RAPPORT DE MISSION

→ STEFANOS ←

INTRODUCTION

• Type de projet : Création d'une application web.

• Description : Afin de se lancer dans la publication de pronostics sportifs par abonnement, l'équipe de Stefanos a fait appel à moi afin de développer une application web permettant de répondre à ce besoin.

• Client : Stefanos.

• Durée : du 04/07/2022 au 25/07/2022.

• Contexte : Freelance.

CLIENT

• Courte présentation du client :

Suivis par plus de 15000 abonnés sur les réseaux sociaux, Stefanos est spécialisé dans le paris sportif et notamment dans le tennis.

• Besoin :

- Espace de mise en ligne et de gestion des pronostics sportifs permettant de pouvoir restreindre leur accès selon plusieurs niveaux d'abonnement.
- Panel administrateur permettant de gérer les utilisateurs, les abonnements et de démarrer un Grand Chelem.
- Chaque tournoi doit être associé à un abonnement à durée limitée.
- Espace membre permettant de voir/gérer ses informations personnelles ainsi que ses abonnements.
- Page d'accueil présentant l'entreprise.
- Possibilité de se connecter avec Google.
- Les utilisateurs doivent pouvoir s'abonner pour une durée d'un mois, renouvelable manuellement, ou pour la durée de l'événement en cours.
- L'application web doit s'adapter aux mobiles et tablettes.

• Objectifs :

- Rendre la publication des pronostics sportifs payante afin de commencer à générer des profits.
- Se professionnaliser et étendre sa communauté.

• **Contraintes :**
- L'application doit absolument être mise en ligne avant le 31/08/2022.
- L'application doit être développée comme SPA (Single Page Application).

PROJET

• **Arborescence** :

1. Page d'accueil
    a. A propos
    b. Abonnements
        i. Page de paiement
            1. Success/error page
    c. Résultats
    d. Contact
2. Pronostics
    a. V.I.P
    b. Evènement
3. Me connecter
    a. Se connecter
    b. S'inscrire
4. Mon compte
    a. Mes informations personnelles
    b. Mes abonnements
5. Administration
    a. Gérer les événements
    b. Gérer les utilisateurs
    c. Gérer les abonnements

• **Environnement technologique :**

- Front-end (SPA React.js) :

  _ React.js : librairie javascript orientée composants, React permet de développer les Single Page Applications. Elle permet donc de concevoir des applications web fluides avec des temps de chargement très courts voire inexistants.

  _ Redux : tout comme la React Context API, Redux permet de mettre en place un store, rendant accessibles les données partout et à tout moment dans l'application.

  _ Bootstrap & Material UI : frameworks CSS, ils permettent de rendre très facilement les sites web responsives (via grid system) et mettent à disposition différents types de composants rendant plus simple la conception des interfaces.

  _ Sass : grâce à Sass, le codage des styles est beaucoup plus simple, en permettant notamment l'indentation, les variables, les boucles...

- Back-end (API REST Node.js) :
  _ Node.js & Express : environnement d'exécution javascript, Node.js permet d'exécuter du javascript à l'extérieur du navigateur et donc de développer des API. Il offre de très bonnes performances grâce à ses opérations non-bloquantes.
  _ Passport.js : afin de gérer l'authentification de manière optimisé, Passport.js est une très bonne solution. Middleware pour Node.js, il permet l'authentification utilisant FaceBook, Google, Twitter...
  _ Mongoose : ODM (Object Data Modeling), il permet de mapper les données d'une base de données NoSQL grâce à ses fonctions embarquées, la définition de modèles...
  _ Nodemailer : utilisé pour l'envoi de mails via Google SMTP.
  _ Stripe API : afin de gérer plus facilement les paiements, j'ai fait le choix de Stripe qui permet de créer des paiements et abonnements, des produits et propose des statistiques détaillées.
  _ Notion API : dans cette application, Notion a servi de base de données pour les différents pronostics. Bien paramétré, il est un véritable CMS. Les pronostics sont donc rentrés sur une base de données Notion puis envoyés à l'API qui transmet le tout au front-end.
- Base de données :
  _ MongoDB (NoSQL) : les différentes données étant croisées et non structurées, j'ai choisi une base de données NoSQL, orientée document, qui me permet de croiser les données plus facilement en ne suivant pas toujours les mêmes modèles.
- Gestion de versions :
  _Git/GitHub : afin de gérer correctement les différentes versions de mon code et ainsi me permettre de travailler plus sereinement sur le projet, l'entièrement de l'application (front et back) a été versionné via GitHub.

• Logiciels utilisés :
- WebStorm comme IDE pour le développement des interfaces et de l'API.
- DataGrip comme IDE de base de données pour la visualisation des données.
- FileZilla pour le transfert de données sur le serveur.

• Déploiement :
- Instance AWS EC2 (Ubuntu) pour l'hébergement de l'application React et de l'API.
- MongoDB Atlas pour la base de données NoSQL.
- AWS Route 53 pour le nom de domaine et la sécurité (HTTPS).

BILAN

Ce projet a été réalisé dans le cadre de mon activité en tant que développeur indépendant, sur une période de trois semaines. Mon rôle était d'effectuer un travail

de qualité dans les temps impartis, répondant parfaitement aux besoins émis par le client.

Travaillant seul et encore étudiant, les différents problèmes auxquelles j'ai dû faire face m'ont poussé à effectuer des recherches et à persévérer sur des erreurs prenant parfois plusieurs heures à être réglées.

Avancer seul sur un projet fullstack et from-scratch requiert beaucoup de temps, de recherches, de discipline et de persévérance et c'est pourquoi travailler sur ce projet a été enrichissant pour moi.