist' and the tagline 'Helps you to create string-based playlists without the constraint of keeping it up to date. Works with Google YouTube Search API.' Below this is a large text input area containing several lines of text: 'super meat boy hot damned', 'hotline miami daisuke', 'athletic theme', 'hotline miami perturbator', and 'oot lost woods'. To the right of the text input are two optional fields: 'Author' and 'Description', each with an 'Optional' label. Below these is a CAPTCHA section with a green checkmark, the text 'Je ne suis pas un robot', and the CAPTCHA logo. At the bottom right is a blue button labeled 'Create a playlist'. The footer contains the text 'Designed and developed by bijl' and '2015 © Dat Playlist'.

L'utilisateur entrera une requête par ligne. Il spécifiera un nom d'auteur et une description s'il le souhaite. Il vérifiera auprès du site s'il n'est pas un robot et il pourra créer sa playlist.

Page de playlists publiques

Dat Playlist Create Edit View All FAQ

Public Playlists

The code of your playlist is **JBOETSHN** (datplaylist.com/play/CNPYCANHKHD/0). Copy it somewhere if you plan to edit / delete your playlist, or leave it here !

Anonymous Jan 10, 2016 20:34 UTC		Playlist unnamed super meat boy hot damned hotline miami daisuke athletic theme hotline miami perturbator oot lost woods
biji Jan 10, 2016 04:01 UTC		random breakbeat masstudio prod chisla qpad sense emphasis silk walk broken dreams regenkind lm1 nerver disciple galimatias major crimes unicorns feverkin koresma folds enzalla somewhere auditive escape plume de son dk okawari temperature of tears bonobo noctuary emancipator dusk to dawn kvzus memories of the past mellowmatix full bright sublab last time i saw u
biji Jan 9, 2016 23:11 UTC		1er playlist de test: hip hop us u better recognize official spice 1 strap on the side dpg reality tha shiznit

Designed and developed by biji
2015 © Dat Playlist

L'utilisateur est donc redirigé vers la page « View All ». Le code de sa playlist lui est également affiché. Il peut alors choisir de les sauvegarder s'il veut pouvoir modifier sa playlist plus tard.

Page de recherche de playlist

Dat Playlist Create Edit View All FAQ

Enter the code of the playlist

JBOETSHN

Search

Designed and developed by biji
2015 © Dat Playlist

Page d'édition de playlist


Dat Playlist
Create
Edit
View All
FAQ

Editing a playlist

Jan 10, 2016 20:34 UTC
play/CNPYCANKHKD/0
code: JBOETSHN

super meat boy hot damned
hotline miami daisuke
athletic theme
hotline miami perturbator
oot lost woods

Optional
 Optional

☒ Je ne suis pas un robot


Update
Delete

Designed and developed by biji
2015 © Dat Playlist




Depuis le menu « Edit », l'utilisateur a la possibilité de rechercher une playlist grâce à son code afin de la modifier ou la supprimer (dans cet exemple, il rajoute un auteur et une description).

Page de playlists publiques (après mise à jour d'une playlist)

Dat Playlist
Create
Edit
View All
FAQ

Public Playlists

Your playlist has been updated.

biji Jan 10, 2016 20:34 UTC		game ost super meat boy hot damned hotline miami daisuke athletic theme hotline miami perturbator oot lost woods
biji Jan 10, 2016 04:01 UTC		random breakbeat masstudio prod chisla qpad sense emphasis silk walk broken dreams regenkind lm1 never disciple galimatias major crimes unicorns feverkin koresma folds enzalla somewhere auditive escape plume de son dk okawari temperature of tears bonobo noctuary emancipator dusk to dawn kvzus memories of the past mellowmatix full bright sublab last time i saw u
biji Jan 9, 2016 23:11 UTC		1er playlist de test: hip hop us u better recognize official spice 1 strap on the side dpq reality tha shiznit

Designed and developed by biji
2015 © Dat Playlist

Page de lecture

Dat Playlist


Create

Edit

View All

FAQ

headline miami daisuke



Hotline Miami OST - Daisuke (El Huervo)

<https://www.youtube.com/watch?v=Ycj3Do9apPM>

super meat boy hot damned

headline miami daisuke

athletic theme

headline miami perturbator

oot lost woods

S'il clique sur le bouton de lecture d'une playlist, il sera redirigé vers cette page. Toutes les vidéos de la playlist seront ainsi jouées à la suite.

4. Conclusion

Dans l'ensemble, je suis satisfait du projet. En me basant sur un problème et une simple idée de base, j'ai réussi à réaliser un site utilisable avec encore, plusieurs fonctionnalités à implémenter dans le futur.

J'ai acquis des connaissances en Ruby, Ruby on Rails, les API Google (Youtube et ReCaptcha) et la configuration de serveur VPS. Mon objectif de base a été atteint et des idées sont encore à implémenter.

Cette utilitaire a été avant tout développé pour un usage personnel, mais étant donné son côté simpliste et ouvert, je serais ravi d'apprendre que d'autres personnes l'utilisent aussi.