Rapport

de

Projet

Table des matières

1. Introduction :

Le but de ce projet était la création et mise en ligne d'un site internet tout en appliquant des principes et concepts vus en classe. L'objectif était pour moi de faire quelque chose qui me plaisait car nous n'avions pas de sujet imposé tout en apprenant à me gérer pour progresser.

2. Présentation du site :

Mon site est une plate-forme permettant de voter sur une simple question : « Qui va régner sur Westeros », d'après la série Game of Thrones. De plus, je demande la provenance des votants permettant de faire une cartographie des votes en France. Une rubrique de contact, un mini-chat et des citations sont aussi présents sur le site. J'ai eu cette idée de site après avoir visité le site suivant : chocolatineoupainauchocolat.fr .

3. Description technique :

J'ai utilisé le framework Django pour mon site, qui se base sur du python et une architecture MVT (Model, View, Template). Cette organisation diffère de celle d'une MVC car le contrôleur est entièrement géré par django. Ce schéma introduit aussi le concept de template qui est une page HTML à laquelle on a intégré la possibilité d'afficher des variables ainsi que traiter des structures conditionnelles ou de boucle grâce à des balises de la forme {% %}.

Ce qui diffère aussi du PHP est que dans le modèle on définit des classes en python qui deviendront nos tables de notre database. Ces classes sont aussi directement traduite en SQL sur un fichier SQLite permettant le développement en local. L'utilisation de cette base de donnée étant, bien entendu, déconseillée en production.

Une dernière fonctionnalité bien pratique est la création de formulaire qui est

en fait une table et qui est appelé grâce à une balise dans le template. De plus, un token permet d'éviter les attaques de type CSRF (Cross-site request forgery).

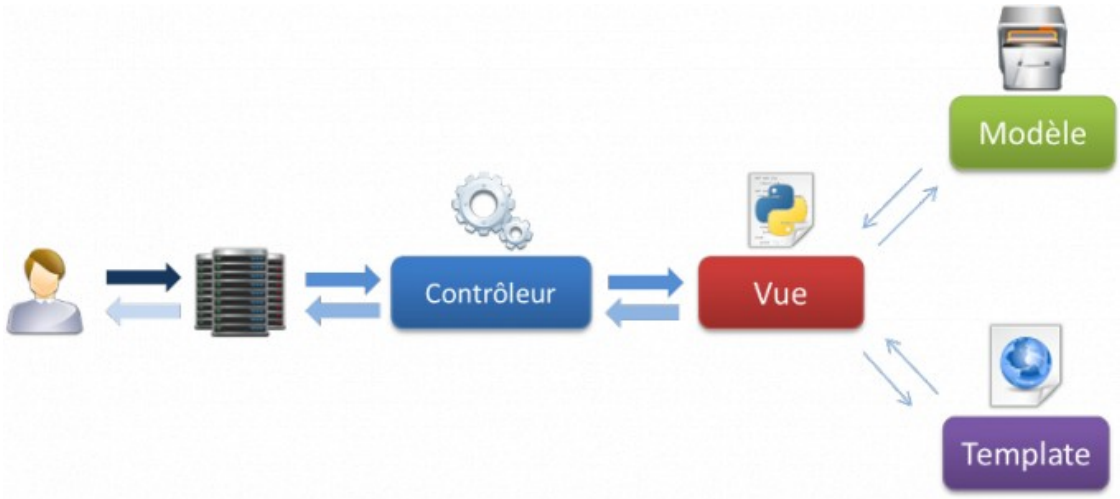

Illustration 1: Une architecture MVT

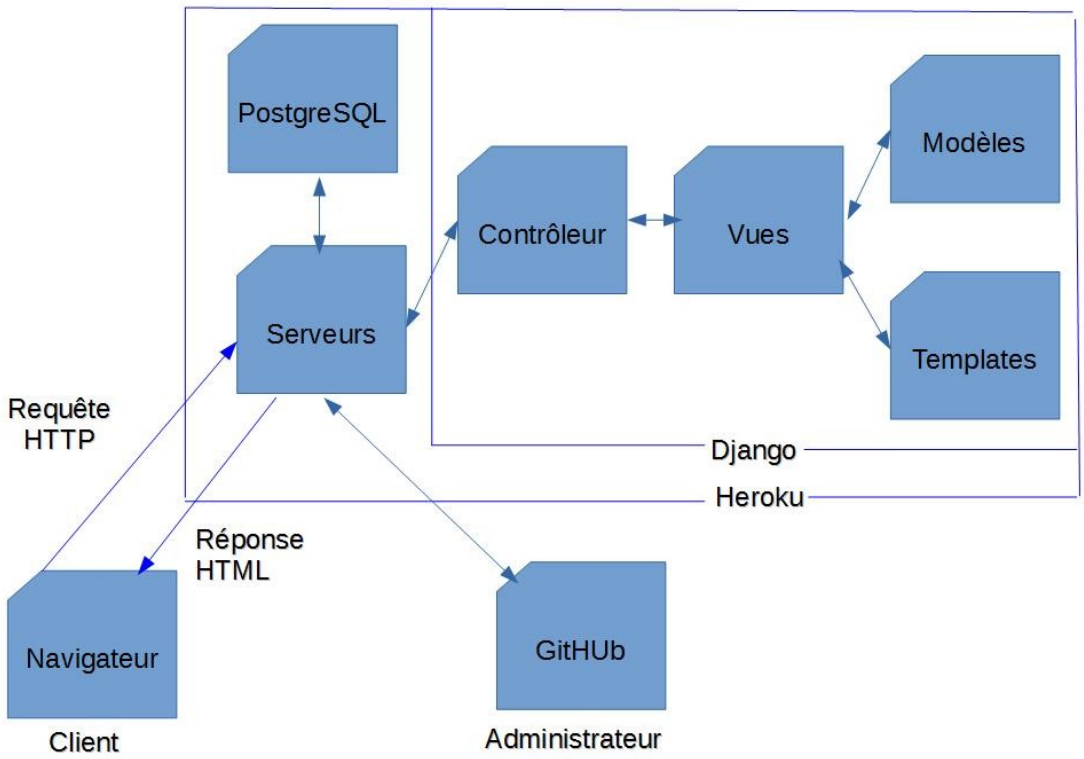

Illustration 2: L'architecture de mon site

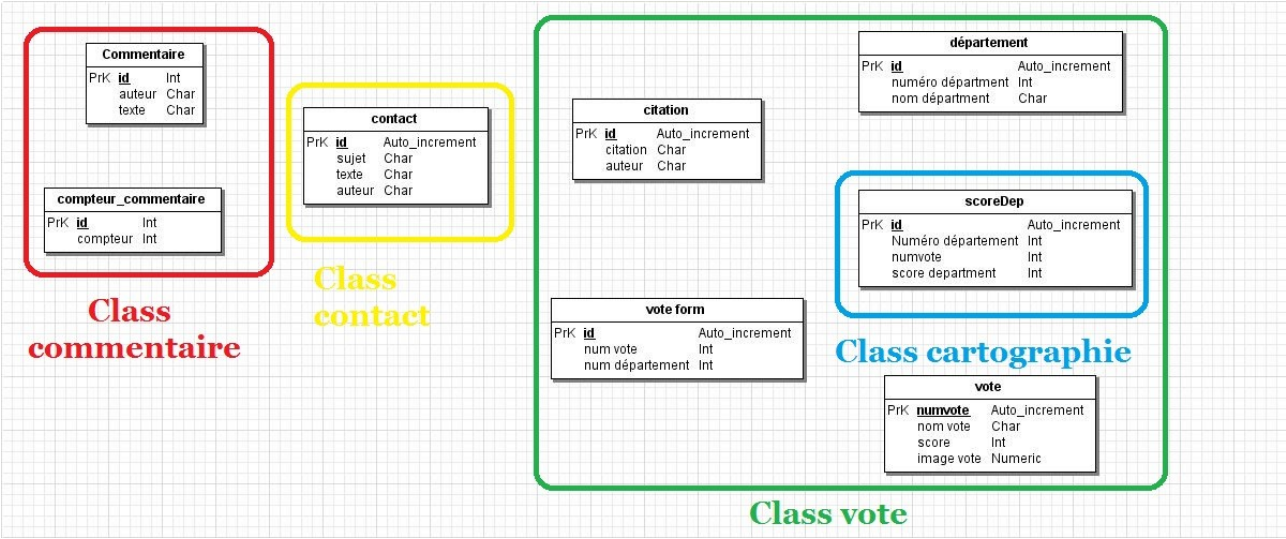

Illustration 3: Les classes et tables de mon site

Mon site se divise en 4 applications que j'ai nommé au-dessus (vote, cartographie, contact et commentaire). Chacune d'elles remplit un rôle bien précis et sont (quasi-)indépendantes. Pour l'hébergeur, le premier que j'ai trouvé qui était gratuit et qui supportait mon framework était heroku. Cela m'a donc obligé à travailler ma base de donnée en langage PostgreSQL, car c'est le seul proposé gratuitement par cet hébergeur. Trois fichiers à la racine de mon code source permettent à Heroku de savoir que lui envoie un site en langage Django.

J'ai créé 2 fonctions triggers (en PostgreSQL le trigger appelle une fonction) sur ma base de donnée. L'un qui permet de compter les commentaires du mini-chat. L'autre qui me supprime les formulaires de contact une fois qu'un booléen « lu » est coché. J'ai aussi ajouté une fonctionnalité pour les fans de cette série (dont je fait partie) qui permet de générer un nombre aléatoire, à chaque nouvelle page, qui affiche une citation et son auteur.

Pour les parties contact et commentaires, ce sont 2 formulaires qui sont relativement identiques. Les deux sont sauvegardés dans la base de donnée. Évidemment, j'affiche les commentaires du mini-chat. Enfin les parties vote et cartographie permettent de voir qui semble en tête dans cette question cruciale.

L'unicité des votes des participants est surveillée grâce à leur adresse IP. Lors du vote, je demande à chaque participant de m'indiquer, en plus de leur vote, leur département d'origine. Après ce vote, un premier score est affiché sur la page principale sans tenir compte du lieu d'origine du votant. Le second, qui est dans la page cartographie, affiche le score et le vote en tête dans chaque département. Je stocke donc chaque vote dans 2 tables différentes (vote et scoreDep).

Lors de la conception, j'avais pensé à garder en mémoire, en plus de l'IP, le vote et le département. Mais je me suis dit que c'était inutile puisqu'il me fallait juste vérifier, lors de la connexion si leur IP était présent en mémoire. Cependant je n'avait pas pensé que l'on pouvait voter plusieurs fois grâce au changement d'IP. Ainsi si un utilisateur avait voter plusieurs fois pour la même personne et dans le même département ça se serait vu s'il avait fait cela d'affilé. Cependant avoir plusieurs ordinateurs permet de voter plusieurs fois sans que cela soit considéré comme de la triche. Ainsi si un utilisateur veut voter plusieurs fois sur mon site cela montre qu'il y porte de l'intérêt, bien que cela fausse légèrement les résultats, et je ne peux lui en tenir rigueur. De plus les membres d'une même famille peuvent avoir un personnage préféré en commun donc le fait de rajouter les colonnes dans ma table votant ne permettait pas d'être à l'abri de supprimer des votes faits par 2 personnes différentes.

4. Développement :

Le choix de django m'a été proposé en consultant la page du tutoriel d'openclassrooms (anciennement site du zéro) sur le php suggérant une facilité pour apprendre ce langage en connaissant déjà le python. J'ai donc suivi le cours d'openclassrooms ainsi que celui de djangogirls (djangogirls.org), une association ciblant principalement les femmes, qui permet d'acquérir des compétences en programmation web, via notamment django.

Étant novice dans ces projets les principales difficultés ont été de cerner ce dont j'étais capable ou pas dans le temps qui m'était imparti, savoir où et comment chercher sur le net ainsi que de ne pas bloquer sur mes erreurs pour y revenir plus tard ou de s'en détacher pour mieux les cerner. La plupart de mes recherches ont, bien sûr, été en anglais et principalement sur les sites de la documentation django (docs.djangoproject.com), les très actifs forums de stackoverflow et la documentation PostgreSQL quoique moins fournie que celle de django.

Au début je travaillais en localhost grâce à un outil en ligne de commande de django, qui permet de simuler un serveur sur mon poste de travail. Ensuite lorsqu'il a fallu que je publie mon site sur heroku, j'ai utilisé github. Enfin, lorsque mon site a été presque terminé, j'ai diffusé le lien du site à mes connaissances pour remplir ma base de donnée.

J'avais bien sûr pensé à des fonctionnalités abandonnées par la suite mais que je peux garder en tête pour un futur développement. Je pensais laisser aux utilisateurs la possibilités d'ajouter des choix de vote (après modération) voire de nouveaux votes, faire des duels entre personnages, faire un forum plutôt qu'un mini-chat, etc.

5. Conclusion :

Malgré l'appréhension du départ, je suis bien arrivé à créer un site fonctionnel et à le diffuser sur le net. Les travaux de projet sont toujours pour moi très motivants et instructifs. L'utilisation de Django s'est avérée très pratique, cependant je me suis un peu « isolé » du reste de la classe qui codait en grande majorité en PHP et en MySQL. Malgré cela, je ne regrette pas mon choix car les projets django se doivent d'être très structurés et cela m'a appris et permis à savoir où chercher mes erreurs.