ECOLE PRATIQUE DES HAUTES ETUDES COMMERCIALES

HAUTE ÉCOLE EPHEC
Le chemin pratique vers la réussite

Avenue du Ciseau, 15 1348
Louvain-la-Neuve

Rolisticae : Un outil de Maitre de jeu, développée en Django & React

Travail de fin d'études présenté en vue de l'obtention du diplôme de bachelier en Informatique et Systèmes orientation Technologie de l'Informatique

Valentin DERREUMAUX

DREAM OF MORE
ROLISTICAE

Rapporteur : VROMAN Marie-Noël

Année Académique 2022 – 2023

Remerciements

Au terme de ce TFE, je tiens à remercier les différentes personnes qui m'ont aidé et soutenu. Premièrement, je tiens à remercier Madame VROMAN, pour son implication en tant que rapporteur de TFE, ainsi que pour le temps pour répondre à mes questions.

Je souhaite ensuite remercier Opese Kevin pour avoir accepté de prendre le rôle de client et pour le temps qu'il m'a consacré durant toute la durée du projet.

Je remercie également Olivier Percil pour ses conseils et astuces concernant l'aspect technique de mon projet.

Enfin, je tiens à remercier mon entourage pour le soutien qu'ils m'ont apporté. Plus spécifiquement, je remercie Mathilde Vandevoorde qui m'a aidé pour l'orthographe et les tournures de phrases de ce rapport.

Tables des Matières

Introduction

Ces dernières années le jeu de rôle a repris un certain élan avec l'arrivée de série telle que *Stranger Things*, *The ring of Power*, *The Witcher* ou de jeu vidéo comme *Baldur'Gate III*.
Le jeu de rôles a énormément gagné en popularité et semble parti pour continuer.[1]

Figure 1Trends D&D

Le nombre de personnes intéressées pour rejoindre ses tables de jeu a donc augmenté que ce soit au niveau de ses campagnes personnelles ou des demandes de groupes externes (jeune en difficulté, anniversaire…).

Afin de simplifier la gestion de toutes ses tables sans pour autant se noyer sous la masse d'information, un outil lui est plus que nécessaire afin de pouvoir rester un minimum compétitif avec d'autres Games Masters. Il lui est important de pouvoir avoir accès à ses informations en ligne afin de tenir ses joueurs au courant à tout moment.

Intervenant

Nom	Description	Contact
Valentin DERREUMAUX	Développeur du projet	HE201579@students.ephec.be
Kevin OPESE	Maitre de Jeu, Client, interlocuteur principal pour le projet	kevindugrenier@gmail.com
Marie-Noël VROMAN	Professeure dans le département Technologie de l'informatique à l'EPHEC, Rapporteur TFE	mn.vroman@ephec.be

Cahier de charge

Présentation du client

Mon client, Kevin Opese, est un passionné du jeu de rôle, endossant le rôle de Game Master pour de multiples tables de jeu, accueillant des publics diversifiés pendant son temps libre.

Kevin exerce le métier de développeur en région de Mons. Parmi ses joueurs réguliers, on retrouve :

- Des jeunes en situation difficile / Assossiation,
- Des joueurs amateurs,
- Ainsi que des joueurs confirmés.

L'ambition est de concevoir pour lui une vitrine numérique où il pourra présenter ses différents projets et campagnes, tout en optimisant la communication avec l'ensemble de ses joueurs.

Présentation du projet

Mon Travail de Fin d'Études (TFE) concerne la création d'un outil d'assistance pour le Game Master à travers une plateforme en ligne.

Cette démarche vise à éviter le recours à plusieurs plateformes, souvent onéreuses, en centralisant les fonctionnalités essentielles.

L'objectif premier de cette plateforme est de valoriser ses projets personnels de jeux de rôle et d'optimiser la gestion de ses tables de jeu.

La plateforme est articulée autour des éléments suivants:

1. Un outil de création de fiches de personnages en ligne :
    a. Pour centraliser la création des personnages et fournir un accès facile aux nouveaux joueurs.
    b. Pour permettre d'avoir une version virtuelle de cette fiche et des outils de gestion de celle-ci.
2. Un portfolio de ses campagnes passées
    a. Afin de permettre à ses anciens joueurs de retrouver leur histoire.
    b. Afin de montrer le genre de campagne déjà géré et donc l'expérience acquise après celle-ci. c. Afin de servir d'exemple pour d'éventuels intéressés (groupe éducatif, groupe d'amis, joueur en recherche)
3. Une fenêtre sur ses campagnes actives et son actualité.
    a. Afin de centraliser les informations concernant celles-ci pour les joueurs actifs
    b. Permettre un suivi pour les joueurs en attente
4. Une gestion de calendrier et de prise de rendez-vous
    a. Pour centraliser et simplifier la prise de rendez-vous pour une session de jeu
5. Un emplacement de stockage pour ses ressources numérique :
    a. Illustration
    b. FanArt
    c. Memes
6. Une Gestion du Lore
7. Un Accès aux réseaux sociaux
8. Une gestion rapide des personnages (affichage, état, inventaire… des joueurs)
    a. a afin de faciliter la vue du GM durant les parties de jeu.

Cible/Utilisateur

Le projet vise principalement

Joueur confirmé / actif

Un joueur ayant déjà pris part à une ou plusieurs sessions est familier avec les dynamiques d'une table de jeu.

- Il aspire à trouver un lieu où il peut revivre l'atmosphère unique de ses parties.
- Il souhaite un site où il peut se tenir informé des prochaines sessions

Joueur débutant

Un joueur, attiré par un univers captivant et désireux de s'immerger dans le monde du jeu de rôle, souhaite :

- Disposer d'une vitrine lui permettant d'explorer l'essence d'une campagne ou d'un GM.

Association

Une entité recherchant des maitres de jeu pour des évènements, découverte…

- Demande une vitrine afin de découvrir les différents univers et la manière de fonctionner du maitre de jeu

Maitre de jeu

Un Maitre de jeu à la recherche d'un lieu de communication et de contact pour gérer ses personnages et campagnes.

- Demande une administration capable de gérer des calendriers, campagnes, feuille de personnages…
- Demande un Blob

Solution déjà existante

Dans l'univers en constante évolution du jeu de rôle, les plateformes en ligne ont émergé comme une réponse essentielle à la nécessité d'unir des joueurs du monde entier.

Ces plateformes ont non seulement brisé les barrières géographiques, mais ont aussi enrichi l'expérience ludique grâce à une panoplie d'outils interactifs et immersifs.

Alors que certains joueurs privilégient encore les rencontres en face à face autour d'une table, il est indéniable que l'ère numérique a introduit une variété de solutions pour répondre aux besoins diversifiés des rôlistes. De ce fait pléthore de plateforme existe déjà.

Roll20

Roll20 est une plateforme en ligne de jeux de rôle sur table (JDR). Elle permet aux joueurs de se retrouver virtuellement pour jouer à des jeux de rôle comme "Dungeons & Dragons", "Pathfinder" et bien d'autres. Voici son principal attrait :

Avantage :

+ Plateforme web : tout se passe directement dans le navigateur.

+ Cartes et jetons interactifs : Les maîtres du jeu peuvent importer des cartes et utiliser des jetons pour représenter les personnages, les monstres, etc. Ces jetons peuvent être déplacés sur la carte en temps réel.

+ Système de dés virtuels : Les joueurs peuvent lancer des dés directement dans l'interface, avec des formules complexes si nécessaire.

+ Fiches de personnage : Roll20 propose des fiches de personnage.

+ Marketplace : les utilisateurs peuvent acheter des modules, des cartes, des jetons, des musiques et d'autres ressources pour améliorer leurs jeux.

+ Communauté : La plateforme possède une grande communauté, avec des forums, des groupes de jeu et des ressources pour aider tant les nouveaux joueurs que les vétérans.

Inconvénient :

- Courbe d'apprentissage : Bien que Roll20 offre de nombreuses fonctionnalités, cela signifie également qu'il peut être compliqué à apprendre pour les nouveaux utilisateurs, ou nouveaux joueurs, la plateforme restant très obscure sur les campagnes pour non-initié.

- Coûts : Bien qu'une version gratuite existe, celle-ci est très limitée et limite la visibilité de vos campagnes.

- Fiche personnalisée extrêmement complexe : Pour réussir à produire une fiche de personnage complexe, la quantité de travail demandée est beaucoup trop grande.

Bien que ces plateformes soient indéniablement efficaces, elles nécessitent souvent un abonnement payant pour exploiter pleinement leurs fonctionnalités. Souhaitant éviter la multiplication des abonnements sur différentes plateformes, la solution idéale pour lui serait de disposer de son propre site. Toutefois, il est essentiel de souligner que ce projet n'aspire pas à supplanter ces plateformes existantes.

Au contraire, il vise à combler efficacement les lacunes ou à simplifier les aspects jugés trop complexes par le client.

Planification future du projet

Étant régulièrement joueur à ses côtés et également maître de jeu dans ses campagnes, j'assume la responsabilité de veiller à la maintenance de la plateforme aussi longtemps que nécessaire. De plus, puisque mon client est lui-même développeur, nous aurons la capacité d'enrichir le site avec de nouvelles fonctionnalités au fil du temps.

Pérennité du futur du projet

Tant qu'il n'existe pas de plateforme offrant une visibilité aussi importante, à la fois simple d'utilisation et économique, nous sommes déterminés à assurer la maintenance et l'amélioration continue de la nôtre.

Aspect financier

Tant que les coûts de maintenance ne dépassent pas 120$ par an, il demeure plus avantageux de conserver la plateforme.

Si toutefois ces coûts venaient à excéder ce montant, il serait essentiel de réévaluer l'état et la pertinence du projet avant de décider de sa continuation.

Demandes fonctionnelles

- Le client souhaite disposer d'une section administrateur pour garder un contrôle total sur les informations circulant sur le site.
- Le client désire la capacité de créer des fiches de personnages sur mesure, similaires à celles qu'il utilise actuellement, sans avoir à passer par Google Sheet.
- Le client souhaite intégrer un blog pour rédiger des résumés des parties jouées.
- Le client veut une galerie pour archiver les mèmes, illustrations, et autres créations des joueurs.
- Le client veut un espace dédié au stockage et à l'affichage de fichiers, principalement des PDF, pour que tous puissent consulter les règles et autres informations pertinentes.
- Le client souhaite offrir une présentation succincte des campagnes pour donner un aperçu aux joueurs intéressés.
- Le client souhaite un outil de recherche pour parcourir les fichiers, facilitant ainsi l'accès au document contenant la réponse recherchée.
- Le client désire permettre aux joueurs de se connecter et de modifier les informations relatives à leurs personnages.
- Le client veut un système affichant la liste des personnages détenus par chaque joueur.
- Le client souhaite des liens redirigeant vers ses réseaux sociaux.
- ~~Le client souhaite un outil d'inscription et de liste d'attente pour de nouveaux joueurs~~
  Suite à de nombreuses arrivées, il n'est plus capable de gérer la quantité de joueurs et mets en pause cette fonctionnalité

Charte graphique / ergonomique

Le client souhaite une interface conviviale et épurée.
L'interface doit être simple, intuitive et adaptée à des utilisateurs novices en informatique et sur internet.

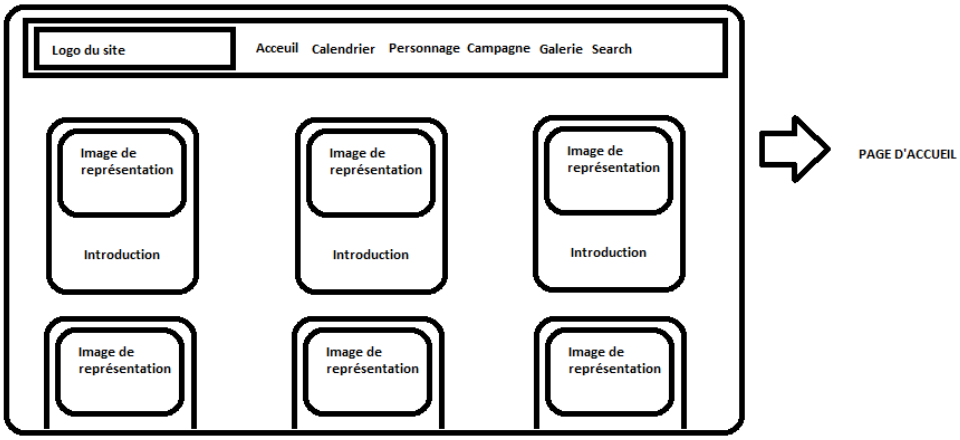

Figure 2Visuel envoyé par le client

Le client possédait une vision bien définie de l'aspect visuel qu'il souhaitait pour le site. Suite à plusieurs réunions collaboratives, ensemble, nous avons réussi à cerner avec précision ses attentes esthétiques. Nous avons finalement opté pour un design simple et épuré, mettant ainsi l'accent sur les illustrations.

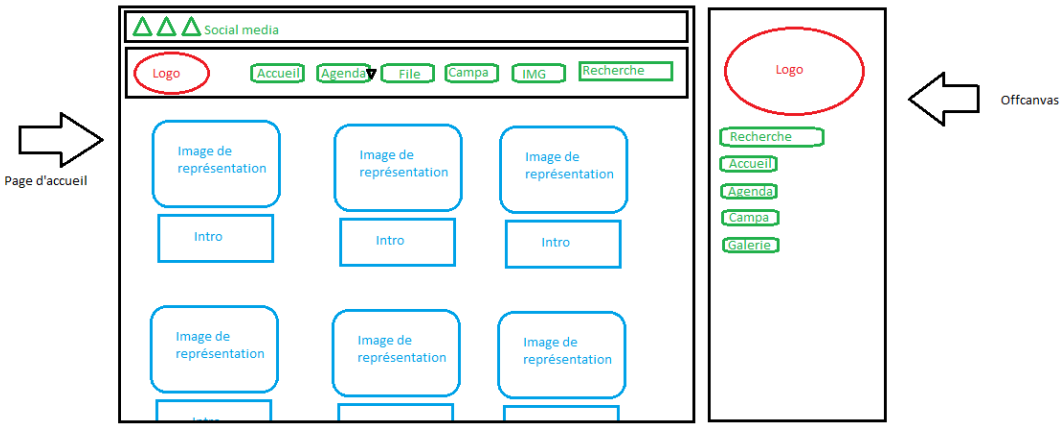

Figure 3Visuel créé avec le client

Lors de chaque réunion, le visuel était un point qui était rediscuté afin de coller au mieux à la vision du client

Contraintes

Suite aux différentes réunions avec le client, nous avons pu mettre en évidence les contraintes suivantes :

- Git et un gestionnaire de source devra être utilisé afin de partager le code.

- Un outil de gestion des tâches du projet devra être utilisé et partagé avec le client.

- Un gestionnaire de temps devra être utilisé et partagé avec le client.

- Le site web devra utiliser uniquement HTTPS afin de fournir une solution sécurisée.

- Le site doit être accessible par le DNS fourni par le Client

- Le Frontend doit être en React afin qu'il puisse, si besoin, efficacement effectuer des changements.

- Le projet sera hébergé sur un serveur fourni par le client et sous sa responsabilité.

- Le site web doit être compatible avec les principaux navigateurs web.

- Aucune contrainte de temps autre que celle du TFE

- Une réunion client est prévue toutes les 2 semaines (le vendredi).

- Lors de cette réunion, une démonstration de l'avancement du projet devra être faite.

Planification

- 1re étape : Planification de l'ensemble des fonctionnalités et de la structure que l'application respectera.

- 2e étape : Implémentation du MVP (Minimum Viable Product) représentant l'ensemble des fonctionnalités minimum du programme afin de pouvoir visualiser au mieux le projet et pour avoir des retours sur ce qu'il y aurait à ajouter potentiellement. Cela représente une sorte de prototype.

- 3e étape : Tester le bon déroulement de toutes les fonctionnalités implémentées dans le MVP et démonstration au client le bon fonctionnement du prototype.

- 4e étape : Prise en compte des remarques éventuelles par rapport au MVP et ajouts de fonctionnalités supplémentaires tels que la création de personnages récurent.

- 5e étape : Test du fonctionnement de l'application globale.

- 6e étape : Finalisation et déploiement de l'application.

26 Janvier - 5 Février	Squelette du Projet terminer, Github , Clockify lancer, database ready
6 Février - 19 Février	Mise en Place de la partie « Actualité »
20 Février - 12 Mars	Mise en Place de la partie « Création de Joueur et gestion de fichier »
13 Mars - 26 Mars	Mise en place de la « gestion de calendrier »
27 Mars - 9 Avril	Mise en place des « Outils de gestion rapide des joueurs »
10 Avril - 23 Avril	Mise en place du « créateur de Token et de la gestion OCR »
24 Avril - 6 Mai	Mise en place des tests
7 Mai - 21 Mai	Finalisation du produit et bug hunting
22 Mai - Juin	Correction bug et Présentation TFE

Figure 4Planning défini au début du projet

Analyse et choix des technologies

Framework Backend

Comparatif

Django

Figure 5 logo Django

Django est un Framework web de haut niveau pour Python, qui comprend de nombreuses fonctionnalités intégrées pour aider à accélérer le développement web [2].

Django fournit des fonctionnalités web courantes afin de ne pas devoir les créer soi-même [3].

Ceci est utile lors du développement de sites web nécessitant des fonctionnalités communes telles que les interfaces d'administration, l'authentification des utilisateurs et les connexions à la base de données [4].

Concernant la création du panneau d'administration, il peut être généré automatiquement à partir du code. Django est « pluggable », ce qui permet d'ajouter facilement de nouvelles fonctionnalités [5].

Django possède également un ORM (« object relational mapping ») intégré qui est une technique de programmation qui donne l'illusion de travailler avec une base de données orientée objet. Il va faciliter l'interaction avec les bases de données en utilisant des objets Python plutôt que des requêtes SQL [6].

Django a une communauté très active de développeurs et beaucoup de documentation disponible et utile pour résoudre facilement les différents types de problèmes rencontrés.

Pour l'aspect de la sécurité, il est considéré comme une solution sûre [7].

Il est le second framework le plus utilisé [8], il est basé sur le langage Python qui est en constante progression [9] et il est donc relativement facile de trouver des développeurs disponibles [10].

Au niveau de la politique de publication, Django est soucieux de garder la rétrocompatibilité avec les versions précédentes, même en cas de version majeure. Les échéances d'obsolescence sont connues et peuvent être anticipées [11].

Tableau 1 Django Avantage et Inconvénients

Avantage	Inconvénients
+ Le Framework possède des vues génériques afin d'optimiser le travail du développeur.	- Taillé pour de gros projets de grosse taille
+ Il possède un ORM très puissant, car il prend les données présentes dans nos tables et les transferts sous forme d'objet.	- Pas d'AJAX intégré directement au framework
+ La sécurisation avec Django est plus facile, car le Framework empêche déjà les failles les plus fréquentes.	- Seulement une seule requête est gérée, ce qui augmente le temps de réponse.
+ Occupe de plus en plus de place au niveau professionnel et est de plus en plus demander.	-Documentation parfois erratique.
+ La création et ajout d'application en cours de développement est simplifié.	

Flask

Figure 6 logo Flask

Flask est un Framework web Python open source conçu pour la simplicité.

Flask est un microframework, ce qui signifie qu'il a un noyau simple presque minimaliste qui peut être étendu si nécessaire [7].

Si on ajoute des extensions pour disposer de certaines fonctionnalités, cela devient plus compliqué, car il faut s'assurer que les librairies restent à jour. Concernant la gestion des données, Flask ne prend pas en charge le framework ORM et les modèles de données.[12]

Il supporte les bases de données relationnelles et non relationnelles [13]. Il est le troisième framework le plus utilisé [8].

Sa communauté est active et possède beaucoup de documentation.

Flask, en tant que framework web, est également connu pour sa rétrocompatibilité [14]. Flask est idéal pour les petites applications web uniques qui ne nécessitent pas de nombreuses fonctionnalités web courantes [13].

Il est souvent utilisé pour des projets web simples, les prototypes et les projets web éducatifs [15].

Tableau 2 Flask Avantage et Inconvénients

Avantage	Inconvénients
+ Léger et flexible : Flask est minimaliste par défaut, ce qui signifie qu'il ne charge pas d'outils ou de bibliothèques supplémentaires à moins que vous en ayez besoin. Cela le rend flexible et adapté à de nombreux types de projets.	- Pas "tout-en-un" : Contrairement à d'autres frameworks comme Django, Flask ne vient pas avec une pile complète. Par exemple, il n'inclut pas d'ORM (système de gestion de bases de données) ou d'authentification par défaut. Vous devrez intégrer ou construire ces fonctionnalités vous-même si vous en avez besoin.
+ Facilité d'utilisation : Avec sa syntaxe simple et son approche "convention sur configuration", il est facile de démarrer rapidement avec Flask	- Moins d'outils intégrés : En raison de son caractère minimaliste, les développeurs peuvent avoir besoin de passer plus de temps à rechercher et à intégrer des outils tiers pour des fonctionnalités spécifiques.
+ Intégration avec Python : Flask est écrit en Python, ce qui le rend idéal pour les développeurs familiarisés avec le langage. Cela permet également une intégration facile avec d'autres bibliothèques et outils Python.	- Évolutivité : Bien que Flask soit adapté aux petites et moyennes applications, il peut nécessiter des ajustements et des optimisations pour gérer de grandes applications ou un trafic très élevé.
+ Communauté active : Flask bénéficie d'une communauté solide et active qui contribue à son développement, fournit des extensions utiles et offre un soutien par le biais de forums et de tutoriels.	

Node Js

Figure 7 Logo Nodejs

NodeJS est un Framework JavaScript qui permet donc l'utilisation d'un seul langage en backend et en front end.

Node.js est une plateforme logicielle open source qui permet d'exécuter du code JavaScript côté serveur [16].
Il est basé sur le moteur JavaScript V8 de Chrome [17].
Grâce à npm, le gestionnaire de paquets de Node.js, il est possible d'accéder à des centaines de milliers de paquets pour étendre notre application [18].

Node.js est largement adopté dans l'industrie et est utilisé par de nombreuses grandes entreprises.

Tableau 3 Nodejs Avantage et Inconvénients

Avantage	Inconvénients
+ Sa rapidité qui est due à son langage de programmation qui est le JavaScript. De plus il fonctionne avec le moteur v8 de Google qui est également très rapide étant donné qu'il peut compiler le JS directement.	- L'interface de l'API n'est pas stable
+ Il profite également des avantages de NPM. Cela lui permet de facilement implémenter de nombreux modules.	- Node ne dispose pas d'une bibliothèque correcte pour gérer les opérations de base de données ou traiter des images. Il faut se fier à la bibliothèque commune.
+ Il s'agit d'une solution idéale pour les applications web en temps réel. En effet, son architecture évènementielle est très fiable. De plus, la synchronisation est très facile à faire, car, le langage utilisé est le même.	- Mauvais pour le calcul lourd, car cela demande beaucoup de ressources au niveau du CPU.
+ Le programme est monothread, il met en place des boucles ainsi que des méthodes de rappel afin de pouvoir gérer plusieurs clients en simultané et ainsi éviter le temps d'attente durant le processus serveur	- Mauvais avec les bases de données relationnelles, car les outils sont fort peu développés actuellement par rapport à ses concurrents.

Choix de la technologie

Lors de la conception de TFE, le choix du Framework backend était une étape décisive.

Après avoir étudié plusieurs options et m'être entretenu avec mon client, nous avons décidé d'opter pour Django pour les raisons suivantes:

1. **Historique impressionnant**: Depuis son lancement en 2005, Django a été au cœur de nombreuses applications web reconnues. Sa longévité témoigne de sa robustesse et de sa fiabilité. Ce qui est l'idéal si le projet doit durer dans le temps.
2. **Philosophie "Batteries Included"**: L'une des forces de Django est sa capacité à offrir une multitude de fonctionnalités intégrées. Ce qui permettra de gagner un temps précieux.
3. **Priorité à la Sécurité**: Django offre des protections intégrées contre des vulnérabilités courantes, ce qui est pratique.
4. **ORM Intégré**: Grâce à l'ORM de Django, il sera possible de lier des Database sans problèmes.
5. **Interface d'Administration incluse**: Avec son interface d'administration intégrée, il sera plus facile de gérer l'application sans devoir développer celle-ci ce qui est un gain de temps considérable.
6. **Communauté dynamique**: Avoir accès à une communauté active est un avantage indéniable. En cas de problème il est possible de trouver de l'aide sans trop de soucis.

Framework Frontend

Comparatif

React

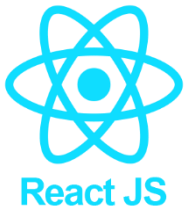

Figure 8 React logo

React, développé et maintenu par Facebook, est une bibliothèque JavaScript open source conçu pour construire des interfaces utilisateur (UI) pour des applications web monopages [19]. Depuis sa première publication en 2013, React est devenu l'une des solutions les plus populaires pour le développement frontend, et voici pourquoi:

1. **Composants réutilisables**: L'une des principales caractéristiques de React est la capacité de diviser l'interface utilisateur en composants indépendants. Ces composants peuvent être réutilisés, ce qui facilite la maintenance et permet une cohérence dans l'ensemble de l'application [20].
2. **Virtual DOM**: React utilise le Virtual DOM, une représentation légère du DOM réel. Lorsqu'un changement se produit, React crée un nouveau Virtual DOM et le compare avec l'ancien pour déterminer les modifications à apporter. Cela optimise les performances en minimisant les manipulations réelles du DOM .
3. **Flexibilité**: Contrairement à d'autres frameworks, React se concentre uniquement sur la construction d'interfaces utilisateur. Cela offre une grande flexibilité, car les développeurs peuvent choisir les bibliothèques ou Frameworks qu'ils souhaitent utiliser pour d'autres aspects de l'application, tels que la gestion des états ou les appels API .
4. **Communauté active**: La popularité croissante de React a conduit à la formation d'une communauté dynamique. Cela signifie un accès à une multitude de ressources, de tutoriels, et une vaste bibliothèque de composants tiers disponibles [21].
5. **Adaptabilité**: React n'est pas limité au développement web. Avec des solutions comme React Native, les développeurs peuvent également créer des applications mobiles natives pour iOS et Android en utilisant la même base de code [22].

Tableau 4 React Avantage et Inconvénients

Avantage	Inconvénients
+ Le framework est assez intuitif étant donné que tous ses états peuvent être pris en compte.	- Il y a un manque de documentation certain
+ Le développement peut se faire rapidement, car on peut travailler sur une partie individuelle sans pour autant impacter le reste du code	- Le JSX n’est pas optimisé avec React
+ La création d’un nouveau composant est assez simple	
+ Le testing du code est assez facile vu que la structure est bien définie.	

Angular

Figure 9Angular logo

Angular, développé et maintenu par Google, est un framework JavaScript open source destiné à la création d'applications web monopages et mobiles [23].
Depuis son lancement en 2010 sous le nom d'AngularJS, puis sa refonte complète en 2016 sous le nom d'Angular, il est devenu un acteur majeur dans le domaine du développement frontend. Voici les raisons de sa popularité:

1. **Architecture modulaire**: Angular est basé sur une architecture modulaire, ce qui permet une meilleure organisation du code, une réutilisabilité accrue et une facilité de maintenance [24].
2. **Two-Way Data Binding**: l'une des caractéristiques distinctives d'Angular est le two-way data binding. Cela permet une synchronisation automatique entre le modèle et la vue, simplifiant ainsi le développement et améliorant les performances.
3. **TypeScript**: Angular est écrit en TypeScript, un sur-ensemble de JavaScript. TypeScript offre des avantages tels que la vérification de type statique, ce qui peut aider à identifier et corriger les erreurs plus tôt dans le processus de développement [25].
4. **Injection de Dépendances**: Angular utilise l'injection de dépendances pour fournir des instances de services aux composants, améliorant ainsi la modularité et l'efficacité du code [26].
5. **Communauté active**: Avec le soutien de Google et d'une vaste communauté de développeurs, Angular bénéficie d'une multitude de ressources, de tutoriels et de mises à jour régulières [27].
6. **Angular CLI**: L'interface de ligne de commande d'Angular (CLI) facilite la création, le développement et le déploiement d'applications Angular, améliorant ainsi l'efficacité du processus de développement [28].

Tableau 5 Angular Avantage et Inconvénients

Avantage	Inconvénients
+ Architecture modulaire: Angular est basé sur une architecture modulaire, ce qui permet une meilleure organisation du code, une réutilisabilité accrue et une facilité de maintenance. Cela facilite également le chargement paresseux, permettant de charger des fonctionnalités de l'application à la demande.	- Performance: Bien qu'Angular soit performant pour la plupart des applications web, il peut rencontrer des problèmes de performance pour des applications très dynamiques ou avec de grandes quantités de données. Le two-way data binding, bien que pratique, peut parfois entraîner des problèmes de performance s'il n'est pas géré correctement.
+ La liaison de données est bidirectionnelle, c'est-à-dire que les données peuvent changer dans la vue ainsi que dans le code.	- Taille du Bundle: Les applications Angular peuvent avoir une taille de bundle initiale plus grande par rapport à d'autres frameworks ou bibliothèques. Cela peut affecter le temps de chargement des applications, en particulier sur des connexions lentes.
+ Il est possible de mettre en place certaines directives personnalisées	
+ La prise en main est assez simple	

Choix de la technologie

Lors de la planification de mon projet, le choix du framework frontend était aussi une étape cruciale. Bien que plusieurs options aient été envisagées, le choix s'est finalement porté sur React pour les raisons suivantes:

1. **Demande du Client**: La première et principale raison de ce choix a été la demande explicite du client. Il souhaitait spécifiquement utiliser React pour le développement de l'application, car il voulait avoir la possibilité de modifier les composants lui-même ultérieurement.
2. **Composants réutilisables**: React est basé sur une architecture de composants. Ces composants sont modulaires et peuvent être réutilisés à travers l'application, ce qui facilite la maintenance et permet une cohérence dans l'ensemble du projet.
3. **Virtual DOM**: React utilise le Virtual DOM, ce qui optimise les performances en minimisant les manipulations réelles du DOM. Cela permet une mise à jour efficace et rapide de l'interface utilisateur.
4. **Communauté active**: React jouit d'une grande popularité et d'une communauté dynamique. Cela garantit un accès à une multitude de ressources, de bibliothèques tierces et de mises à jour régulières.
5. **Flexibilité**: React, en tant que bibliothèque, offre une grande flexibilité. Il peut être intégré à diverses solutions backend et s'adapte bien à diverses architectures.

En conclusion, bien que la demande du client ait été le facteur déterminant dans le choix de React, les caractéristiques intrinsèques de React et ses avantages en font une solution adaptée pour le développement de ce projet.

Déploiement du site

Analyse du besoin

Django dispose d'un mécanisme de déploiement, mais celui-ci n'est adapté que pour un environnement de développement. De ce fait il est possible d'héberger le site localement, mais lors de la mise en ligne de celui sur l'environnement de production il y aura de multiples problèmes.

Comme il faut un environnement de production, il faut réfléchir aux mécanismes qui permettront de l'implémenter.

Django possède deux interfaces différentes permettant le déploiement en production [19]:

1. WGSI : C'est le standard Python qui est utilisé pour la communication en synchrone avec le serveur web et les applications.
2. ASGI : C'est un nouveau standard Python qui contrairement à WGSI permet les communications asynchrones. Un problème vis-à-vis de ces deux possibilités est qu'aucune ne permet de gérer les accès aux fichiers statiques. Il faudra donc aussi réfléchir à une solution pour gérer les fichiers statiques tels que des fichiers CSS et JavaScript.

En accord avec le client j'ai choisi WSGI par défaut

Comparatif

Nginx

Nginx est un serveur web open source, ainsi qu'un serveur proxy inverse. Initialement conçu pour résoudre les problèmes de performance causés par la gestion des connexions dans les serveurs web traditionnels, Nginx utilise un modèle basé sur des événements, ce qui le rend extrêmement léger et performant.

Tableau 6 Nginx Avantage et Inconvénients

Avantage	Inconvénients
Hautes performances : Nginx est conçu pour servir un grand nombre de connexions simultanées avec une faible utilisation de la mémoire.	**Moins de modules** : Bien que Nginx offre des modules pour de nombreuses fonctionnalités, Apache à une gamme de modules plus vaste.
Serveur proxy inverse : Nginx excelle dans le routage du trafic vers les applications en arrière-plan, ce qui est utile pour le load balancing.	**Complexité** : Certaines configurations avancées peuvent être plus difficiles à mettre en œuvre que sur Apache.
Performance statique : Très rapide pour servir du contenu statique grâce à son architecture basée sur des événements.	
Configuration flexible : Son fichier de configuration est souvent considéré comme plus propre et plus facile à lire que celui d'Apache.	

Apache

Apache, également connu sous le nom d'Apache HTTP Server, est l'un des serveurs web les plus anciens et les plus fiables. Il utilise un modèle basé sur les processus ou les threads pour gérer les requêtes.

Tableau 7 Apache Avantage et Inconvénients

Avantage	Inconvénients
Modularité : Apache offre une large gamme de modules pour étendre ses fonctionnalités.	**Performances** : Dans certaines configurations, Apache peut utiliser plus de ressources que Nginx, en particulier avec un grand nombre de requêtes.
.htaccess : les fichiers .htaccess permettent des configurations détaillées au niveau du répertoire.	**Modèle basé sur les processus** : Peut être moins efficace que le modèle basé sur les événements de Nginx pour gérer un grand nombre de connexions simultanées.
Documentation : Disposant d'une longue histoire, Apache est bien documenté et largement utilisé, ce qui facilite la recherche de solutions à des problèmes spécifiques.	
Flexibilité : Peut être configuré de nombreuses manières pour répondre à divers besoins.	

Choix de la technologie

1. **Séparation des préoccupations** :
    - **Nginx** : excellente gestion des fichiers statiques, libérant Django de cette responsabilité.

    Performances :
    - **Nginx** : Hautement, optimise pour servir du contenu statique et possède des capacités de mise en cache.
2. **Gestion des connexions** :
    - **Nginx** : conçu pour gérer un grand nombre de connexions simultanées.
3. **Sécurité** :
    - **Nginx** : Ajoute une couche de protection, gère le chiffrement SSL/TLS et peuvent bloquer certaines requêtes malveillantes.
4. **Flexibilité** :
    - **Nginx** : Règles de routage complexes et intégration facile d'autres technologies.
5. **Simplification du développement** :
    - **Nginx** : Uniformité entre les environnements de développement et de production.

Conclusion :

Nginx a été choisi en raison de sa capacité à gérer efficacement un grand nombre de connexions simultanées avec une faible utilisation des ressources, ce qui le rend idéal pour les applications modernes et les environnements à haute disponibilité. De plus, sa capacité à servir rapidement du contenu statique et sa performance en tant que serveur proxy inverse, en particulier pour le load balancing, le rendent particulièrement adapté aux applications web modernes, où ces fonctionnalités sont souvent nécessaires. La configuration claire et la faible empreinte mémoire de Nginx, combinée à ses performances élevées, ont motivé ce choix par rapport à Apache.

Base de données

Analyse du besoin

Le client n'a pas un très grand nombre de données à stocker, il n'a donc pas besoin d'une base de données spécialement performante.

Il a par contre besoin de pouvoir stocker des images et des fichiers, mais ceux-ci seront directement enregistrés sur la machine hébergeant le site web et la DB.

Ayant décidé de nous baser sur le Framework Django, nous nous limiterons aux bases de données officiellement prises en charge et gratuites.

Nous exclurons donc Oracle de la comparaison.

Nous utiliserons le mécanisme de bases de Django, à savoir la technique des modèles/ORM et la propagation des modifications via migration

Comparatif

PostgreSQL

Base de données relationnelle open source.

Figure 10logo Postgres

Voici ses avantages :

+ Il a une communauté active et diversifiée
+ Support de SQL complet et conforme aux standards
+ Support de plusieurs indexations et recherche de données avancée
+ Possibilité de l'utiliser pour un service de localisation
+ Traitement rapide des données
+ Support des types de données riches tels que les tableaux et les objets JSON

Inconvénients :

- Installation et réplication compliquée
- Méthode parfois complexe

SQLite

Il s'agit d'une base de données relationnelle open source, voici ses avantages :

Figure 11logo sqlite

+ Assez léger
+ Bonne performance que c'est au niveau de l'écriture ou de la lecture
+ Pas d'installation nécessaire
+ Fiable parce qu'il met à jour son contenu de manière continue
+ Facile à étendre

Inconvénients :

- Taille de la base de données limitée à 2 GO
- Support des types de données riches tels que les tableaux et les objets JSON
- Support des types de données riches tels que les tableaux et les objets JSON

MySQL

Il s'agit d'une base de données relationnelle non open source qui est géré par Oracle.

Figure 12Logo Mysql

Voici ses avantages :

+ Distribué gratuitement
+ Support des types de données riches tels que les tableaux et les objets JSON
+ Possède des privilèges de haute sécurité
+ Capable de gérer de gros volumes de données
+ Réponds rapidement aux requêtes
+ Peu de ressources nécessaires pour son fonctionnement donc faible cout
+ Support des types de données riches tels que les tableaux et les objets JSON
+ Stable

Ses inconvénients :

- Nécessite une grande capacité de stockage en mémoire selon l'utilisation
- Support des types de données riches tels que les tableaux et les objets JSON
- Le fonctionnement dépend de la connexion au serveur.
- Nécessite une licence pour être utilisée pour de vrais projets.
- Support des types de données riches tels que les tableaux et les objets JSON

Choix de la technologie

Pour ce projet, SQLite a été sélectionné comme DB pour plusieurs raisons :

1. Intégration native avec Django : SQLite est le système de base de données par défaut pour les projets Django. Sa facilité d'intégration signifie qu'il n'y a pas besoin de configurations supplémentaires pour commencer à l'utiliser, ce qui est idéal pour un développement rapide.

2. Légèreté et simplicité : SQLite est une base de données serveur-less, ce qui signifie qu'elle ne nécessite pas de processus serveur séparé pour fonctionner. Tout est stocké dans un fichier unique, rendant le déploiement et le déplacement de la base de données d'une machine à une autre extrêmement simple.

3. Idéal pour le développement et les tests : En raison de sa simplicité et de son intégration, SQLite est parfait pour le développement local. Il permet une mise en place rapide de l'environnement de développement, des tests et des prototypes.

4. Contrainte de temps : Lors de la réalisation d'un projet, le temps est souvent une contrainte majeure. La mise en place d'autres bases de données plus complexes, telles que PostgreSQL ou MySQL, peut nécessiter une configuration supplémentaire, des ajustements et des optimisations qui consomment un temps précieux. Avec SQLite, la mise en route est quasi instantanée, permettant ainsi de consacrer plus de temps à la logique métier et au développement des fonctionnalités.

5. Efficacité pour les projets de petite à moyenne envergure : Si le projet n'a pas besoin de gérer un très grand volume de transactions ou de données simultanées, SQLite peut s'avérer suffisant. Il peut gérer une quantité surprenante de requêtes et de données pour sa taille.

Limitations :
Il convient toutefois de noter que SQLite présente certaines limitations, en particulier lorsqu'il s'agit de gérer un grand nombre de requêtes concurrentes, ce qui peut le rendre moins idéal pour des applications de grande envergure ou à fort trafic.

Cependant, pour les besoins actuels de ce projet et compte tenu de la contrainte de temps, SQLite s'est avéré être le choix le plus judicieux.

En conclusion, le choix de SQLite pour ce projet Django a été principalement motivé par la simplicité, la rapidité et la facilité d'intégration qu'il offre, en particulier lorsqu'on est confronté à des contraintes de temps. Il offre un excellent point de départ et peut toujours être remplacé par des bases de données plus robustes à mesure que le projet évolue et que les besoins changent.

Présentation de la solution

Implémentation

Chronologie et méthodologie de la mise en place du projet

Dans le cadre de mon TFE, j'avais le désir profond de concrétiser ce projet qui m'est cher. Mon premier objectif était d'obtenir l'approbation de mon sujet par mon rapporteur, tout en veillant à répondre aux critères de l'EPHEC en matière de TFE. Par conséquent, j'ai entrepris la rédaction d'un cahier des charges.

Des rencontres avec le client ont été organisées dès cette première étape, pour clairement identifier ses besoins et les fonctionnalités attendues pour le site web. Après avoir partagé ce cahier des charges avec mon rapporteur et effectué quelques modifications nécessaires, le sujet a été finalement validé.

Suite à cette approbation, j'ai approfondi la compréhension des besoins et organisé des réunions toutes les 2 semaines avec le client. Ces échanges ont permis de dessiner le schéma de la base de données et de concevoir les maquettes des interfaces du site. Une fois le projet lancer, un Trello , Clockify et Github ont été mis en place afin de me permettre de travailler dans un environnement idéal.

Ayant des restrictions certaines au niveau du temps j'ai été dans l'obligation de définir un horaire de travail clair.

Tableau 8 horaire semaine

<table>
<tr><th></th><th>Lundi</th><th>Mardi</th><th>Mercredi</th><th>Jeudi</th><th>Vendredi</th><th>Samedi</th><th>Dimanche</th></tr>
<tr><td>08:30-10:30</td><td colspan="5" rowspan="2">Travail à temps plein Bois de la Pierre Wavre</td><td rowspan="2">Temps personnel</td><td>Temps personnel</td></tr>
<tr><td>10:30-12:30</td><td>TFE</td></tr>
<tr><td>12:30-13:30</td><td colspan="7">Temps de Midi</td></tr>
<tr><td>13:30-16:30</td><td colspan="5">Travail à temps plein Bois de la Pierre Wavre</td><td rowspan="2">TFE</td><td>TFE</td></tr>
<tr><td>17:00-19:00</td><td colspan="2">TFE</td><td></td><td>TFE</td><td></td><td></td></tr>
<tr><td>19:00-21:00</td><td colspan="2">Temps personnel / Rattrapage TFE</td><td>Gestion Unitée scout</td><td>Temps personnel</td><td>1 fois toutes les 2 semaines
RDV Clients</td><td colspan="2">Temps personnel</td></tr>
</table>

J'ai tout d'abord passé un certain temps à étudier et apprendre le fonctionnement des technologies employé afin de proposer des solutions techniques aux nombreuses demandes du client.
Une fois cette étape passée, j'ai commencé par mettre en place le serveur Django en développement.
Une fois le squelette du projet mis en place, le serveur React a été mis en place.
Une fois ces éléments en place, le développement du site a véritablement démarré.

Après chaque création de composants ou de modèles, un test complet du fonctionnement de chaque partie était réalisé afin de vérifier le bon fonctionnement du code.

La méthodologie adoptée pour ce projet s'inspire de l'approche AGILE. J'ai constamment adapté le projet en fonction des retours du client, avec des échanges réguliers et des démonstrations lorsque cela était pertinent.
Cependant il est à noter que le projet imaginé par le client a fortement évolué entre le début de celui-ci et la date ou je rédige ce rapport. De ce fait certaines des demandes ont été modifiées, revues à la hausse ou revues à la baisse. Certaines parties du code ont été abandonnées au profit de partie plus ''urgente''.

Outils utilisés

Au cours de l'élaboration de ce TFE, j'ai mobilisé une panoplie d'outils, chacun ayant une influence spécifique sur la mise en œuvre du projet et la coordination avec les différents acteurs.

- **PyCharm** : Cet IDE m'a servi à concevoir les différentes interfaces de mon site. Grâce à sa capacité de « développement à distance », j'ai pu programmer directement sur le VPS via SSH. Cette approche m'a facilité les tests locaux, m'épargnant des cycles de commit. De plus, travaillant sur un système Windows, cela m'a évité des soucis de compatibilité.
- **Github Copilote,**
  La société dans laquelle je travaille m'ayant fourni un accès a Github Copilote, j'ai pu utiliser cet « aide de travail » afin d'optimiser le temps que je passais à développer
- **Facebook Messenger / Discord** : Étant ami avec mon client, Messenger était notre canal de prédilection pour planifier des réunions et obtenir des réponses en temps réel.
- **Clockify** : Cet instrument m'a permis de suivre le temps consacré à chaque segment du projet.
- **Github**: j'ai utilisé Github pour la gestion des versions de mon travail.
- **Trello** : Cette plateforme de gestion de tâches m'a aidé à surveiller l'évolution de mon projet
- **Putty & WinSCP** : j'ai employé cette console pour accéder à mon VPS à distance et effectuer des opérations essentielles au TFE.
- **Internet** : Étant encore en apprentissage sur l'utilisation de certaines technologies je me suis souvent retrouvé face à une erreur que je n'arrivais pas à comprendre, l'accès à internet m'a permis de me tourner vers des personnes plus compétentes ou ayant eu les mêmes problèmes que moi.
  Il est important de noter qu'internet est vaste et nombre d'heures sur le projet ont été passées à écumer des forums en quête de réponses sans pour autant réussir à les trouver.

Architecture du système

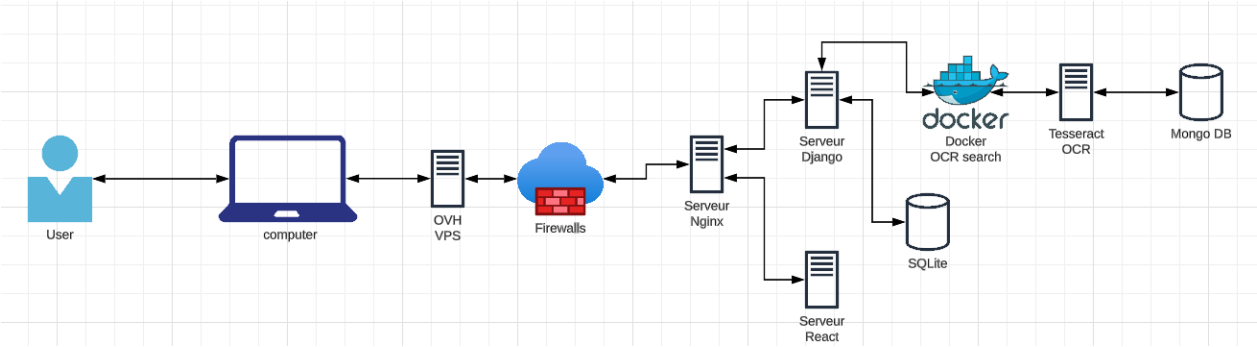

Figure 13 Architecture

Présentation du site Web

L'architecture du site web repose sur trois principaux serveurs :

- **Serveur NGINX** : Il sert principalement de reverse proxy, assurant le routage des requêtes vers les serveurs appropriés, qu'il s'agisse de Django ou de React. NGINX sert également les fichiers statiques et médias, comme les scripts JavaScript et les feuilles CSS.
- **Serveur Django** : Ce serveur traite directement les requêtes des utilisateurs. Il interagit avec la base de données SQLite pour récupérer ou stocker des données, et avec le conteneur Docker pour effectuer des opérations OCR via Tesseract.
- **Serveur React** : Il s'occupe de l'interface utilisateur et de la présentation. Les interactions côté client sont traitées ici, avant d'être envoyées au serveur Django pour le traitement côté serveur.

Concernant la persistance des données, étant donné que j'utilise SQLite, il n'y a pas de conteneur séparé pour la base de données comme dans une configuration plus traditionnelle avec, par exemple, PostgreSQL. Au lieu de cela, SQLite stocke tout dans un fichier unique, simplifiant la gestion des données.

Un conteneur Docker distinct est utilisé pour Tesseract, permettant à Django d'effectuer des recherches OCR via ses API.

Schéma DB

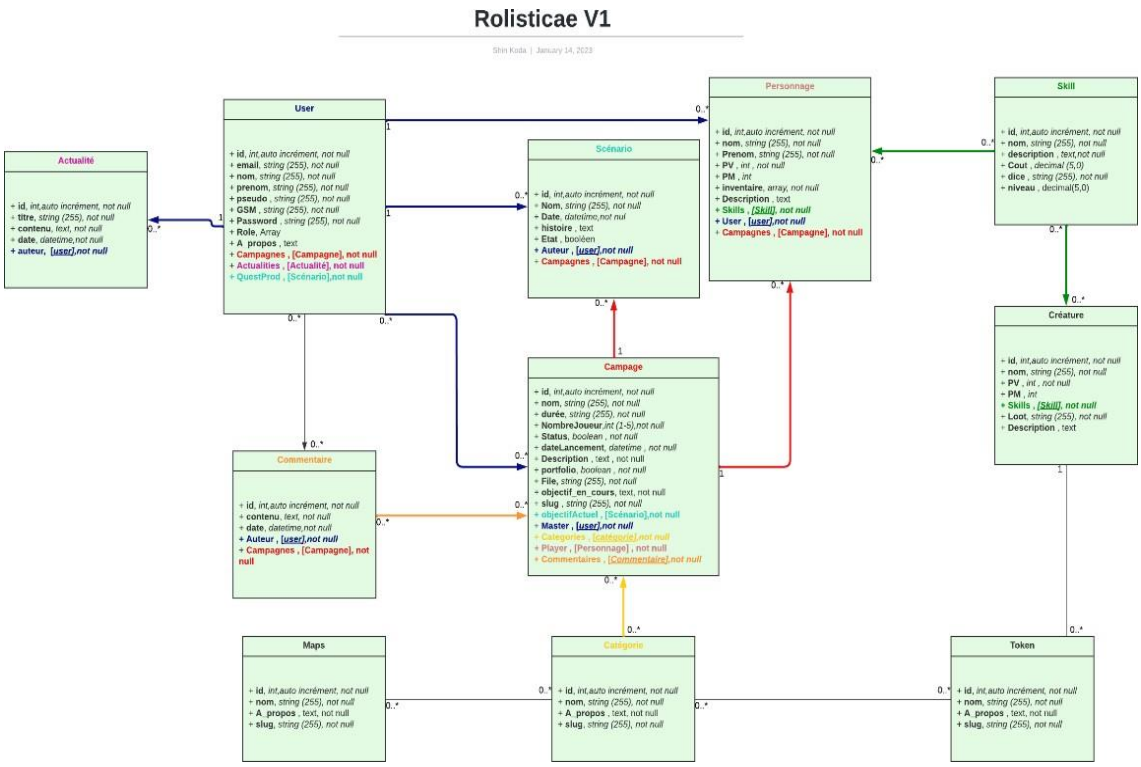

Figure 14 Database Schema

Sécurité

Site web : Analyse des menaces et mesures de sécurité mises en place

HTTPS et SSL

L'utilisation du protocole HTTP présente un risque, car les informations échangées entre le serveur et le client sont transmises sans cryptage. Cela pourrait permettre à une personne malintentionnée d'accéder à des informations sensibles telles que les mots de passe et les noms d'utilisateur, ainsi qu'aux données relatives à la soumission de formulaires. Il est donc essentiel de rediriger tout le trafic du site via le protocole HTTPS. Pour cela, il est nécessaire de générer un certificat SSL valide. Django et Nginx disposent de mécanismes permettant de rediriger tout le trafic entrant utilisant le protocole HTTP vers le protocole HTTPS.
HTTPS mis en place via Letsencrypt [28]

Injection SQL

L'injection SQL permet à des personnes malintentionnées d'introduire des requêtes SQL dans le but d'accéder ou d'altérer la base de données. Cependant, le Framework Django a mis en place un système de formulation de requêtes SQL qui prévient ces attaques. Django élabore les requêtes SQL indépendamment des paramètres fournis par l'utilisateur lors d'une sollicitation au serveur web.

CSRF

Les attaques CSRF visent à usurper l'identité d'un utilisateur lors de l'envoi d'une requête POST par un client. Pour contrer ces attaques, Django intègre un Token propre à chaque utilisateur, empêchant ainsi l'usurpation d'identité. Toutefois, si le trafic passe par le protocole HTTP, il est envisageable d'intercepter ce Token, rendant caduque la protection offerte par Django. D'où l'importance de privilégier le protocole HTTPS pour sécuriser le trafic.

Pour mettre en œuvre cette vérification de token dans Django, il convient d'ajouter le paramètre 'CsrfViewMiddleware' dans le fichier 'settings' et d'insérer la balise '{% csrf_token %}' dans les formulaires
En raison de Problème lors de la mise en production , cette protection a été suspendue

Firewall

Pour filtrer les communications du serveur, une configuration de Firewall est essentielle. C'est pourquoi j'ai opté pour UFW, facilitant la mise en place des règles d'IPtables. UFW, en tant que mécanisme de Firewall épuré, met en œuvre les règles d'IPtables, déjà présentes sur Ubuntu. Les ports autorisés comprennent les ports 80 et 443 dédiés à HTTP et HTTPS, le port spécifié pour le protocole SSH, ainsi que les ports 8000 et 3000. Le port 8000 est privilégié, car utilisé par le serveur de développement Django, tandis que le port 3000 est dédié à React, permettant une reconstruction du site à chaque changement. Il est à noter que sans l'activation de ce dernier port, l'accès au serveur depuis l'extérieur serait impossible.

Bilan du Projet

Avant de terminer, j'aimerais faire un récapitulatif des leçons que j'ai tirées de ce projet, des améliorations potentielles et des éléments que je n'ai pas pu mettre en œuvre en raison de contraintes de temps. Pour conclure cette section, je décrirai les prochaines étapes envisagées pour ce projet.

Apprentissage résultant de la création de ce TFE

Compétence technique

Premièrement, je vais aborder le sujet du développement web.

Avec une connaissance préalable du langage Python, spécifiquement du Framework Django, je ne m'attendais pas initialement à apprendre autant.

Cependant, j'ai été surpris par la quantité de recherches que j'ai dû mener pour concevoir le site web selon les besoins du client. Les diverses compétences que j'ai acquises lors de mes études à l'EPHEC et de mon stage à la Clinique du Bois de la Pierre ont été indéniablement utiles pour développer et concevoir le site web, mais c'était la première fois que je faisais face réellement à la complexité de concevoir un site web entièrement par moi-même. Et sans l'aide de nombreux internautes et les membres du service IT de la clinique, je n'aurais jamais autant progressé.

Bien que je sois conscient d'avoir encore beaucoup à apprendre pour maîtriser le Framework Django et le langage Python, je considère avoir énormément progressé durant ce semestre.

Je crois aussi avoir été trop obstiné à vouloir tout mettre en œuvre sans les conseils de personnes éclairées, et j'aurais dû chercher des conseils auprès de collègues plus expérimentés ou de professeurs. À cause de cet entêtement, j'ai pris énormément de temps sur des erreurs risibles.

Deuxièmement, la conception de la base de données et des différentes maquettes d'écran m'a permis de renforcer mes compétences en matière d'anticipation des besoins.

Plus précisément, lors de la conception du schéma de base de données, j'ai omis d'identifier certaines contraintes nécessaires, ce qui a entraîné de nombreuses modifications et complications.

Troisièmement, lors de la phase d'analyse, je pensais que choisir les technologies serait simple. Cependant, j'ai rapidement réalisé que cette étape serait plus complexe que prévu. J'ai dû mener des recherches approfondies pour concevoir une architecture fonctionnelle.

Enfin, l'aspect qui m'était totalement inconnu était la pris en charge d'un client et bien qu'étant un ami, cela a été pour moi une véritable épreuve, celui-ci changeant constamment d'avis.

Quand j'ai commencé à coder, je m'attendais à ce que cela prenne beaucoup de temps. Cependant, j'ai réalisé que l'apprentissage et la recherche de bugs ont également été très chronophages. Et, je pense que, tout comme pour le développement web, j'aurais dû chercher des conseils auprès de personnes expérimentées avant de me lancer, cela m'aurait économisé de nombreuses heures.

Compétence liée à la gestion d'un projet

Je suis fier d'avoir su diviser adéquatement les différentes étapes de l'implémentation du TFE. Dans tous mes projets précédents, j'avais l'habitude de démarrer plusieurs tâches simultanément, ce qui entraînait un manque de stratégie et d'efficacité.

Malgré mes difficultés, j'ai en partie réussi à faire fit de ce défaut, car cela n'a presque pas été le cas pour ce projet, surtout lors de la mise en place du site web. Je me suis consacré à une seule grande fonctionnalité à la fois, me permettant de me concentrer exclusivement sur celle-ci jusqu'à son achèvement.

Toutefois, j'ai rencontré des difficultés dans l'analyse des technologies et la justification de mes choix. Bien que j'aie progressé, je réalise que j'éprouve de grandes difficultés à effectuer une analyse cohérente en fonction des besoins du client, ce qui me demande beaucoup de temps et d'énergie.

Je reconnais également qu'il y a eu une des phases assez longues pendant laquelle le projet n'a pas avancé comme il le devait.

Cela était en partie dû à un trouble de l'attention, mais aussi à mes difficultés en matière d'analyse. Néanmoins, je me suis repris et j'ai redoublé d'efforts pour rattraper ce retard. Cette expérience m'a fait comprendre l'importance d'une planification bien pensée et rigoureusement suivie.

Amélioration du projet et élément restant à implémenter

Je vais commencer par dresser une liste des éléments que je n'ai pas encore pu implémenter, du moins pas complètement.

- Les tests unitaires ne couvrent pas l'entièreté des fonctionnalités.
- Il manque plusieurs formulaires au niveau de React.
- La liste d'attente a été mise en pause à la demande du client
- L'affichage n'est pas nickel partout
- Le site n'est pas correctement responsif
- La page d'inscription n'est pas accessible à la demande du client.
- L'outil d'affichages des personnages en cours de partie n'a pas été implémenté

Concernant les améliorations encore possibles, il s'agit surtout de créer et mettre en page des formulaires React. Étant donné la nature du projet et l'apprentissage réalisé durant celui-ci il est surement possible de reprendre le projet à zéro en étant plus efficace grâce aux connaissances apprises durant celui-ci.

Pour le client il s'agit surtout de fonction mineure, mais certaines restent très intéressantes à mettre en place.

Il vient ensuite la question de la sécurité et du RGPD, car je suis conscient des lacunes de mon projet dans ce domaine.

Suite du projet

Comme mentionné plus haut c'est un projet qui me tient à cœur et que je compte utiliser assez fréquemment, il est donc prévu de le maintenir et de l'améliorer tant que nécessaire.

Dans l'immédiat il est déjà utilisé par les joueurs de plusieurs tables et semble satisfaire ceux-ci.
Avec l'aide de leur retour en tant qu'utilisateur je pourrais porter se projet aussi loin que possible.

Conclusion

Au terme de ce TFE, j'ai pu constater l'ampleur de ce que je peux accomplir. La réalisation de cette plateforme destinée à mettre en lumière le talent de Kevin Opese en tant que Maître de Jeu n'a pas été sans embûches. Bien que son succès fulgurant de ces derniers mois ait entraîné certains ralentissements vis-à-vis de certaines fonctionnalités, l'intérêt croissant d'autres maîtres de jeu me conforte dans l'idée que cette initiative a un potentiel immense et prometteur.

Face aux (nombreux) défis rencontrés, j'ai su persévérer, preuve que la méthode apprise et la détermination peuvent nous permettre de dépasser nos limites. L'entrée dans la vie professionnelle, due à mon embauche à temps plein, a été une étape majeure lors de cette année. Elle m'a aussi rappelé l'importance et la complexité de la gestion du temps, notamment dans le contexte exigeant d'un TFE. Toutefois, ce parcours m'a permis de réaliser à quel point chaque second compte et combien il est crucial d'optimiser ses moments de travail.

Au cours de ce travail, les phases de débogage ont été parmi les plus éprouvantes. Passer d'innombrables heures à traquer des erreurs, souvent subtiles, m'a appris la patience et la rigueur nécessaires en développement. Plus d'une fois, je me suis retrouvé à douter, à remettre en question non seulement mon travail, mais aussi ma motivation. Ces périodes de doute m'ont aussi rappelé l'importance de prendre du recul, de faire une pause lorsque la solution semble insaisissable, pour ensuite revenir avec un esprit frais et une perspective renouvelée.

Il est vrai que lors de ces moments de défi, ma motivation a vacillé. Les longues sessions de travail, les impasses et les retours en arrière ont parfois engendré des pertes d'intérêt temporaires. Cependant, chaque obstacle surmonté m'a redonné une bouffée d'énergie et de détermination. J'ai appris que dans chaque projet, il est naturel d'avoir des hauts et des bas, et que le plus important est de rester focalisé sur l'objectif final, de se rappeler la raison pour laquelle on a entrepris ce voyage.

Je reconnais que si j'avais eu davantage de temps à consacrer à ce projet, le rendu aurait pu être encore plus abouti. Néanmoins, je reste fier de ce que j'ai accompli et je suis reconnaissant envers l'EPHEC pour les connaissances et les compétences qu'elle m'a apportées. Ces enseignements, je compte bien les mettre au service de la continuité de ce projet.

Enfin, je tiens à remercier sincèrement toutes les personnes qui m'ont soutenu tout au long de ce projet. Vos conseils, votre patience, et votre foi en moi ont été des piliers essentiels. Sans vous, ce travail n'aurait pas été le même. Je suis éternellement reconnaissant pour chaque moment de soutien, chaque conseil et chaque encouragement. Alors que je regarde vers l'avenir, je suis excité à l'idée des nouveaux défis qui m'attendent, et je suis certain que les leçons apprises lors de ce TFE me serviront dans tous mes futurs projets.

Bibliographies

[1] Google Trends JDR
https://trends.google.fr/trends/explore?date=today%201-m&q=JDR&hl=fr
[En ligne].

[2] Django. «Django makes it easier to build better web apps more quickly and with less code
https://www.djangoproject.com/
[En ligne].

[3] Wikipedia. «Django (web framework)».
https://en.wikipedia.org/wiki/Django_(web_framework)
[En ligne].

[4] Django. «Django - Documentation - Bases de données»
https://docs.djangoproject.com/fr/4.0/ref/databases/ .
[En ligne]

[5] Mobiskill. «Pourquoi utiliser Django pour du développement web ?».
https://mobiskill.fr/blog/conseils-emploi-tech/pourquoi-utiliserdjango-pour-du-developpement-web/
[En ligne].

[6] Python Doctor, «ORM Django»
https://python.doctor/page-django-ormapprendre-base-donnees-database-queryset-modeles
[En ligne].

[7] IONOS, «Flask vs. Django : comparatif des deux frameworks».
https://www.ionos.fr/digitalguide/sites-internet/developpement-web/flask-vsdjango/
[En ligne].

[8] Statistics and data. «Most Popular Backend Frameworks (2012/2022)».
https://www.youtube.com/watch?v=-RTaFJAgWSU
[En ligne].

[9] Cosmos Data. «Most Popular Programming Languages 1965-2023 ».
https://www.youtube.com/watch?v=jC_0NxJBn4A
[En ligne].

[10] Clark, Jessica. Back4app. «Les 10 meilleurs Frameworks de développement pour les startups».
https://blog.back4app.com/fr/les-10-meilleursframeworks-de-developpement-pour-les-startups/
[En ligne].

[11] Django. «Processus de publication de Django»
https://docs.djangoproject.com/fr/4.1/internals/release-process/
[En ligne].

[12] Flask VS Django
https://kinsta.com/fr/blog/flask-vs-django/#:~:text=Django%20dispose%20d'un%20large%20%C3%A9ventail%20de%20fonctionnalit%C3%A9s%20int%C3%A9gr%C3%A9es.&text=Flask%20est%20l%C3%A9ger%20et%20plus,effort%20de%20projets%20plus%20importants.
[En ligne].

[13] Mobiskill. «Flask vs Django : Quel framework choisir ?»
https://mobiskill.fr/blog/conseils-emploitech/flask-vs-django-quel-framework-choisir/
[En ligne].

[14] Mobiskill. «Pourquoi utiliser Python pour du développement web ?». Mise à jour le 26 08 2021. [En ligne]. https://mobiskill.fr/blog/conseils-emploi-tech/pourquoi-utiliser-python-pourdudeveloppement-web/
[En ligne].

[15] Flask, «Flask Documentation (1.1.x)».
https://flask.palletsprojects.com/_/downloads/en/1.1.x/pdf/
[En ligne].

[16] Node.js Official About Page
https://nodejs.org/en/about
[En ligne].

[17] V8 Official Page
https://v8.dev/
[En ligne].

[18] npm Official Website
https://www.npmjs.com/
[En ligne].

[19] React Official Documentation.
https://legacy.reactjs.org/docs/introducing-jsx.html
[En ligne].

[20] React Training.
https://reactrouter.com/en/main
[En ligne].

[21] GitHub Repository.
https://github.com/facebook/react
[En ligne].

[22] React Native Official Documentation.
https://reactnative.dev/
[En ligne].

[23] Angular Official Documentation.
https://angular.io/docs
[En ligne].

[24] Packt Publishing.
https://www.packtpub.com/
[En ligne].

[25] TypeScript Official Documentation
https://www.typescriptlang.org/
[En ligne].

[26] GitHub Repository
https://github.com/angular/angular
[En ligne].

[27] Angular CLI Documentation.
https://cli.angular.io/
[En ligne].

[28] Let's Encrypt
https://letsencrypt.org/fr/
[En ligne].

[29] Trello
https://trello.com/invite/b/dOkBSPsT/ATTI7fe9c2426c3005098e5f857ddb503ec0D95E6CA3/rolisticae
[En ligne].

[30] Github
https://github.com/Derrevale/Rolisticae
[En ligne].

[31] Clockify
https://app.clockify.me/reports/summary?start=2023-01-01T00:00:00.000Z&end=2023-12-31T23:59:59.999Z
[En ligne].

[32] TFE
https://www.rolisticae.fun/
[En ligne].

https://www.rolisticae.fun/

Table des Illustrations