hashtag your world

PROJET

HASHTREND

Agrégateur de contenu web

Dossier de conception technique

Version 1.0

Auteur du document :

Pellegrini Emanuele
Analyste et développeur

TABLE DES MATIÈRES

1 – VERSIONS

AUTEUR	*DATE*	*DESCRIPTION*	*VERSION*
Pellegrini Emanuele	26/02/2019	Création du document	1.0

2 – INTRODUCTION

2.1 – Objet du document

Le présent document constitue le dossier de conception fonctionnelle de l'application « Hashtrend », l'agrégateur de contenu multimédia basé sur les centres d'intérêt de l'utilisateur.
Objectif du document: présenter l'architecture technique de l'application web « Hashtrend ».

2.2 – Références

Pour de plus amples informations, se référer également aux éléments suivants :

1. **D.C.F. – Hashtrend :** Dossier de conception fonctionnelle de l'application.

3 – ARCHITECTURE TECHNIQUE

3.1 – Composants généraux

3.1.A – Interface utilisateur

L'utilisateur doit disposer d'une connexion Internet ainsi que d'un navigateur web afin d'accéder aux fonctionnalités de l'applicatif.

3.1.B – Application Hashtrend

a) Une partie front-end, notamment l'ensemble des pages web composant le site web où les clients peuvent profiter de ses fonctionnalités;
b) une partie back-end comprenant la base de données Htrend et le SGBDR PostgreSQL.

3.1.C – APIs externes

Les services des APIs qui interagissent avec l'application afin de collecter les données souhaitées par l'utilisateur (pour des plus amples informations sur ce point se référer à la section 3.2 - Application web, sous-sections 3.2.B – Diagramme des composants et 3.2.C – Détails des composants pages 6-7-8 du présent document.

3.2 – Application web

3.2.A – Pile logicielle

La pile logicielle est la suivante:

- Système d'exploitation Windows 10 Home édition, processeur Intel(R) core i7-8550U, CPU @1.80GHz – 1.99GHz, RAM 8Go, architecture système et processeur à 64 bit.
- Base de données Htrend conçue avec PostgreSQL 11.1 et administré à l'aide du SGBDR PgAdmin4.
- Serveur de déploiement : Heroku PaaS.
- Interpréteur de scripts:
    - coté back-end: Python 3.7.2

- coté front-end: HTML5, CSS3, JavaScript

- Principales ibrairies utilisées:
  - JavaScript: Bootstrap4, jQuery 3.3.1
  - Python: tweepy 3.7.0 pour l'interaction avec l'API de Twitter, praw 6.1.1 pour la communication avec l'API de Reddit, newsapi-python 0.2.3 pour l'API de Google News, oauthlib 3.0.1 pour gérer les protocoles d'uthentification, requests 2.17.1 pour la gestion des requêtes http.

3.2.B – Diagramme des composants

Le diagramme des composants en dessous aide à décrire l'organisation du système du point de vue des éléments logiciels et de mettre en évidence les dépendances entre les différents composants du système.

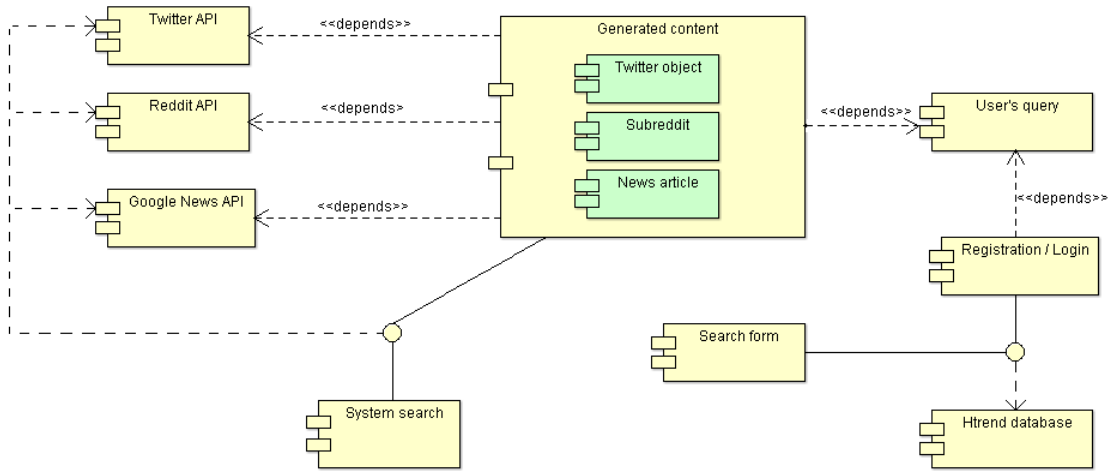

Fig. 3.1 – Diagramme des composants application Hashtrend

3.2.C – Détails des composants

Twitter API

Il s'agit de l'API Standard Search de Twitter qui fournit les services pour la récupération des données relatives aux tweets par rapport à un sujet donné.
Elle communique avec Hashtrend à l'aide de la libraire Python *Tweepy* qui facilite les opérations de requête et renvoi des données.

Reddit API

Il s'agit de l'API de Reddit qui est en charge de fournir les services utiles à la récupération des données relatives aux posts (appelés aussi *subreddits*) sur un sujet donné.
Elle interagit avec Hashtrend grâce à la librairie Python *Praw* qui facilite les requêtes et le renvoi des données souhaitées.

Google News Api

Il s'agit de l'API de Google qui collecte les articles recherchés depuis plus que 30.000 sources. Elle fournit les résultats relatifs aux articles désirés en communicant avec Hashtrend grâce à la librairie Python dédiée *Newsapi-python* qui facilite les opérations de requête et renvoi des données.

Generated content

Il représente la page des résultats (ou *newsfeed*) sous-forme de mur interactif. À son intérieur on retrouve trois autres instances de composants, **Tweet object**, **Subreddit** et **News article** qui représentent à leur tour la composition du *newsfeed*; respectivement ils indiquent un contenu provenant de Twitter, Reddit ou Google News.
On remarque que le contenu généré par Hashtrend est dépendant des services fournis par les API externes qui interagissent avec le système.

System search

Il s'agit de la recherche des résultats de la part du système. Il est le composant qui fait le pont entre les APIs et le contenu généré par l'applicatif.

User's query

Il indique l'input de l'utilisateur afin de démarrer une recherche. C'est à partir de ce input (ou mot-clé dans notre cas spécifique) que le système démarre une recherche en interrogeant les APIs concernées. On peut voir comment le contenu généré dépend aussi de ce composant.

Registration / Login
Est l'étape requise pour utiliser l'application. Un utilisateur non connecté ne pourra pas profiter des services d'Hashtrend. Elle est nécessaire pour le déroulement des actions du composant User's query.

Htrend database
Il représente la base de données de l'application Hashtrend qui est en charge de stocker les informations relatives aux utilisateurs connectés.

Search form
Il est le composant intermédiaire entre Registration/Login et Htrend database. En fait lorsq'un utilisateur se connecte à son compte ou en crée un, la base de donnée stocke ses informations personnelles et il lui sera possible de débloquer le formulaire de recherche à travers lequel effectuer une recherche avec Hashtrend.

4 – ARCHITECTURE DE DÉPLOIEMENT

L'architecture de déploiement de l'application à été créée en mappant les blocs fonctionnels logiques (l'architecture logique) sur un environnement informatique physique de sorte que les exigences de qualité de service spécifiées dans le scénario de déploiement soient respectées. Le scénario de déploiement est converti en une architecture de déploiement, comme illustré à l'aide du diagramme UML en dessous.

4.1 – Diagramme de déploiement

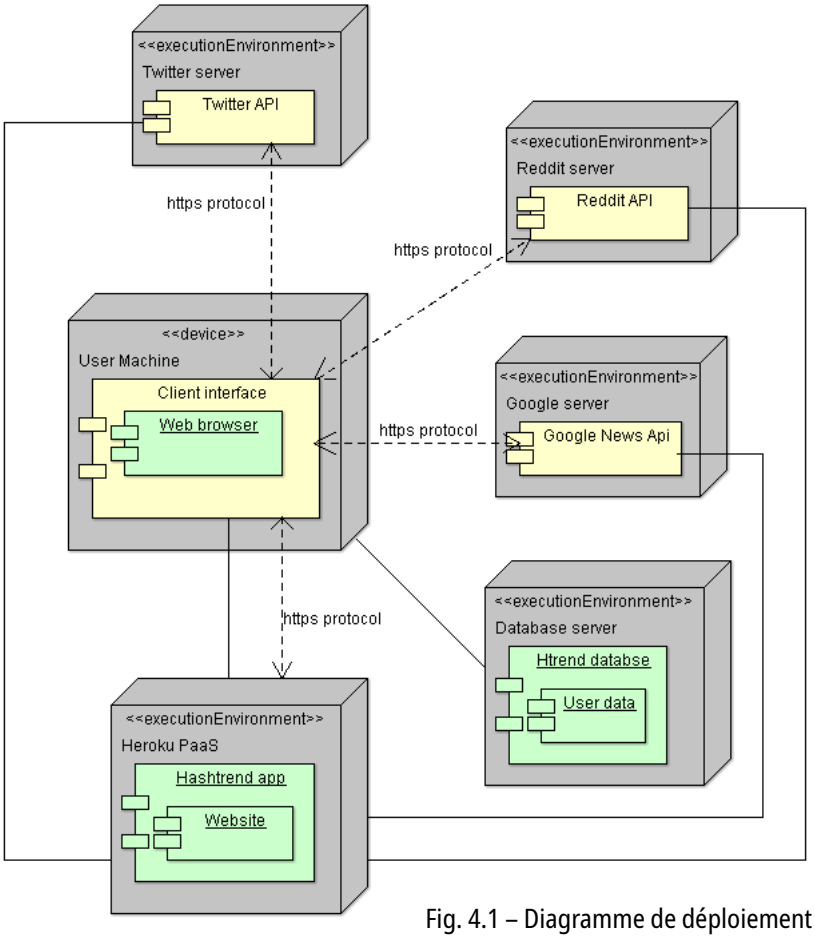

Fig. 4.1 – Diagramme de déploiement application Hashtrend

Comme on peut le voir das le schéma, il y a six nœuds principaux représentant les éléments hardware ou software du système.
Plus en détail, on y retrouve un nœud physique (d'où le mot-clé *device*) appelé **User Machine** qui indique l'ordinateur avec lequel l'utilisateur se connecte. **Client interface** représente l'interface utilisateur, à travers laquelle il peut accéder aux services de l'application Hashtrend à travers le navigateur web **Web Browser**.

Il y a ensuite les trois nœuds logiciels (donc marqués avec le mot-clé *executionEnvironment*) représentant les serveurs des applications propriétaires des APIs qui interagissent avec le système, **Twitter**, **Reddit** et **Google**.
Ces trois nœuds communiquent avec Client interface à l'aide du protocole https.
On remarque ensuite la présence du nœud logiciel **Database server**, caractérisant la base de données **HTrend database** qui est en charge du stockage et récupération des données relatives aux utilisateurs **User data**.
Au final on voit le nœud logiciel représenté par **Heroku**, qui contient l'application **Hashtrend app** ainsi que son site web **Website**. Heroku et Client interface communiquent avec le protocole https.

4.2 – Serveur de base de données

L'application en objet utilise le serveur de base de données PostgreSQL dans sa version 11.1. Il est en charge de stocker les informations personnelles des utilisateur nécessaires à la consommation des services de Hashtrend comme par exemple user ID, mot de passe et email.

4.3 – Serveur de déploiement

Hashtrend est déployé grâce aux services fournis par Heroku, une Platform as a Service. Elle est chargée de stocker et délivrer les pages web demandées par les utilisateurs de l'application.

5 – ARCHITECTURE LOGICIELLE

5.1 – Principes généraux

Les sources et les versions du projet Hashtrend sont gérées par **Git**, la structure du projet est gérée par **Django** dans sa version 2.1.5 selon le principe Model, View, Template (MVT) dans un environnement virtuel, le stockage des informations est pris en charge par le serveur de base des données **PostgreSQL** et la *Platform as a Service* **Heroku**.

5.1.A – Les couches

L'architecture applicative est la suivante:

les répertoires et les fichiers sont crées de façon à respecter la philosophie MVT:

– **Model**: interagit avec la base de données. Sa mission est de chercher dans une base de donnée les éléments correspondant à une requête et de renvoyer une réponse facilement exploitable par le programme. Il traduit les résultats d'une requête SQL en objets Python grâce à un ORM (Object Relational Mapping).

– **View**: sa responsabilité est de recevoir une requête HTTP et d'y répondre de manière intelligible par le navigateur. Si une interaction avec la base de données est requise, la vue appelle un modèle et récupère les objets renvoyés par ce dernier; si un gabarit est nécessaire, la vue l'appelle. Chaque vue est associée à une URL.

– **Template**: il s'agit d'un fichier HTML qui peut recevoir des objets Python et qui est lié à une vue.

5.1.B – Structure des sources

L'arborescence pour le projet sous Django est la suivante:

- Hashtrend: dossier contentant les scripts des parcours du projet (urls.py), les réglages des composants (settings.py) et le Web Server Gateway Interface (wsgi.py).
- Hashtrend_app: dossier contenant:
  - les scripts auto-générés par Django (admin.py, apps.py);
  - les scripts de gestion de l'applicatif comme:
    - forms.py pour les formulaires dédiés aux utilisateurs de l'application;

        - models.py pour la création et modification de la base des données;
        - urls.py pour les parcours des différents sections de l'application;
        - views.py pour interpréter les requêtes http et les rendre intelligible par le navigateur;
        - le dossier des migrations avec toutes les versions des modifications apportées à l'application;
        - le dossier « static » contenant:
            - ✔ le dossier avec les feuilles de style .css;
            - ✔ le dossier avec les scripts en JavaScript;
            - ✔ le dossier des images utilisées;
            - ✔ le dossier des plugins utilisés;
            - ✔ le dossier de Bootstrap;
            - ✔ le dossier des fonts utilisés;
            - ✔ le dossier « templates» contenant les scripts .html
- manage.py: script auto-généré par Django à la racine du projet et se chargeant de ses tâches administratives.
- requirements.txt: fichier contenant toutes les modules et libraires utilisées dans le projet.

5.1.C - Schéma arborescence projet Hashtrend

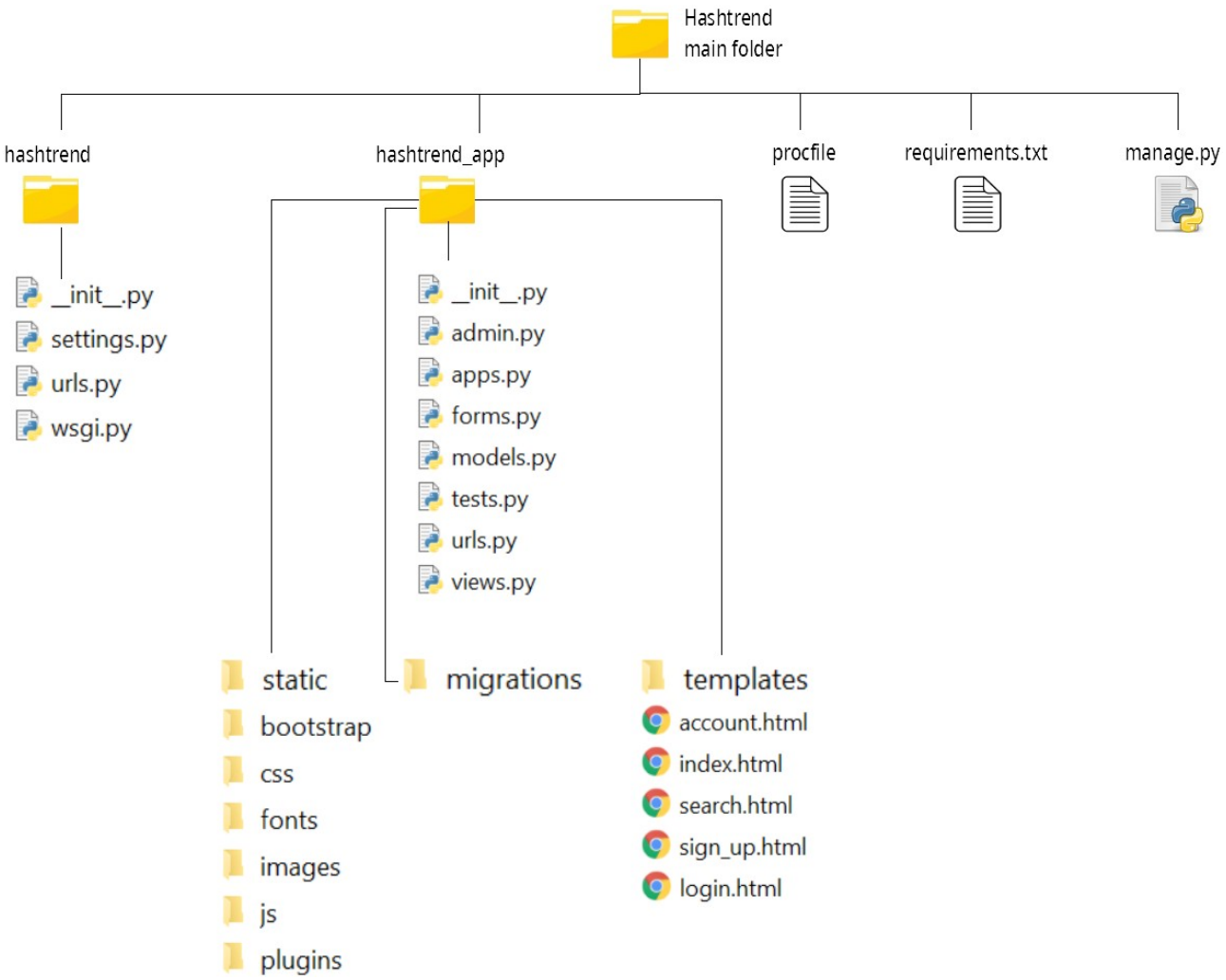

Fig. 5.1 – Schéma structure projet Hashtrend

6 – POINTS PARTICULIERS

6.1 – Fichiers de configuration

La configuration de l'application en objet est dépendante du fichier **settings.py** situé dans le dossier du projet Hashtrend (se référer à la figure 5.1 – Schéma structure projet Hashtrend en dessus). Il contient:

- la constante relative à la clé sécrète de Django;
- les constantes stockant les clés sécrètes pour accéder aux services des APIs de Twitter, Reddit et Google News;
- les *hosts* autorisés par l'application;
- les paramètres relatifs à l'application installée hashtrend_app, aux *middlewares*, aux URLs et aux templates;
- les spécifiques de la base de données: son nom, le moteur, le nom d'utilisateur, le mot de passe et la porte;
- les paramètres de traduction et localisation;
- les URLs de login et redirection.

6.2 – Environnement de développement

Le projet Hashtrend a été développé sous un environnement virtuel dans lequel ont été téléchargées toutes les dépendances nécessaires à sa création.

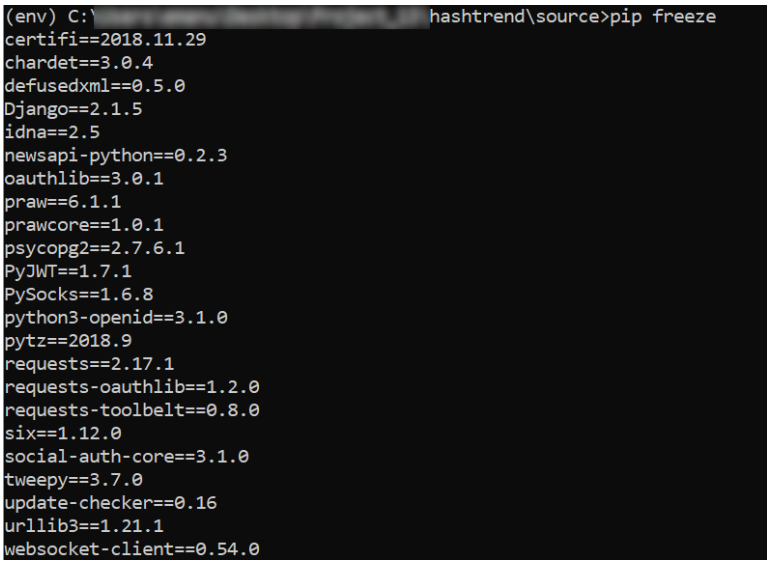

Fig. 6.1 – Dépendances du projet dans virtual env