

Book My Gigs

Dossier Projet

31678 - Concepteur développeur d'applications

Novembre 2024

Gabriel PARIZET

Présentation du projet6

I) Project Overview

The Project

BOOK MY GIGS was born from my personal experience as an electronic music enthusiast. Faced with a constantly thriving French and European music scene, I never found a unique platform to discover all musical events happening around me. The issue is simple but problematic: information is scattered, making the discovery of events tedious and time-consuming.

This fragmenting of information is getting even worse with the diversification of communication channels. Facebook Events, once central to event organization, is seeing a decline in usage, while announcements are dispersed across Instagram, Shotgun, and other specialized platforms. However, none manage to offer a comprehensive view of the music scene's offer.

It is within this context that I developed Book My Gigs, an application that centralizes and simplifies access to musical events. It allows users to list and share events from different sources, organizing them intuitively by location, music genre, and type of event (Club, Concert, Festival). The goal is to make it easier to discover events that match individual tastes, while enabling organizers to reach an engaged and passionate community.

The project presented here is a minimum viable product (MVP). We will strive to address the future of Book My Gigs in this file. For its development, I chose a robust tech stack combining Elixir and Phoenix on the backend (offering excellent scalability for real-time applications) with React.js and Tailwind CSS on the frontend. This solid technical foundation will allow the application to be gradually enriched with new features to meet the evolving needs of the community.

I) Présentation du projet

Le Projet

BOOK MY GIGS est née de mon expérience personnelle en tant que passionné de musique électronique. Face à une scène française et européenne en perpétuelle effervescence, je me suis heurté à l'absence d'une plateforme unique permettant de découvrir tous les événements musicaux qui ont lieu autour de moi. Le constat est simple mais problématique : l'information est dispersée, rendant la découverte d'événements fastidieuse et chronophage.

Cette fragmentation de l'information s'accentue avec la diversification des canaux de communication. Facebook Events, autrefois central dans l'organisation d'événements, voit son utilisation décliner, tandis que les annonces se dispersent entre Instagram, Shotgun et d'autres plateformes spécialisées. Pourtant, aucune ne parvient à offrir une vue d'ensemble satisfaisante de la scène musicale.

C'est dans ce contexte que j'ai développé Book My Gigs, une application qui centralise et simplifie l'accès aux événements musicaux. Elle permet aux utilisateurs de répertorier et de partager des événements issus de différentes sources, en les organisant de manière intuitive par localisation, genre musical et type d'événement (Club, Concert, Festival). L'objectif est de faciliter la découverte d'événements correspondant aux goûts de chacun, tout en permettant aux organisateurs de toucher une communauté engagée et passionnée.

Le projet présenté ici est un produit minimum viable (MVP). Nous tâcherons de parler du futur pour Book My Gigs dans ce dossier. Pour son développement, j'ai choisi une stack technique robuste combinant **Elixir** et **Phoenix** en backend (offrant une excellente scalabilité pour les applications temps réel) avec **React.js** et **Tailwind CSS** en frontend. Cette base technique solide permettra d'enrichir progressivement l'application de nouvelles fonctionnalités pour répondre aux besoins évolutifs de la communauté.

I.A Courte présentation



Gabriel PARIZET

Apprentice Backend Developer chez Welcome To The Jungle

<https://www.linkedin.com/in/gabriel-parizet/>

<https://github.com/Gabrielparizet>

“Ancien sommelier, ma curiosité m'a naturellement poussé à explorer de nouveaux horizons dans le développement web. Après une formation intensive en programmation, j'ai eu l'opportunité d'effectuer mon alternance chez Welcome To The Jungle, où j'ai consolidé mes compétences techniques. Cette reconversion m'a permis de découvrir une nouvelle forme de création, tout en conservant mon goût pour l'innovation et le partage.”

II) Outils

II.A : Environnement de développement

- Terminal : Iterm2 & Zsh

J'ai opté pour ITerm2 comme terminal, couplé avec Zsh comme shell et personnalisé via Oh-My-Zsh. Cette combinaison m'a offert une expérience de ligne de commande plus efficace grâce à la coloration syntaxique, l'autocomplétion intelligente et un système de plugins puissant.

- IDE : Visual Studio Code

J'ai choisi Visual Studio Code comme environnement de développement intégré pour sa polyvalence et sa légèreté. Son large écosystème d'extensions m'a permis d'adapter l'IDE à mes besoins spécifiques, notamment pour Elixir et React. L'intégration native avec Git a facilité la gestion de versions directement depuis l'éditeur, tandis que la personnalisation poussée de l'interface et des raccourcis clavier a optimisé mon flux de travail quotidien.

- POSTMAN

J'ai utilisé Postman pour tester et documenter mes API endpoints tout au long du développement. Cet outil m'a permis de visualiser facilement les réponses de mes requêtes HTTP, de configurer mes headers d'authentification et de valider le format des données envoyées et reçues. En organisant mes requêtes en collections, j'ai pu maintenir une documentation pratique de mon API et réutiliser efficacement mes tests lors des différentes phases de développement.

- POSTICO 2

J'ai utilisé Postico 2 comme interface graphique pour gérer ma base de données PostgreSQL. Cet outil m'a permis de visualiser et manipuler facilement mes données, de vérifier la structure de mes tables et d'exécuter des requêtes SQL complexes. Son interface intuitive m'a été particulièrement utile pour déboguer et valider mes schémas de base de données pendant le développement.

The screenshot shows the Postico 2 interface connected to the "book_my_gigs_dev" database. The left sidebar lists tables: accounts, events, events_genres, genres (which is selected), locations, schema_migrations, types, users, users_genres, pg_catalog, and information_schema. The main area displays the "genres" table with the following data:

	id	name	inserted_at	updated_at
	35851a74-794a-4856-9882-1f674c11e477	Symphonic Black Metal	2024-11-13 17:13:59	2024-11-13 17:13:59
	9ae20546-444e-4eb5-80bb-1f03cb4c21bf	Symphonic Metal	2024-11-13 17:13:59	2024-11-13 17:13:59
	1add4818-c96c-425e-a60c-2673a0334d4e	Symphonic Rock	2024-11-13 17:13:59	2024-11-13 17:13:59
	77cdb533-8813-400d-bc4e-b446cac23dad	Synthpop	2024-11-13 17:13:59	2024-11-13 17:13:59
	df3f60e7-4cf4-40ac-b6e7-5788c00bfd35	Taiwanese Pop	2024-11-13 17:13:59	2024-11-13 17:13:59
	334b79af-5a8e-40f6-af51-02c0faf420b6	Talent Show	2024-11-13 17:13:59	2024-11-13 17:13:59
	515e49ba-c785-482b-8e18-cc8c5ececc2e	Tango	2024-11-13 17:13:59	2024-11-13 17:13:59
	18a58ab6-6016-480f-b608-2cea90063453	Tech House	2024-11-13 17:13:59	2024-11-13 17:13:59
	65c4ce84-df02-4553-8a7d-41252d95d3a2	Technical Brutal Death Metal	2024-11-13 17:13:59	2024-11-13 17:13:59
	d6a36e1f-a1b5-4436-9c59-605c9a0e9a3b	Technical Death Metal	2024-11-13 17:13:59	2024-11-13 17:13:59
	edccf94f-a3e9-4fbf-ad92-d06d5c459407	Techno	2024-11-13 17:13:59	2024-11-13 17:13:59
	8d6c108a-356d-4b31-b4f8-720dffc31bb9	Teen Pop	2024-11-13 17:13:59	2024-11-13 17:13:59
	9e0e0122-01fa-4ef3-8cdf-13eb2f18cd8b	Tejano	2024-11-13 17:13:59	2024-11-13 17:13:59
	7b99f38c-0086-44e1-8db6-dfce7beafe8f	Tekno	2024-11-13 17:13:59	2024-11-13 17:13:59
	f44f6bac-f7a5-45c7-981a-e4ed23975599	Terrorcore	2024-11-13 17:13:59	2024-11-13 17:13:59
	4e956277-5ea5-4f2f-8efc-b62b0ce4c9b4	Texas Blues	2024-11-13 17:13:59	2024-11-13 17:13:59

At the bottom, there are buttons for Content, Structure, DDL, + Row, Filter, and Page 2 of 2.

I.B : Gestions de projet et de versions

- Github :

Pour assurer un suivi précis des versions et une sauvegarde sécurisée de mon projet, j'ai utilisé Github en séparant mon application en deux repositories distincts : ***book_my_gigs_front*** pour le frontend et ***book_my_gigs*** pour le backend.

Pour chaque nouvelle fonctionnalité, j'ai créé une branche dédiée suivant la structure : **type/sujet/MESSAGE**.

exemple : feat/events/GET_events_API_endpoint

Mes commits ont respecté la convention **type(sujet): message**.

Cette rigueur dans le nommage, associée à l'utilisation systématique des pull requests, m'a permis de maintenir un historique clair et d'assurer la qualité du code avant son intégration dans la branche principale.

https://github.com/Gabrielparizet/book_my_gigs_front

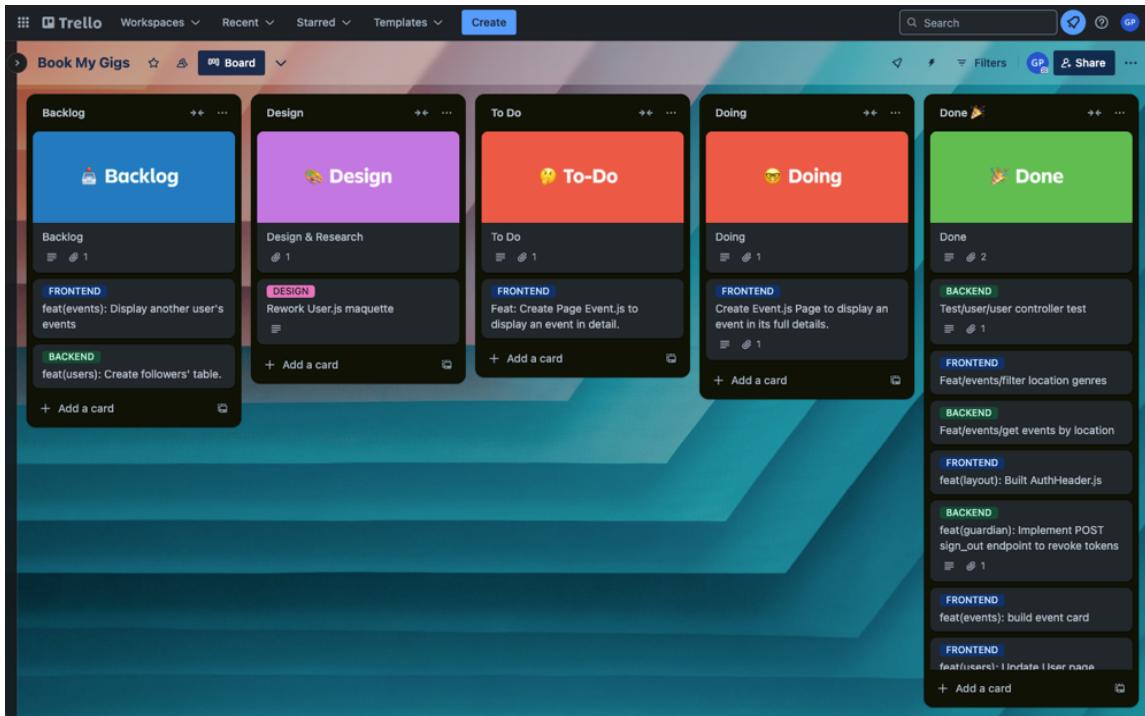
https://github.com/Gabrielparizet/book_my_gigs

The screenshot shows the GitHub repository page for `book_my_gigs`. The repository is public and has 86 commits. The commits are listed in a table, showing the author, commit message, date, and time. The repository has 1 star, 0 forks, and 1 watching. It also shows sections for Releases, Packages, and Languages.

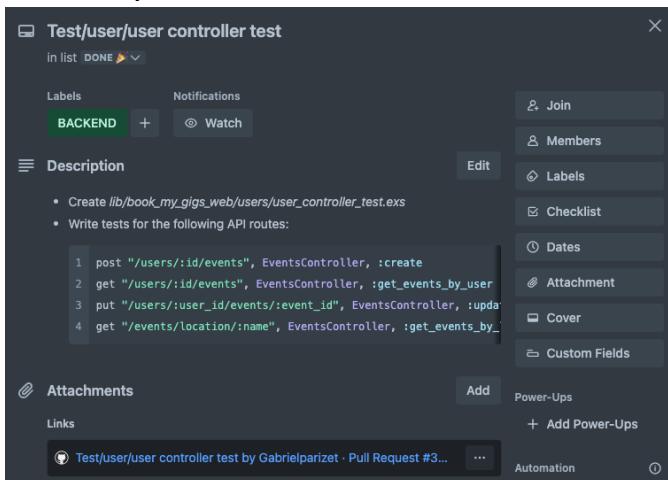
File	Commit Message	Date
<code>chore: create typesresponse schema</code>	2d204e4 · 6 hours ago	
<code>.github/workflows</code>	build(CI/CD): Added elixir_ci.yml 'Build and test' job for Gi...	
<code>assets</code>	feat(book_my_gigs): created backend phoenix project	
<code>config</code>	build(docker): created Dockerfile (#1) (#2)	
<code>lib</code>	chore: create typesresponse schema	
<code>priv</code>	chore: modified event changeset handling	
<code>test</code>	test(users): POST fails when payload is invalid	
<code>.dockerignore</code>	build(docker): created Dockerfile (#1)	
<code>.formatter.exs</code>	feat(book_my_gigs): created backend phoenix project	
<code>.gitignore</code>	feat(book_my_gigs): created backend phoenix project	
<code>Dockerfile</code>	build(docker): created Dockerfile (#1)	
<code>README.md</code>	feat(book_my_gigs): created backend phoenix project	
<code>docker-compose.yml</code>	build(docker): created Dockerfile (#1)	
<code>entrypoint.sh</code>	build(docker): created Dockerfile (#1)	
<code>mix.exs</code>	feat(guardian): Implement POST sign_out endpoint to rev...	
<code>mix.lock</code>	feat(guardian): Implement POST sign_out endpoint to rev...	

- Trello :

Pour organiser et suivre l'avancement de mon développement, j'ai utilisé Trello et sa méthodologie Kanban. J'ai structuré mon tableau en colonnes "*Backlog*", "*To Do*", "*Doing*", "*In Progress*" et "*Done*", en séparant clairement les tâches de backend, de frontend et de design.



Pour chaque carte, j'ai veillé à inclure une description détaillée des spécifications nécessaires à la réalisation de la tâche. De plus, j'ai systématiquement lié mes cartes Trello aux pull requests correspondantes dans Github, assurant ainsi une traçabilité complète entre la planification et l'exécution.



Cette organisation m'a permis de maintenir un rythme de développement efficace tout en gardant une vue claire sur l'avancement global du projet. En maintenant mon tableau Trello constamment à jour, j'ai créé une documentation vivante de l'avancement du projet, rendant ainsi possible l'intégration rapide de nouveaux collaborateurs. Cette approche structurée faciliterait grandement le travail en équipe, car chaque tâche est documentée et tracée depuis sa conception jusqu'à sa réalisation.

I.C : DESIGN & Conception

- **Excalidraw**

J'ai utilisé Excalidraw pour concevoir les maquettes de mon application et créer les diagrammes techniques. Son interface intuitive m'a permis de réaliser rapidement des wireframes de mes pages web ainsi que des schémas UML pour visualiser l'architecture de mon application. La simplicité de l'outil, combinée à son aspect épuré, m'a permis de me concentrer sur l'essentiel : la structure de mes pages et les relations entre les différentes entités de mon application. Ces maquettes ont servi de guide tout au long du développement, assurant une cohérence entre la vision initiale et l'implémentation finale.

- **DBDiagram**

J'ai utilisé DBDiagram pour concevoir et modéliser ma base de données PostgreSQL. Cet outil m'a permis de visualiser clairement les relations entre mes différentes tables grâce à sa syntaxe simple et son rendu graphique explicite. En définissant les clés primaires, les clés étrangères et les relations entre les tables, j'ai pu établir une structure de base de données solide avant même de commencer l'implémentation. Cette approche visuelle m'a aidé à anticiper les besoins en données de mon application et à optimiser les relations entre les entités dès la phase de conception.

III) Cahier des Charges

III.A : Besoins et Objectifs

- Contexte et Besoins Identifiés

Dans le contexte actuel de la scène musicale électronique, plusieurs problématiques ont été identifiées concernant l'accès à l'information sur les événements musicaux. Le principal besoin auquel répond Book My Gigs est la centralisation des informations événementielles, actuellement dispersées sur de multiples plateformes. Cette fragmentation de l'information crée plusieurs difficultés :

- La perte de temps dans la recherche d'évènements.
- Le risque de manquer des évènements pertinents.
- La difficulté de suivre l'actualité de plusieurs lieux ou artistes simultanément.

- Objectifs de l'Application

Book My Gigs s'est fixé plusieurs objectifs pour répondre à ces besoins :

- **Centralisation de l'Information**
 - Agréger les événements provenant de différentes sources en un point unique.
 - Standardiser la présentation des informations pour une meilleure lisibilité.
 - Maintenir une base de données à jour des événements à venir.
- **Amélioration de la Découverte**
 - Faciliter la découverte de nouveaux événements correspondant aux goûts des utilisateurs.
 - Faciliter la découverte de nouveaux événements correspondant aux goûts des utilisateurs.
 - Créer un système de recommandation basé sur les préférences utilisateur.

- **Création d'une Communauté**
 - Permettre aux utilisateurs de partager leurs découvertes.
 - Faciliter la communication des événements pour les organisateurs.
 - Créer un écosystème dynamique autour des événements musicaux.
- **Public Cible**

L'application s'adresse principalement à trois catégories d'utilisateurs :

 - **Les Audiophiles et Passionnés de Musique Electronique**
 - Personnes régulièrement à la recherche d'événements musicaux.
 - Amateurs souhaitant découvrir de nouveaux artistes et lieux.
 - Participants actifs de la scène électronique.
 - **Les Organisateurs d'Évènements**
 - Promoteurs de soirées et de concerts.
 - Gérants de clubs et de salles de concert.
 - Collectifs et associations musicales.
 - **Les Artistes**
 - DJs et producteurs cherchant à promouvoir leurs événements.
 - Collectifs artistiques organisant des performances.
 - Labels musicaux organisant des showcases.

Le profil type de l'utilisateur de Book My Gigs est un amateur de musique électronique âgé de 18 ans ou plus, vivant en zone urbaine, audiophile et culturellement actif. Cette personne fréquente régulièrement les événements musicaux (2 à 4 fois par mois) et utilise activement les réseaux sociaux pour se tenir informée des actualités musicales.

- Perspective d'Évolution

Cette première version de Book My Gigs pose les fondations d'une plateforme appelée à évoluer et n'est à ce stade qu'une **MVP (Produit minimum viable)**. Les futurs développements envisagés incluent :

- **Interactions entre utilisateurs**
 - Système de messagerie instantanée pour échanger sur les évènements.
 - Possibilité de suivre d'autres utilisateurs et leur agenda d'évènements.
 - Création de groupes d'intérêts par genre musical et localisation.
- **Intégration des APIs externes**
 - Connection directe avec les APIs de plateformes comme *Resident Advisor* et *Shotgun*.
 - Synchronisation automatique des évènements *Facebook*.
 - Agrégation en temps réel des nouvelles publications d'évènements.
- **Dimension artiste**
 - Création de profils dédiés aux artistes.
 - Calendrier personnel des performances par artiste.
 - Système de notification pour les nouveaux évènements d'un artiste suivi.

L'objectif à long terme est de devenir la référence en matière de découverte et de suivi d'événements de musique électronique, tout en maintenant une expérience utilisateur simple et efficace.

III.B : MVP & Fonctionnalités clés

Dans une démarche de développement agile, j'ai choisi de commencer par une MVP (Minimum Viable Product) de Book My Gigs. Cette approche m'a permis de me concentrer sur les fonctionnalités essentielles, de valider rapidement les concepts clés de l'application et d'établir une base solide pour les futures évolutions. En me focalisant sur les besoins fondamentaux des utilisateurs, j'ai pu développer une première version fonctionnelle et cohérente de la plateforme.

- Fonctionnalités Actuelles de Book My Gigs

L'application dans sa version MVP propose déjà plusieurs fonctionnalités clés :

- **Système d'Authentification et Profils**
 - Création de compte utilisateur et authentification sécurisée.
 - Profils personnalisés avec des informations de base (nom, prénom, date de naissance).
 - Sélection des genres musicaux préférés et localisation de l'utilisateur.
- **Gestion des évènements**
 - Création d'évènements avec informations détaillées :
 - Titre et description.
 - Date et heure.
 - Lieu (nom de la salle et adresse complète).
 - Type d'évènement (Club, Concert, Festival).
 - Genres musicaux associés.
 - URL externe pour plus d'informations.
- **Navigation et Découverte**
 - Vue d'ensemble de tous évènements à venir.
 - Section « *My Events* » pour gérer ses propres publications.
 - Système de filtrage par :
 - Localisation.
 - Genre musical.

- Type d'évènement.
- Interface Utilisateur
 - Design responsive et moderne en utilisant Tailwind CSS.
 - Navigation intuitive entre les différentes sections.
 - Affichage clair des évènements sous forme de cards.
 - Formulaires ergonomiques pour la création et modification de contenu.

Cette base solide permet déjà aux utilisateurs de partager et découvrir des évènements, tout en posant les fondations nécessaires pour les développements futurs de l'application.

III.C : Charte Graphique

- Les couleurs :

Book My Gigs se veut être une application efficace et intuitive, où l'accès à l'information prime. Cette vision se reflète dans une identité graphique épurée, construite autour de trois couleurs principales.

#4F46E5 (Tailwind CSS : indigo-600)

Le bleu indigo occupe une place centrale dans notre palette. Cette couleur évoque l'énergie et le dynamisme de la musique électronique, rappelant les néons des clubs et l'électricité qui anime cette scène musicale. Elle insuffle vie à l'interface tout en maintenant une élégance professionnelle.

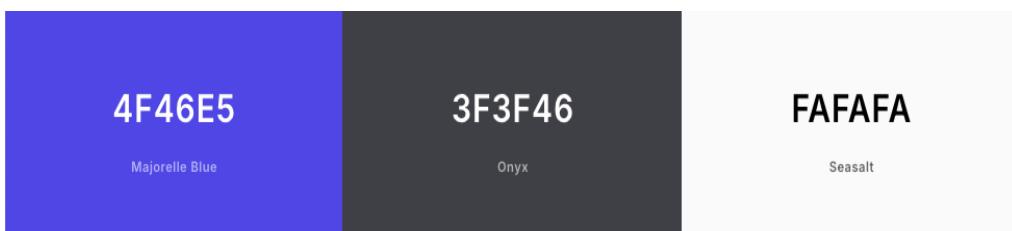
#3F3F46 (Tailwind CSS : Zinc 700)

Le gris foncé, utilisé pour nos éléments textuels, apporte une lisibilité optimale à notre interface. Sa teinte profonde offre un contraste parfait avec nos fonds clairs, permettant une lecture confortable des informations. La constance dans son utilisation maintient une cohérence visuelle tout en assurant une hiérarchie claire de l'information.

#F9FAFB Tailwind CSS : Gray-50)

Le gris clair, omniprésent dans notre design, incarne la clarté et la simplicité d'utilisation. Il crée des espaces de respiration qui permettent une lecture intuitive des informations, tout en apportant une subtile profondeur à notre design. Cette teinte neutre et sophistiquée contribue à l'expérience utilisateur fluide que nous recherchons.

L'association de ces trois couleurs crée une harmonie visuelle qui sert directement notre objectif principal : permettre aux utilisateurs d'accéder efficacement aux informations sur les événements musicaux. Cette approche minimaliste reflète notre engagement envers la simplicité, tout en maintenant une esthétique moderne et professionnelle.



- Le logo & la police

Notre logo reflète parfaitement l'identité épurée de Book My Gigs. En adoptant un fond **#4F46E5**, il capture l'énergie de la scène électronique tout en restant élégant. Le texte en **#F9FAFB** assure une lisibilité optimale tout en maintenant la sobriété recherchée.

La police **Roboto**, avec ses lignes géométriques et modernes, renforce notre engagement envers la simplicité et l'efficacité.

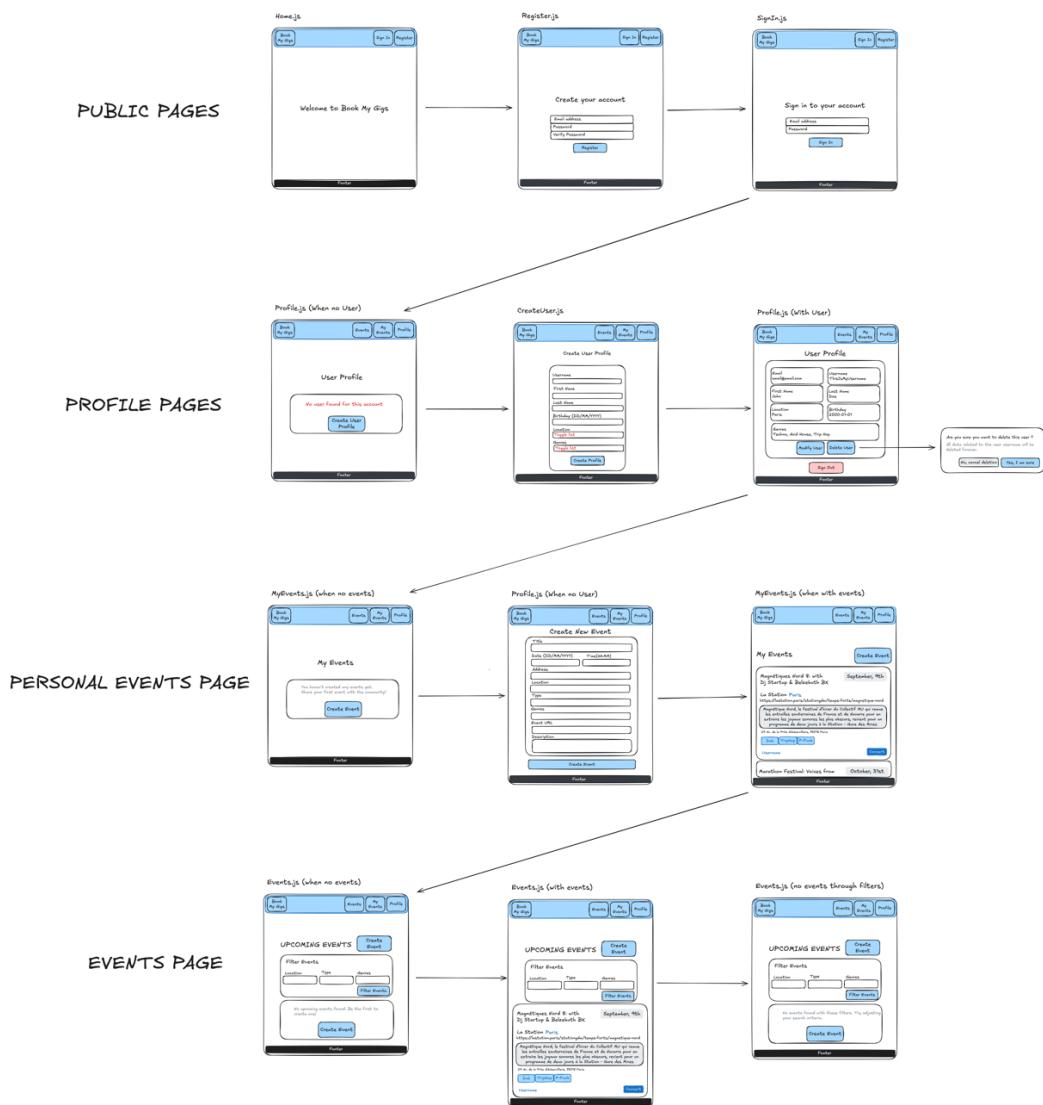
Cette combinaison minimalistique s'aligne parfaitement avec notre charte graphique globale, où chaque élément est choisi pour sa fonctionnalité avant tout. Le logo incarne ainsi notre philosophie : une approche directe et efficace de l'information, sans fioritures inutiles.



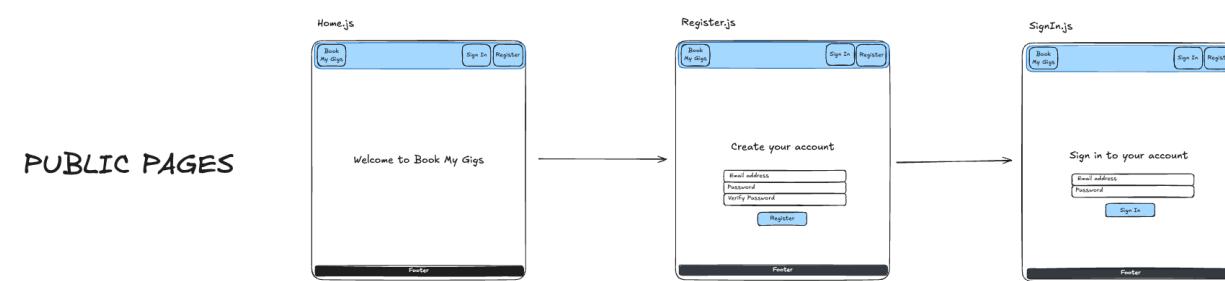
III.D : Maquettes

J'ai choisi d'utiliser Excalidraw pour concevoir les maquettes de mon projet car c'est un outil simple et intuitif qui m'a permis de rapidement visualiser et itérer sur les différentes fonctionnalités et interactions de mon application. Excalidraw m'a offert une grande flexibilité pour dessiner rapidement des wireframes et diagrammes, tout en conservant un aspect visuel minimal. Cette approche m'a semblé plus adaptée, car elle m'a aidé à me concentrer sur la structure et la conception globale de mon projet.

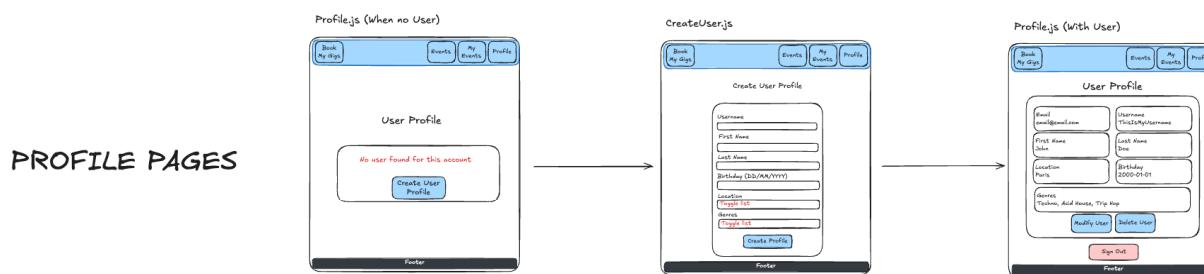
Ci-dessous, le schéma entier de la plateforme :



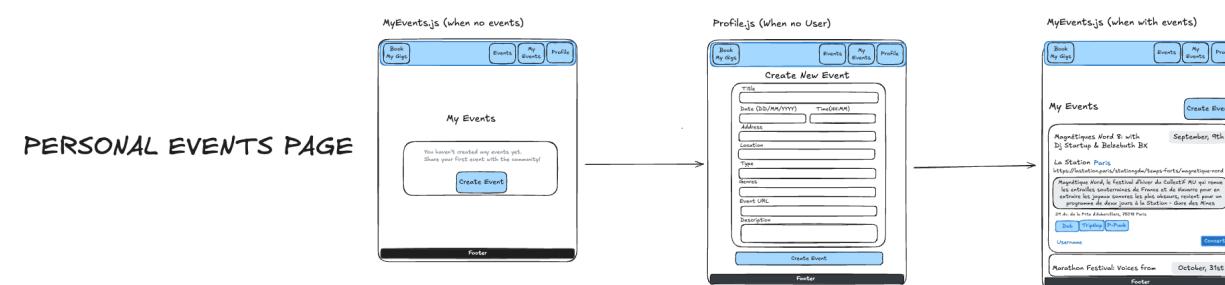
Page d'accueil, d'inscription et de connexion :



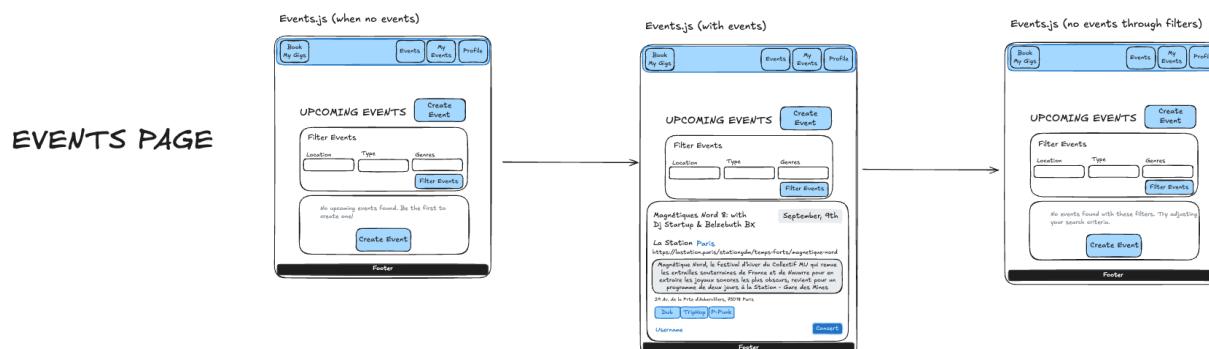
Page de profile et de gestion du compte :



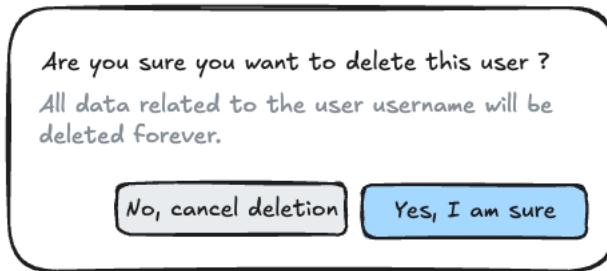
Pages des évènements de l'utilisateur et création d'évènements :



Page d'évènements sur la plateforme et de recherche par filtre :



Modale de confirmation d'action :



Reproduction de ces maquettes dans l'application avec pour exemple, la page MyEvents.js lorsqu'un utilisateur a déjà posté des évènements :

MyEvents.js (when with events)

BOOK MY GIGS

Events My Events Profile

Create Event

My Events

Magnétiques Nord 8: with Dj Startup & Belzebuth BX

September, 9th

La Station Paris

<https://lastation.paris/stationgdm/rendez-vous/2024-11-14-magnetique-nord-1>

Magnétique Nord, le festival d'hiver du Collectif MU qui remue les entrailles souterraines de France et de Navarre pour en extraire les joyaux sonores les plus obscurs, revient pour un programme de deux jours à la Station - Gare des Mines

24 Av. de la Porte d'Aubervilliers, 75018 Paris

Dub Trip Hop Post-Punk

Username Concert

Marathon Festival: Voices from October, 31st

Footer

Dec 12th, 2024, 9:00pm

Magnétiques Nord 8: Lydia Lunch

Lydia Lunch, nom d'artiste de Lydia Anne Koch (de son vrai nom complet), née le 2 juin 1959 à Rochester (États-Unis), est une chanteuse, poétesse, écrivaine et actrice américaine. Sa carrière musicale débute au sein du groupe Teenage Jesus and the Jerks sur l'album anthologique de Brian Eno paru en 1977 : *No New York*, considéré comme l'acte fondateur du mouvement no wave.

Address: 12 Rue des Aureoles, 75020 Paris, France

New Wave Post-punk

Gab75020 Concert

© 2024 BookMyGigs. All rights reserved.

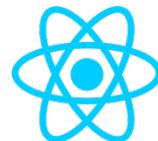
About Privacy Policy Terms of Service

IV) Spécifications Techniques

Book My Gigs est une plateforme développée avec les frameworks React.js et Tailwind CSS côté front, Phoenix couplé à Ecto et PostgreSQL côté back.

IV.A : Frontend

- IV.A.1 : React.js : un choix stratégique :



React.js a été choisi comme framework frontend pour Book My Gigs pour sa flexibilité et sa puissance dans la création d'interfaces utilisateur dynamiques. Cette bibliothèque JavaScript, maintenue par Meta, excelle dans le développement d'applications à page unique (SPA), particulièrement adaptées à notre cas d'usage.

- Pourquoi React.js :

React.js se distingue par sa gestion efficace du DOM virtuel et son système de composants réutilisables, ce qui est parfaitement aligné avec les besoins de Book My Gigs. Dans mon application, cette approche est particulièrement visible dans la structuration de mes composants, comme le montre mon architecture :

- **Composants de layout** : [MainLayout.js](#), [PublicLayout.js](#).
- **Composants réutilisables** : [EventCard.js](#), [AuthHeader.js](#).
- **Pages fonctionnelles** : [CreateEvent.js](#), [Events.js](#), [Profile.js](#).

L'utilisation des **hooks React** m'a permis une gestion d'état claire et efficace, particulièrement importante dans une application de gestion d'évènements où les données sont constamment mises à jour.

- Structure du frontend & Organisation des dossiers

La structure de mon projet React suit une organisation modulaire et intuitive. Au niveau racine, nous trouvons les fichiers de configuration essentiels :

- **package.json** et **package-lock.json** pour la gestion des dépendances.
- Configuration pour **ESLint** et **Prettier** assurant la cohérence du code.
- **Tailwind.config.js**
le fichier de configuration Tailwind qui définit le design system de l'application.

Organisations des dossiers :

Dans le dossier **/src**, mon application est organisée en plusieurs fichiers et dossiers principaux :

- **App.js**
Constitue le point d'entrée principal de mon application React. Il définit la structure de routage fondamentale en utilisant **react-router-dom**.

```
function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/register" element={<Register />} />
        <Route path="/signin" element={<SignIn />} />
        <Route path="/profile/:id" element={<Profile />} />
        <Route path="/create-user" element={<CreateUser />} />
        <Route path="/modify-user/:id" element={<ModifyUser />} />
        <Route path="/my-events" element={<MyEvents />} />
        <Route path="/create-event" element={<CreateEvent />} />
        <Route path="/events" element={<Events />} />
      </Routes>
    </Router>
  );
}
```

Cette structure de routing définit :

- Les routes publiques (« /register », « /signin »).
- Gère les routes authentifiées (« /profile/ :id », « /my-events », etc.).
- Utilise des routes paramétrées pour les pages dynamiques (« /profile/:id », « /modify-user/:id »).

- **Pages/**

Contient les composants principaux de chaque route de l'application :

- **Home.js**
Page d'accueil publique.
- **Event.js**
Liste des évènements.
- **CreateEvent.js**
Formulaire de création d'évènements.
- **Profile.js**
Gestion du profil utilisateur.
- **MyEvents.js**
Vue personnalisée des évènements de l'utilisateur.

- **Components/**

Regroupe les composants réutilisables :

- **EventCard.js**
Affichage standardisé d'un évènement.
- **AuthHeader.js** et **Header.js**
Navigation conditionnelle selon l'authentification.
- **Footer.js**
Pied de page unifié.

- **Layout/**

Définit les structures de page réutilisables :

- **MainLayout.js**
Layout pour les utilisateurs authentifiés.
- **PublicLayout.js**
Layout pour pages publiques.

- **Assets/**

Pour ranger nos assets comme par exemple le logo de notre application.

- **Arborescence de book_my_gigs_front :**

```
notes.txt
package-lock.json
package.json
postcss.config.js
public
  favicon.ico
  index.html
  logo192.png
  logo512.png
  manifest.json
  robots.txt
src
  App.css
  App.js
  App.test.js
  Assets
    logo-color.png
  Components
    AuthHeader.js
    EventCard.js
    Footer.js
    Header.js
  Layout
    MainLayout.js
    PublicLayout.js
  Pages
    CreateEvent.js
    CreateUser.js
    Events.js
    Home.js
    ModifyUser.js
    MyEvents.js
    Profile.js
    Register.js
    SignIn.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  tailwind.config.js
```

- Gestion des formulaires

La gestion des formulaires dans mon application est construite autour des composants React contrôlés, utilisant les hooks d'état pour une validation en temps réel. Prenons l'exemple du formulaire de création d'évènement ([CreateEvent.js](#)) :

```
const [title, setTitle] = useState('');
const [description, setDescription] = useState('');
const [date, setDate] = useState('');
const [time, setTime] = useState('');
```

- Validation des formulaires

J'ai implémenté un système de validation robuste :

- o Validation des dates avec **Regex**.
- o Validation des champs requis.
- o Gestion des erreurs en temps réel.

```
const dateRegex = /^(0[1-9]|1[2-9]|3[01])\/(0[1-9]|1[012])\/(19|20)\d\d$/;
if (!dateRegex.test(date)) {
  setError('Please enter date in DD/MM/YYYY format');
  return;
}
```

- Gestion des états complexes

Pour les fonctionnalités plus complexes comme la sélection de genres musicaux ou de localisation, nous utilisons des états composés :

```
const [locations, setLocations] = useState([]);
const [selectedLocation, setSelectedLocation] = useState('');
const [filteredLocations, setFilteredLocations] = useState([]);
```

- IV.A.2 : Tailwind CSS : Efficacité et Cohérence



- Organisation des styles

Tailwind CSS est utilisé de manière systématique à travers l'application, suivant une approche utility first.

Prenons l'exemple d'une div dans `CreateEvent.js` qui vient définir le conteneur principal du formulaire de création d'évènement.

```
<div className="flex flex-col justify-center w-full max-w-2xl px-4 sm:px-6 lg:px-8">
```

Cette `div` enveloppe le titre « *Create Event* » et le formulaire complet avec tous ses champs (titre, date, heure, etc.)

Elle se caractérise par :

- **Une structure de base**
 - `flex` : Définit un conteneur flexible.
J'utilise intensivement le système de grille de **FlexBox** de Tailwind.
 - `flex-col` : Les éléments enfants seront empilés verticalement.
 - `justify-center` : Centre les éléments verticalement dans le conteneur.
 - `w-full` : Prend 100% de la largeur disponible.
- **Contrainte de largeur maximale**
 - `max-w-2xl` : limite la largeur maximale à 42rem(672px).
- **Padding responsive**
 - `px-4` : Padding horizontal de base de 1rem (16px) sur mobile.
 - `sm:px-6` : Augmente à 1,5rem (24px) sur les écrans de plus de 640px (breakpoint '`sm`').

- **lg:px-8** : Augmente à 2rem (32px) sur les écrans de plus de 1024px (breakpoint ‘**lg**’).

Cette combinaison crée un conteneur centré qui :

- S'adapte à la largeur de l'écran tout en ayant une limite maximale.
- Maintient un espacement horizontal qui augmente progressivement avec la taille de l'écran.
- Garde son contenu organisé verticalement et centré.

Plus globalement, cette structure m'a permis de maintenir une base de code propre et maintenable, tout en offrant une expérience utilisateur cohérente et performante. L'utilisation combinée de React.js pour la logique et de Tailwind CSS pour le style m'a permis de développer rapidement tout en maintenant une qualité de code élevée.

– IV.A.3 : Sécurité

J'ai mis en place plusieurs mesures de sécurité dans mon application React pour protéger les données des utilisateurs et sécuriser les accès :

- Gestion des Tokens

J'utilise le **localStorage** pour stocker le token JWT et l'ID du compte après l'authentification :

SignIn.js :

```
const handleSubmit = async (e) => {
  e.preventDefault();
  setError('');

  try {
    const response = await axios.post('http://localhost:4000/accounts/sign_in', {
      account: {
        email,
        password
      },
      headers: {
        'Content-Type': 'application/json'
      }
    });
  } catch (err) {
    setError(err.response?.data?.message || 'An error occurred');
  }
};
```

Gabrielparizet, 4 months ago • feat(signin): Built sign in page and form

```
// Store token and account info in localStorage
localStorage.setItem('token', response.data.token);
localStorage.setItem('accountId', response.data.account.id);

navigate(`/profile/${response.data.account.id}`)
}
```

- Protection des routes

J'ai implémenté un système de redirection pour les routes protégées. Si un utilisateur non authentifié tente d'accéder à une page sécurisée, il est automatiquement redirigé vers la page de connexion :

Events.js :

```
if (!localStorage.getItem('token')) {
  return <Navigate to="/signin" />;
}
```

- Headers d'Authentification

Pour chaque requête d'API nécessitant une authentification, j'inclus systématiquement le token JWT dans les headers :

MyEvents.js :

```
const userResponse = await axios.get(`http://localhost:4000/api/accounts/${accountId}/users`, {
  headers: {
    'Authorization': `Bearer ${token}`
  }
});
```

- Gestion de la déconnexion

J'ai implémenté une fonction de déconnexion sécurisée qui :

- Révoque le token côté serveur.
- Nettoie le **localStorage**.
- Redirige l'utilisateur vers la page de connexion.

Profile.js :

```
const handleSignOut = async () => {
  const token = localStorage.getItem('token');
  try {
    const response = await axios.post(`http://localhost:4000/accounts/sign_out`, {}, {
      headers: {
        'Authorization': `Bearer ${token}`
      }
    });

    if (response.data.message === "Successfully signed out") {
      localStorage.removeItem('token');
      localStorage.removeItem('accountId');
      navigate('/signin');
    } else {
      setSignOutMessage('Something went wrong');
    }
  } catch (err) {
    setSignOutMessage('Failed to sign out. Please try again.');
  }
};
```

- Gestion des erreurs

J'ai implémenté une gestion globale des erreurs avec des messages utilisateur appropriés, tout en évitant d'exposer des informations sensibles :

Profile.js :

```
    } catch (err) {  
        setError('No user found for this account');
```

Ces mesures de sécurité forment un ensemble cohérent qui protège les données des utilisateurs et assure une expérience sécurisée sur l'application. Cette approche s'aligne de plus avec plusieurs réglementations importantes:

- **RGPD (Règlement Général sur la Protection des Données)**
 - o La gestion sécurisée des tokens et des données utilisateur respecte le principe de confidentialité.
 - o Le système de déconnexion et la possibilité de supprimer un utilisateur permettent aux utilisateurs de contrôler l'accès à leurs données.
 - o Les messages d'erreur génériques évitent l'exposition d'informations sensibles.
 - o La collecte de données est limitée aux informations strictement nécessaires au fonctionnement de l'application.
- **RGS (Référentiel Général de Sécurité)**
 - o L'utilisation de JWT pour l'authentification suit les bonnes pratiques de sécurité.
 - o La validation des données en entrée protège contre les injections et autres vulnérabilités.
 - o La gestion des sessions via tokens respecte les principes de sécurité des échanges.

IV.A.4 : Accessibilité

Notre choix de design et de charte graphique s'aligne également avec plusieurs critères du RGAA (Référentiel Général d'Amélioration de l'Accessibilité) :

- **Contraste et Lisibilité**
 - o L'utilisation du **Zinc-700 (#3F3F46)** pour le texte sur fond clair (**Gray-50 #F9FAFB**) assure un ratio de contraste optimal, dépassant le minimum requis de 4.5:1.
 - o La hiérarchie visuelle claire de nos textes améliore la compréhension du contenu.
 - o Les tailles de police adaptatives garantissent une lecture confortable sur tous les supports
- **Navigation et Structure**
 - o L'interface épurée avec des espacements généreux facilite l'identification des éléments cliquables.
 - o La cohérence visuelle maintenue à travers l'application aide à la compréhension de la navigation.
 - o Les formulaires sont clairement structurés avec des labels explicites et des messages d'erreur visibles.
- **Retours Visuels**
 - o L'utilisation de **l'Indigo-600 (#4F46E5)** pour les éléments interactifs les rend facilement identifiables.
 - o Les états des boutons (hover, focus, active) sont clairement différenciés.
 - o Les messages de feedback utilisent des couleurs distinctives tout en maintenant la lisibilité.
- **Adaptabilité**
 - o Le design responsive s'adapte aux différentes tailles d'écran sans perte d'information.
 - o La mise en page fluide permet le zoom jusqu'à 200% sans perte de fonctionnalité.

Cette approche du design, bien qu'elle ne couvre pas l'intégralité des critères RGAA, pose des bases solides pour une accessibilité numérique de qualité.

IV.B : Backend

IV.B.1 : Elixir et Phoenix

J'ai choisi d'utiliser Elixir et Phoenix pour plusieurs raisons fondamentales qui font de cette stack technologique un excellent choix pour mon application de gestion d'événements musicaux.

- **Elixir : Un langage conçu pour la robustesse et l'évolutivité**

Elixir a été créé par José Valim en 2011, alors qu'il cherchait une solution pour résoudre les problèmes de concurrence et de scalabilité qu'il rencontrait avec Ruby on Rails. Plutôt que de partir de zéro, il a choisi de construire Elixir sur l'**Erlang VM (BEAM)**, qui avait déjà prouvé sa fiabilité depuis les années 1980 chez Ericsson pour leurs systèmes de télécommunication.

L'objectif était de combiner les meilleures caractéristiques d'Erlang (tolérance aux pannes, distribution, concurrence) avec une syntaxe moderne inspirée de Ruby et un système de métaprogrammation puissant.

- **Architecture et compilation**

Le processus de compilation d'Elixir est particulièrement intéressant :