

Portal_Job

PRÉSENTATION PROJET

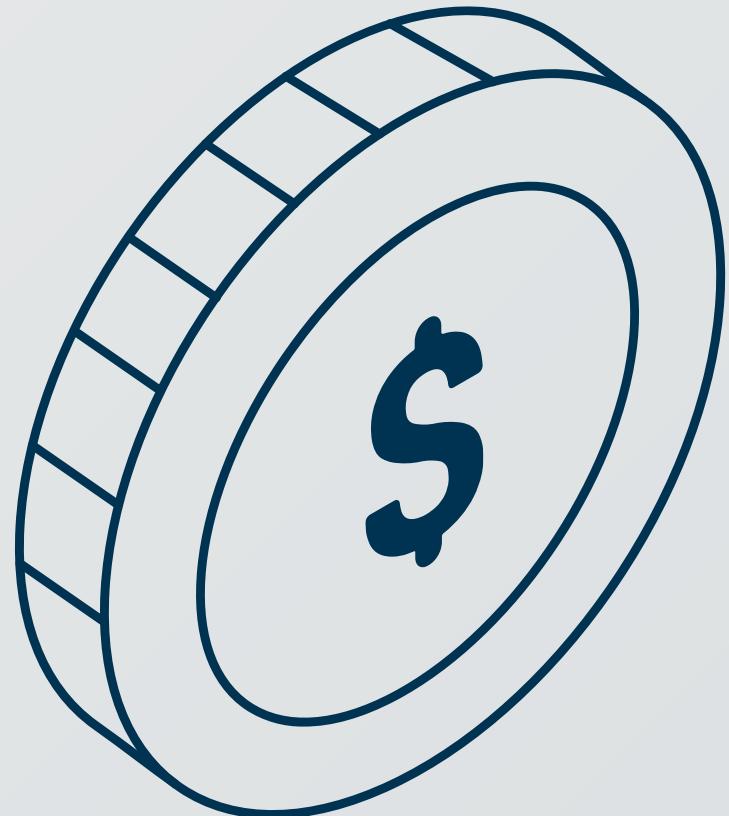
PORTAL_JOB

Bonjour,

Je suis Ahmed, après une carrière dans le domaine de l'alimentation et une transition dans le service client, j'entreprends une reconversion professionnelle pour devenir développeur web.

Mon dernier parcours m'a ancré dans le domaine de l'informatique, ce qui m'a naturellement conduit à explorer le développement web.

Aujourd'hui, je vous présente mon projet qui n'est pas encore fini, ce projet m'est à cœur car il s'agit pour moi du trophé de ma persévérance et mon envie.



Votre avenir professionnel commence ici !



Bienvenue dans le projet de refonte du site
Portal_Job

Cette initiative ambitieuse vise à moderniser

à améliorer l'expérience de l'utilisateur en ligne

Mon objectif sera donc de créer un site web à la hauteur des exigences demandées pour un site d'offres d'emplois

Besoin du client

Mon projet, **Portal_Job**, répond à un double besoin fondamental sur le marché de l'emploi en ligne :

Pour les Candidats (Utilisateurs finaux)

Accès ciblé

Système de recherche avancée (par titre, lieu, type de contrat) et de filtrage précis.

Visibilité de l'offre

Fiches d'offres détaillées et claires, offrant toutes les informations essentielles.

Suivi des opportunités

Possibilité de sauvegarder les offres ("Favoris") et de suivre le statut des candidatures soumises par mail (PhpMailer).

Candidature facilitée

Processus de candidature simple (téléchargement de CV/lettre de motivation) et gestion d'un profil utilisateur.

Pour les Entreprises (recruteurs)

Diffusion rapide

Interface d'administration dédiée (Espace Recruteur) pour publier et modifier des offres instantanément.

Gestion des candidatures

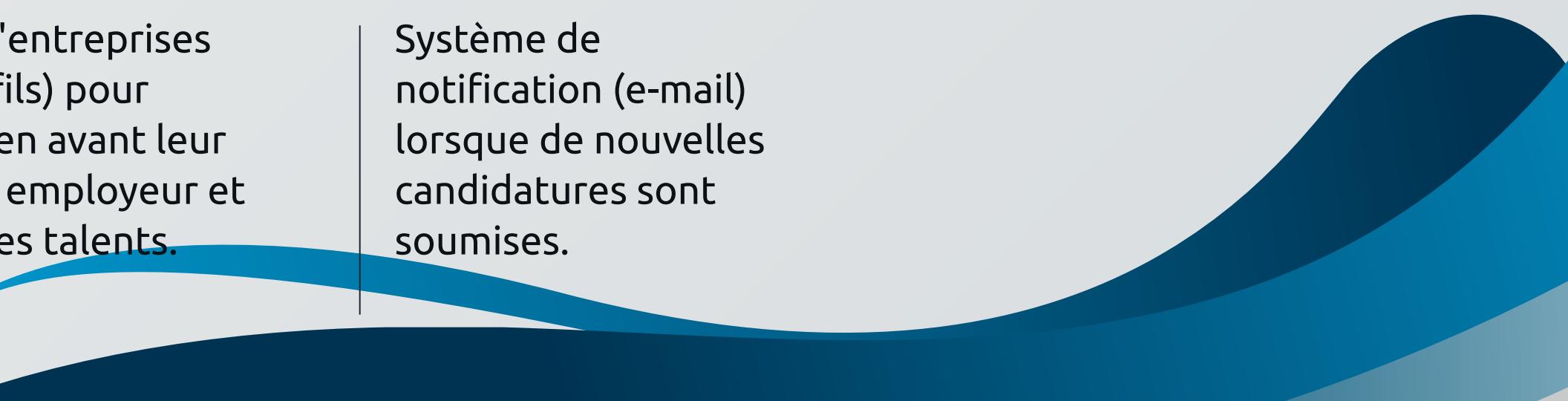
Tableau de bord (Dashboard) permettant de visualiser, trier et gérer toutes leurs offres publiées.

Image de marque

Pages d'entreprises (ou profils) pour mettre en avant leur marque employeur et attirer les talents.

Efficacité

Système de notification (e-mail) lorsque de nouvelles candidatures sont soumises.



Spécifications Techniques Clés du Projet Portal Job

Cette section met en lumière les choix architecturaux et technologiques fondamentaux pour la réalisation de la plateforme :

Architecture Moderne et Robuste (Back-end)

Back-end (Logique Métier) :
Laravel (PHP) et MySQL

**Implémentation stricte du modèle
MVC/POO pour structurer le code.**

**Laravel est utilisé pour créer une API REST
sécurisée.**

**L'API REST est le point de connexion unique
entre le Front et le Back.**

Expérience Utilisateur (Front-end)

Front-end (Présentation) :
Vue.js (avec HTML/CSS/JavaScript).

**Développement d'une Application à Page
Unique (SPA).**

**Cela assure une fluidité et une réactivité
supérieures, améliorant l'expérience
Candidat et Recruteur.**



Gestion de Projet : Le Cycle Formation-Projet (10 Mois)

Le planning a été adapté au rythme de l'alternance, transformant chaque module de formation en un levier d'amélioration pour le projet.

Rythme d'Alternance et Acquisition

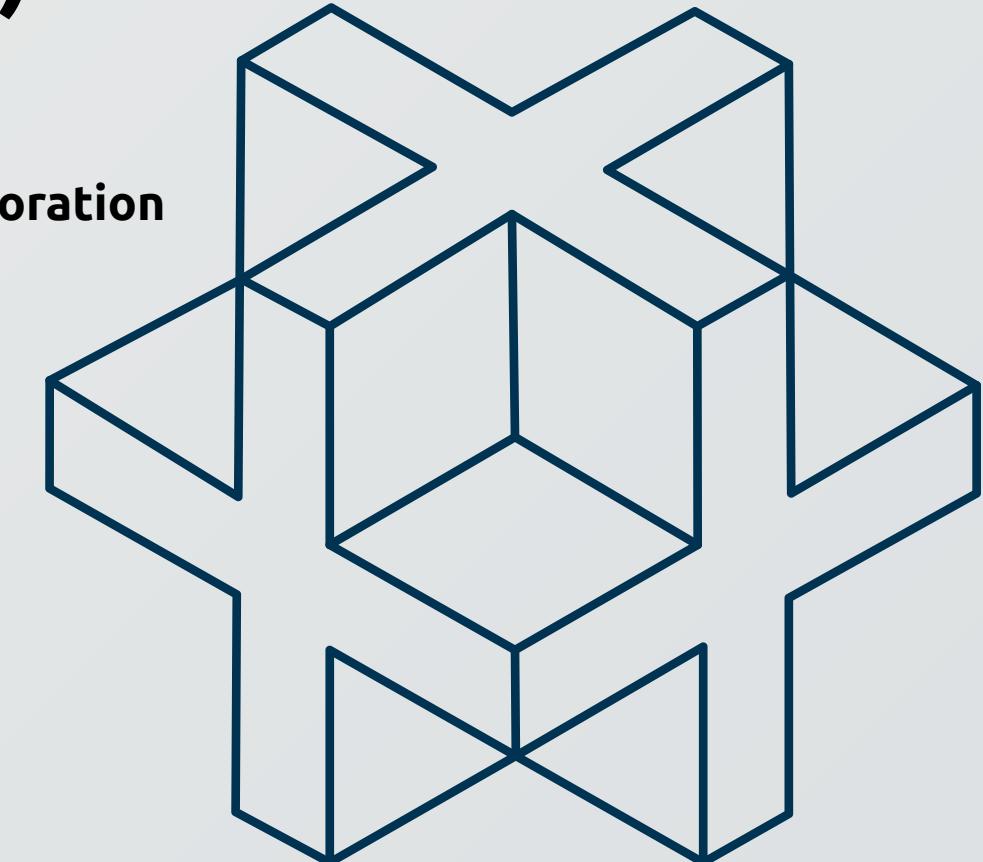
Durée Globale : Le projet s'est étalé sur environ 10 mois (de la conception à la livraison)

Méthode : Un développement itératif, où l'avancement était directement lié à l'acquisition de nouvelles compétences en formation.

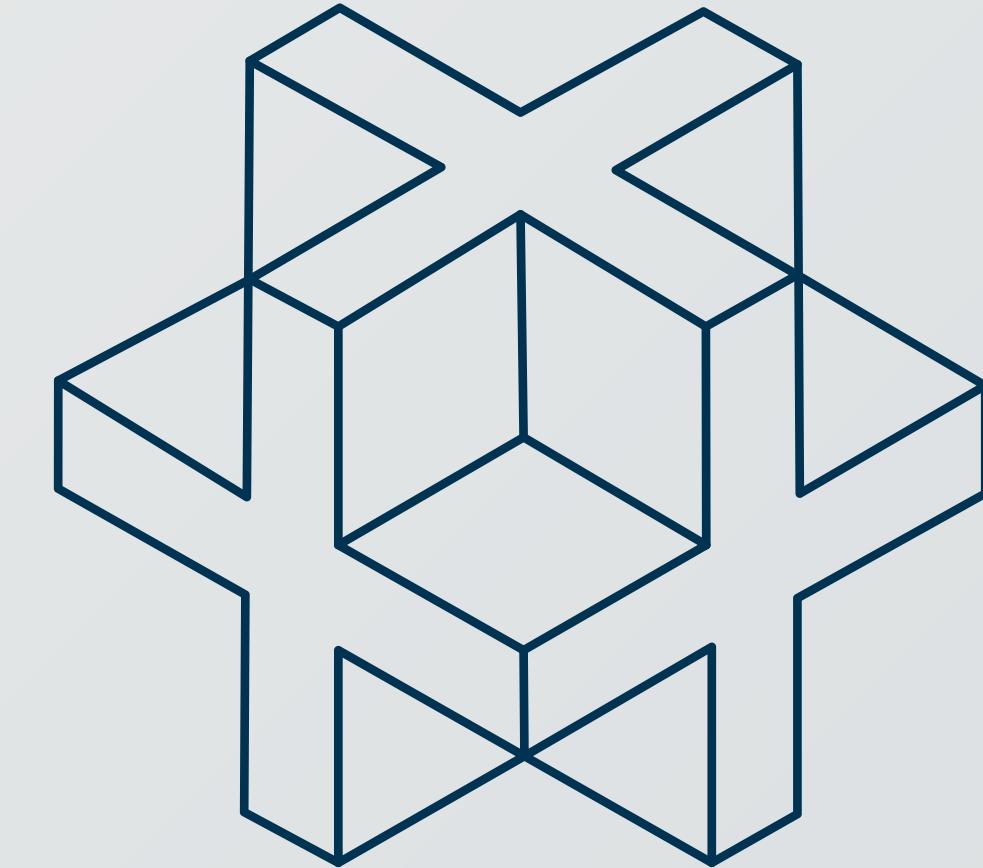
Valeur Ajoutée de l'Alternance

Synergie : Le projet a servi de laboratoire d'application pratique immédiate des concepts théoriques vus en cours (POO, MVC, Frameworks)

Montée en Compétence : Cela a permis une montée en compétence graduelle et l'intégration de technologies avancées (Laravel et Vue.js) qui n'auraient pas été possibles dans un délai court.



Phases Clés du Développement



<u>Phase de Projet</u>	<u>Durée Estimée (vs 10 mois)</u>	<u>Compétences Appliquées</u>
Analyse et Maquettage	1 mois (Initial)	Recueil des besoins, spécifications fonctionnelles, conception UI/UX.
Développement Back-end	4-5 mois (Parallèle à la formation)	Implémentation Laravel (MVC/POO), Modèles Eloquent, Construction de l'API REST et Sécurité.
Postman API	2-3 mois (Suivant l'API)	Intégration Vue.js (Composants dynamiques), gestion des appels à l'API, ergonomie.
Tests	1 mois (Final)	Tests d'API (Postman), revue du code, amélioration de l'UX, corrections finales.

SOMMAIRE



I - VUE D'ENSEMBLE DU PROJET

Contexte du projet
Objectifs
Fonctionnalités

III - PARTIE BACKEND

Création bdd
CRUD
Postman
Composants Backend
PhpMailer

II - PARTIE FRONTEND

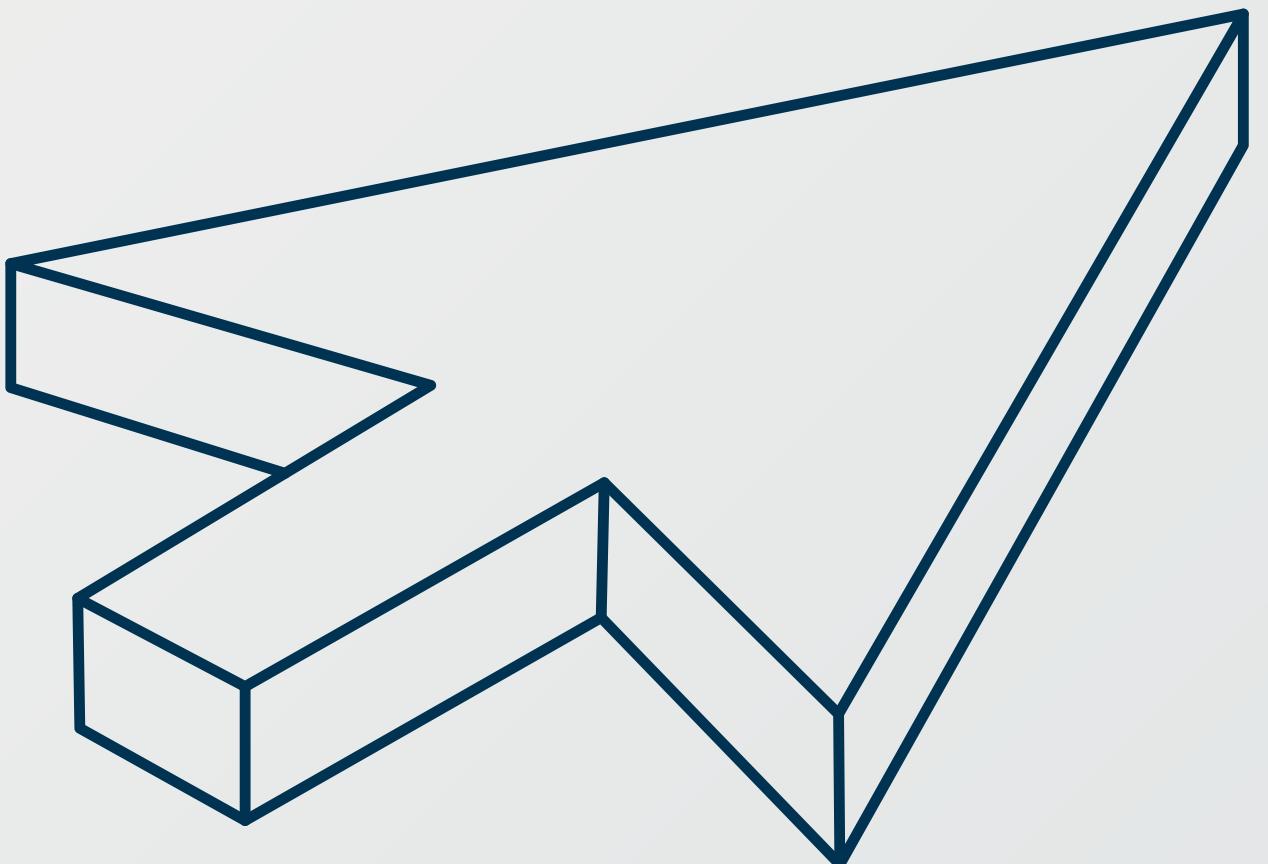
Maquettage du projet
Intégration html/css
Structure MVC

IV - CONCLUSION

Bilan du projet
Perspectives d'évolutions
Conclusion

V - ANNEXES

I -



VUE D'ENSEMBLE DU PROJET

Contexte du projet

Pour l'élaboration de mon projet, j'ai dans un premier temps opéré à une veille technologique et une analyse des tendances actuelles du web, je me suis donc dirigé vers des sites connus comme 'indeed' 'hellowork'

Cette étape m'a permis de définir les fonctionnalités nécessaires, l'aspect visuel, et ainsi établir une feuille de route solide pour le développement du site.

L'objectif de cette refonte est de :

Améliorer l'expérience utilisateur

Rendre la navigation plus intuitive et fluide

Adopter un design moderne et réactif afin que l'oeil du client soit attiré



Les objectifs

**Cette refonte a pour but de transformer Portal_Job en une plateforme
d'emploi
plus dynamique et intuitive, répondant mieux aux besoins des candidats
et des entreprises.**

**Mon engagement est de simplifier radicalement la recherche et le
processus
de recrutement en utilisant des technologies modernes**

En clair,

Améliorer l'accessibilité :

Je m'engage à rendre l'information accessible à tous.

Enrichir l'expérience utilisateur :

Une expérience en ligne immersive à travers une interface conviviale et intuitive.



Promouvoir la facilité de trouver des offres

Devenir une destination incontournable pour les candidats comme les entreprises

Les fonctionnalités

Dans le cadre de la refonte numérique de **Portal_Job**, j'ai identifié plusieurs fonctionnalités essentielles pour enrichir l'expérience utilisateur et répondre aux besoins diversifiés des utilisateurs. Ces fonctionnalités visent à créer une plateforme interactive et moderne.

Gestion des
Offres d'Emploi
(CRUD)

Recherche et
Filtres Avancés

Candidature en
Ligne Simplifiée

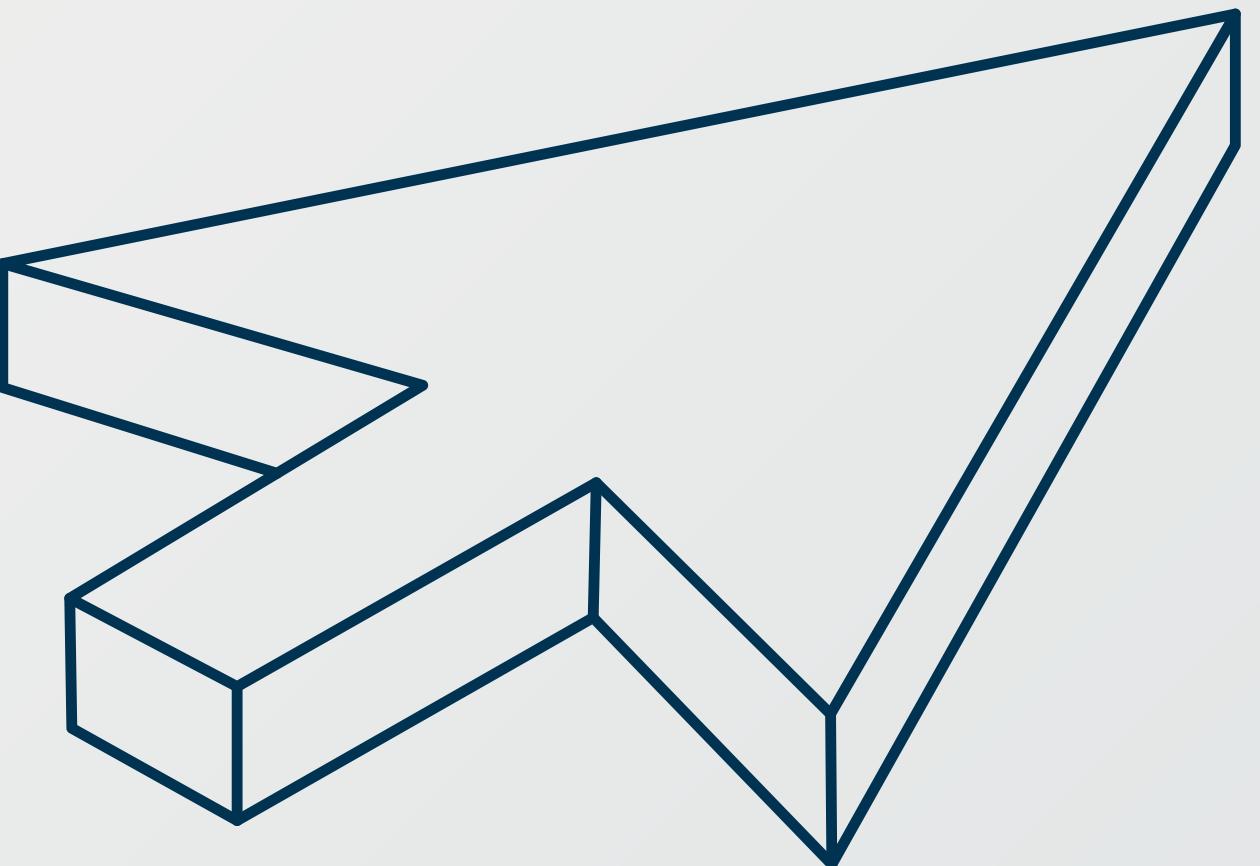
Espace
Utilisateur
Société

Dashboard
Admin

Système de
Favoris

Contact

II -

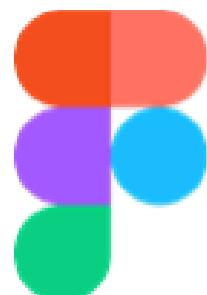
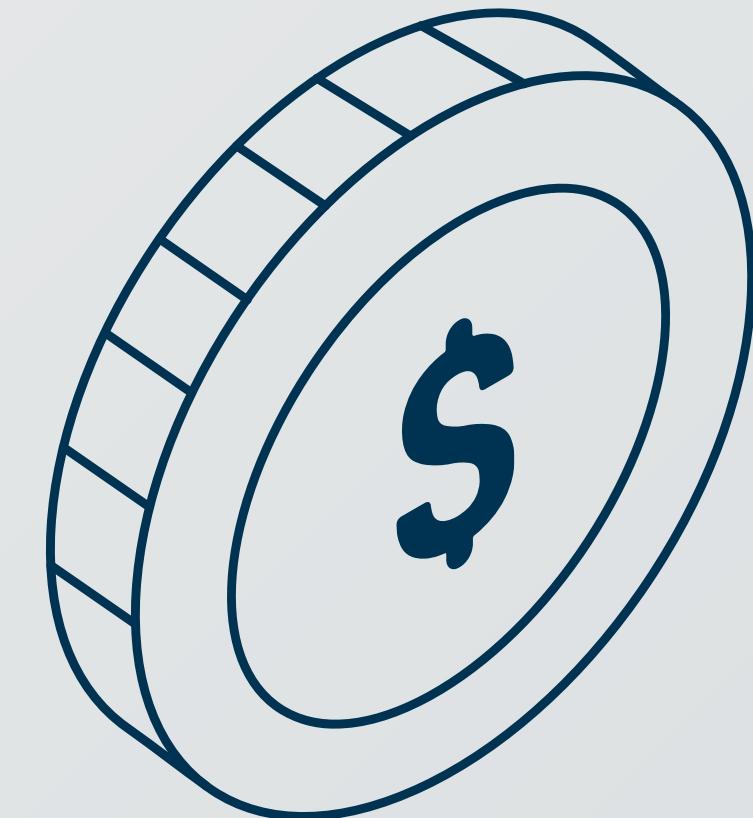


PARTIE FRONTEND

Maquette FIGMA

Avant-propos

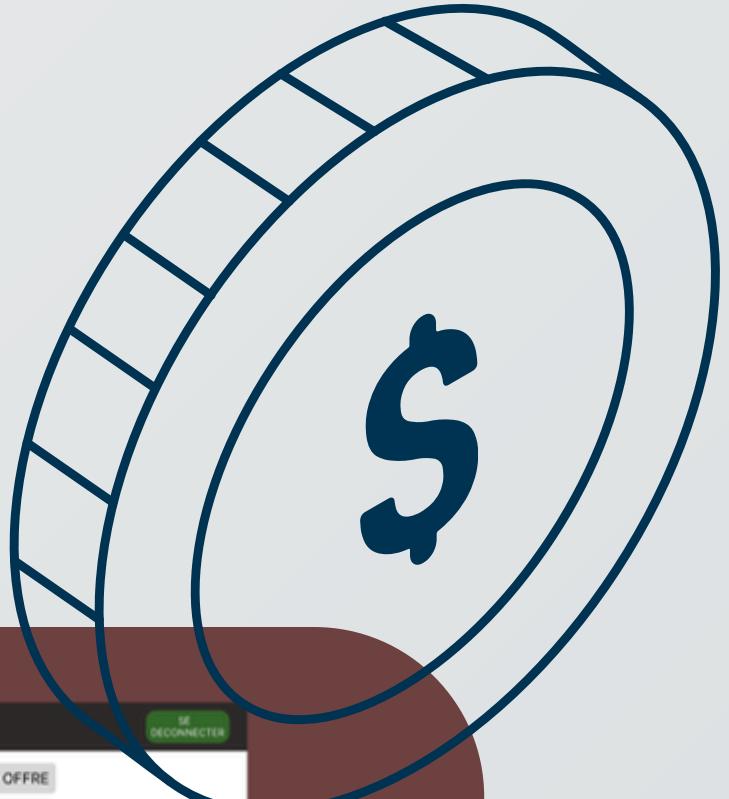
J'ai choisi d'utiliser FIGMA comme principal outil de conception. FIGMA est un logiciel de design collaboratif en ligne qui permet de créer des interfaces utilisateurs et des prototypes interactifs. Grâce à son interface intuitive et à ses nombreuses fonctionnalités avancées, j'ai pu créer le rendu filaire du futur site en tentant de respecter les normes modernes en matière de design web. FIGMA m'a également offert un large éventail de plugins et d'intégrations, qui ont été extrêmement utiles pour optimiser mon flux de travail. Par exemple, l'utilisation de plugins pour gérer les grilles et les alignements ainsi que les plugins d'intégrations d'icônes entre autres.



Figma

Wireframing

Rendu du maquettage



ADMIN

DASHBOARD

BONJOUR, "NAME" - accueil offres compagny dashboard contact profil

SE DECONNECTER

Admin Dashboard

- Home
- Users
- Offers
- Company

les 3 Users les plus récents

ID	NOM	PRENOM	EMAIL	POSTE	ROLE
37	Fontaine	Lucas	lucasfontaine@company.com	Lucasdp	company
36	Lemoine	Clara	clara.lemoine@company.com	Claradp	company
35	Gamer	Marc	marc.gamer@company.com	Marcdp	company

Tableau de bord Admin

- Utilisateurs : 14 utilisateurs actifs
- Offres : 10 Offres
- Entreprises : 5 Entreprises

© 2020 - Tous droits réservés

USERS

BONJOUR, "NAME" - accueil offres compagny dashboard contact profil

SE DECONNECTER

Admin Dashboard

- Home
- Users
- Offers
- Company

les Users

ID	NOM	PRENOM	EMAIL	POSTE	ROLE	ACTION
2	Martin	Sophie	sophie.martin@example.com	sophiedp	client	
3	Lemoine	François	francois.lemoine@example.com	francoisdp	client	
5	Bernard	Lise	lise.bernard@example.com	lisadp	client	
11	Eléonore	Fayolle	fayolle.eleonore@example.com	fayolledp	admin	
30	Laurent	Eloïse	eloise.laurent@example.com	eloisedp	company	

© 2020 - Tous droits réservés

COMPANY

BONJOUR, "NAME" - accueil offres compagny dashboard contact profil

SE DECONNECTER

Admin Dashboard

- Home
- Users
- Offers
- Company

les Compagnies

ID	NOM	SECTEUR	ACTION
1000	TechNova Solutions	Information Technology	
1001	GreenFields AgriCo	Agriculture	
1002	EduPro Learning	Education	
1003	UnderConstruct LS	Construction	
1004	MediCare Plus	Healthcare	

© 2020 - Tous droits réservés

OFFERS

BONJOUR, "NAME" - accueil offres compagny dashboard contact profil

SE DECONNECTER

Admin Dashboard

- Home
- Users
- Offers
- Company

TOUTES LES OFFRES

LOGO COMPANY	POSTE	CONTRAT	ACTION
	Ingénieur DevOps	CDI	
	Data Scientist	CDI	
	Développeur PHP Backend	FREELANCE	
	Développeur Full Stack Java Spring - Application B2B	FREELANCE	
	Développeur Front-End React + Startup	CDI	

© 2020 - Tous droits réservés

OFFRES

BONJOUR, "NAME" - accueil offres compagny contact

SE DECONNECTER

TOUTES LES OFFRES

LOGO COMPANY	POSTE	CONTRAT	ACTION
	Ingénieur DevOps	CDI	
	Data Scientist	CDI	
	Développeur PHP Backend	FREELANCE	
	Développeur Full Stack Java Spring - Application B2B	FREELANCE	
	Développeur Front-End React + Startup	CDI	

© 2020 - Tous droits réservés

DETAILS OFFRE

BONJOUR, "NAME" - accueil offres compagny contact

SE DECONNECTER

Ingénieur DevOps

Description

Mission

Ville : Ville :
Enterprise : Contrat :
Technologie : Avantages :
Nombre de candidats : Date de création :

Postuler à cette offre

© 2020 - Tous droits réservés

AJOUT OFFRE

BONJOUR, "NAME" - accueil offres compagny contact

SE DECONNECTER

AJOUT D'UNE NOUVELLE OFFRE

Logo :
Nom :
Prénom :
Email :
Ville :
Contrat :
Technologie :
Avantages :
Nombre de candidats :
Date de création :
Postuler à cette offre

© 2020 - Tous droits réservés

MODIF OFFRE

BONJOUR, "NAME" - accueil offres compagny contact

SE DECONNECTER

MODIFIER UNE OFFRE

Logo :
Nom :
Prénom :
Email :
Ville :
Contrat :
Technologie :
Avantages :
Nombre de candidats :
Date de création :
Postuler à cette offre

© 2020 - Tous droits réservés

POSTULER OFFRE

BONJOUR, "NAME" - accueil offres compagny contact

SE DECONNECTER

Postuler Maintenant !

NOM :
PRÉNOM :
TELEPHONE :
VILLE :
CODE POSTAL :
MOTIVATION :
Importez votre Lettre de Motivation :
Postuler

L'intégration statique

Avant-propos

Après avoir réalisé mes maquettes, je me suis servi de la combinaison de deux langages, l'un de balisage avec **HTML5** et l'autre de déclaration avec **CSS3** natif pour définir les styles.

Cela a permis de rendre mon projet responsive et m'a offert un panel de balises et de classes préconçues qui ont optimisé mon avancée. L'utilisation de **Git** pour le versionnement a été nécessaire pour gérer l'intégrité du code, assurer un développement fluide et organisé, et me familiariser avec cet outil essentiel.

L'intégration statique

Avant-propos

Après avoir réalisé mes maquettes, je me suis servi de la combinaison de deux langages, l'un de balisage avec **HTML5** et l'autre de déclaration avec **CSS3** natif pour définir les styles.

Cela a permis de rendre mon projet responsive et m'a offert un panel de balises et de classes préconçues qui ont optimisées mon avancée. L'utilisation de **Git** pour le versionnement a été nécessaire pour gérer l'intégrité du code, assurer un développement fluide et organisé, et me familiariser avec cet outil essentiel.

GitHub

GitHub c'est une plateforme basée sur le cloud où nous pouvons stocker, partager et travailler avec d'autres pour écrire du code. Présenter ou partager votre travail. Suivre et gérer les modifications apportées à votre code au fil du temps.

```
git init  
git add .  
git commit -m "Initial commit"  
git remote add origin https://github.com/AhmedTwo/portal_job.git  
git push -u origin main # ou master selon le nom de ta branche
```

```
git init  
git add .  
git commit -m "Initial commit"  
git remote add origin https://github.com/AhmedTwo/portal_job.git  
git push -u origin main # ou master selon le nom de ta branche
```

```
C:\wamp64\www\portal_job>git push -f origin master  
Enumerating objects: 155, done.  
Counting objects: 100% (155/155), done.  
Delta compression using up to 16 threads  
Compressing objects: 100% (152/152), done.  
Writing objects: 100% (155/155), 1.37 MiB | 4.23 MiB/s, done.  
Total 155 (delta 17), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (17/17), done.  
To https://github.com/AhmedTwo/portal_job.git  
+ e0e7d17...5a9a260 master -> master (forced update)
```

Collaborators and teams

Public repository This repository is public and visible to anyone

Manage visibility

Direct access 0 collaborators have access to this repository. Only you can contribute to this repository.

Add people to portal_job

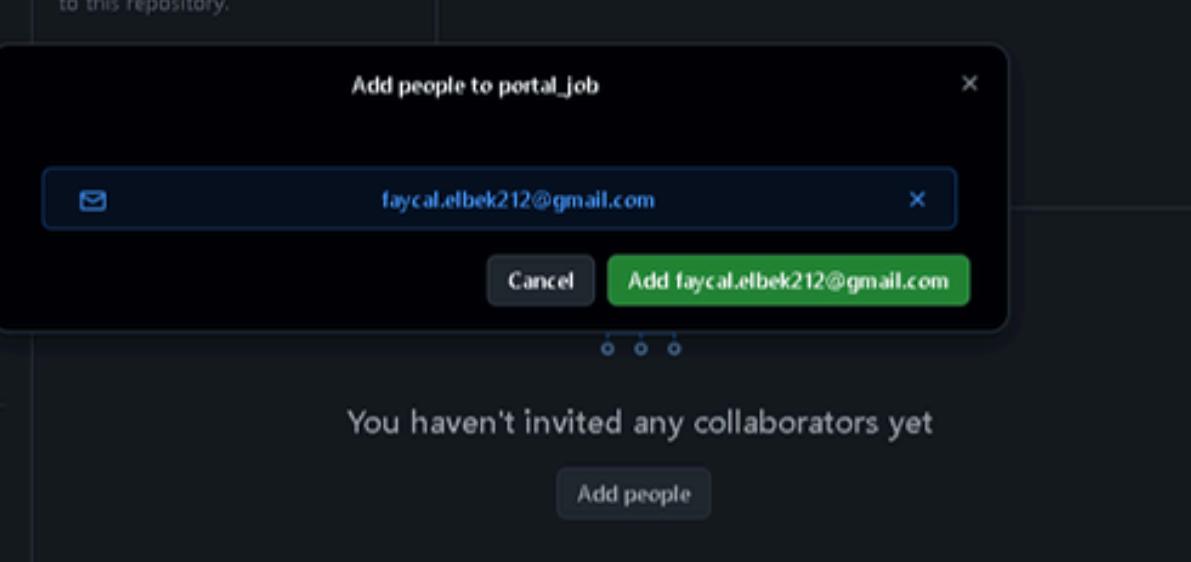
faycal.elbek212@gmail.com

Add faycal.elbek212@gmail.com

You haven't invited any collaborators yet

Add people

Cancel



AppFrontV2 Public

main 1 Branch 0 Tags

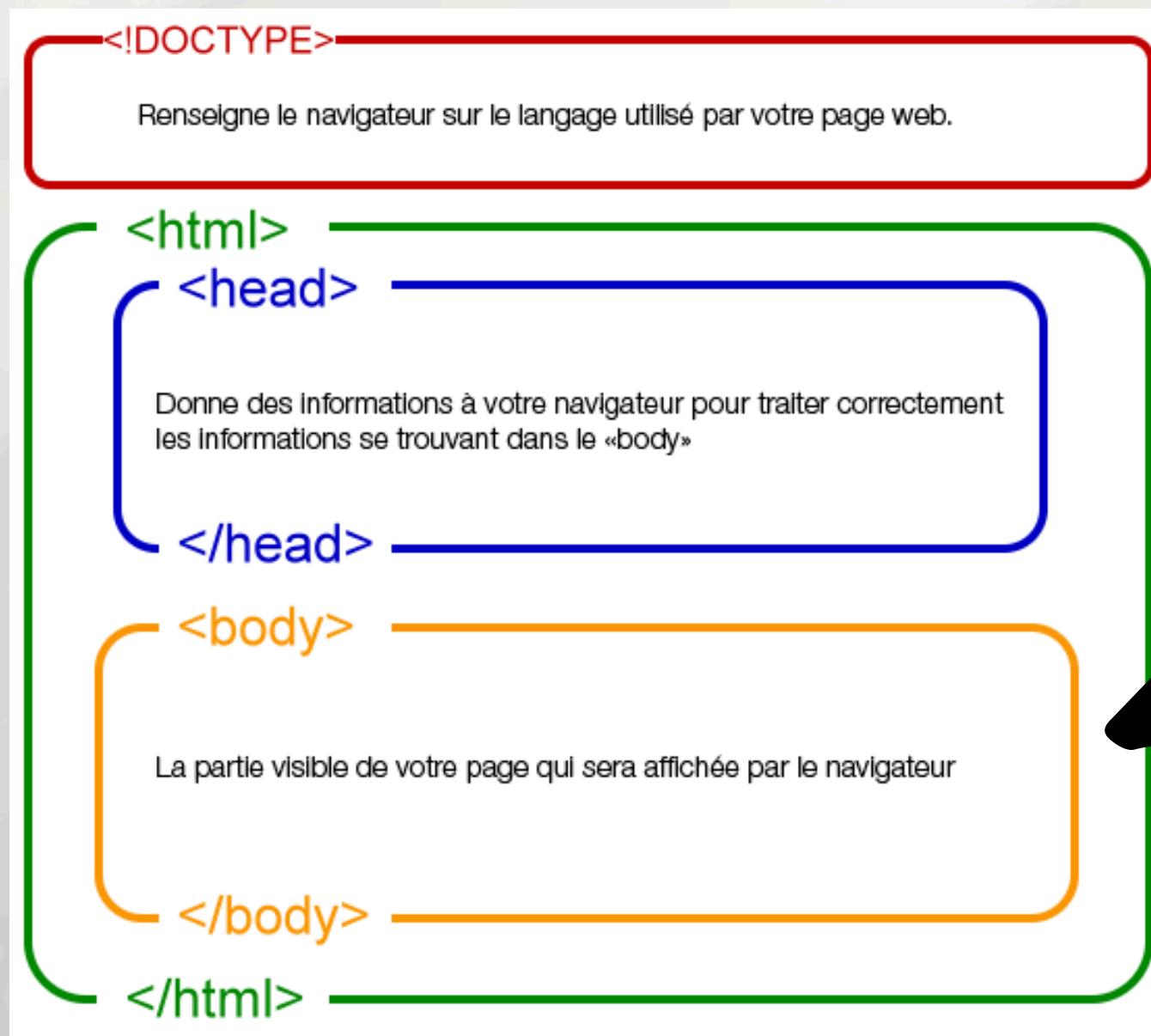
Go to file Add file Code

AhmedTwo DossierMemoire manque details filtre&search & hebergement 791aeda · yesterday 21 Commits

File	Commit Message	Time
.vscode	fisrt commit	last month
public	fisrt commit	last month
src	favorite&filtered	yesterday
.env	fisrt commit	last month
.env.exemple	auth	3 weeks ago
.gitattributes	fisrt commit	last month
.gitignore	fisrt commit	last month
.prettierrc.json	fisrt commit	last month
Projet Ahmed Memoire.pdf	DossierMemoire manque details filtre&search & hebergement	yesterday
README.md	fisrt commit	last month
index.html	fisrt commit	last month
jsconfig.json	fisrt commit	last month
package-lock.json	GET	3 weeks ago
package.json	GET	3 weeks ago
vite.config.js	fisrt commit	last month
vitest.config.js	fisrt commit	last month

HTML5

HyperText Markup Language



C'est ici que je compte mettre mes balises
html afin de structurer ma page comme je
le souhaite

<header>, <nav>, <main>, et <footer>

HTML5 & la sémantique par le code

```
<header id="headerOne">
  <nav id="navOne">
    <div class="nav-left" v-if="!userStore.isAuthenticated"> ...
    </div>

    <div class="nav-left" v-if="userStore.isAuthenticated"> ...
    </div>

    <ul class="menu" v-if="!userStore.isAuthenticated"> ...
    </ul>

    <ul class="menu" v-if="userStore.isAuthenticated"> ...
    </ul>

    <div class="nav-right" v-if="!userStore.isAuthenticated"> ...
    </div>

    <div class="nav-right" v-if="userStore.isAuthenticated">
      <div class="user-info">
        <a href="/favoris" class="heart-link" title="Mes favoris">♥</a>
      </div>
      <div class="user-dropdown"> ...
      </div>
    </div>
  </nav>
</header>
```

CSS3

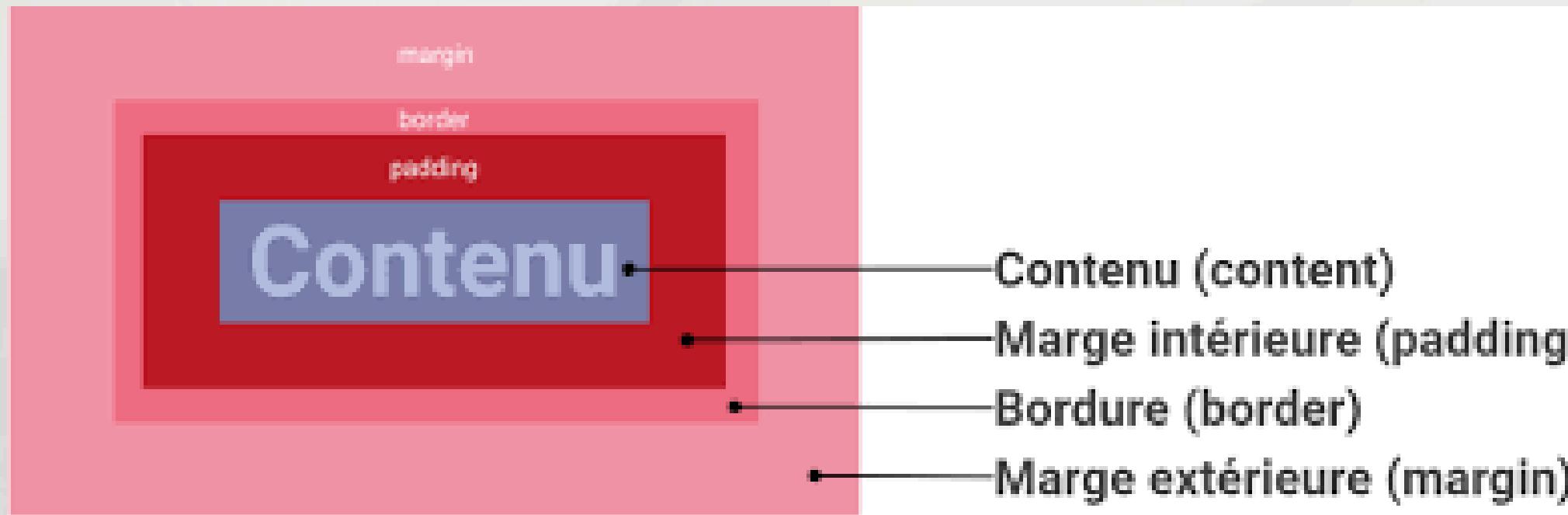
Cascading Style Sheets

J'ai utilisé le langage **CSS3** pour mettre en forme et styliser l'intégralité du site **Portal_job**. Le **CSS** est essentiel pour séparer la présentation (**couleurs, polices, mise en page, etc.**) de la structure **HTML**, assurant ainsi une maintenance plus facile et un code plus clair.

Un aspect crucial de mon travail a été l'adoption d'un design adaptatif (**Responsive Design**). J'ai implémenté les **Media Queries** afin que l'affichage du site s'ajuste parfaitement à toutes les tailles d'écran (ordinateurs, tablettes et mobiles), garantissant une expérience utilisateur optimale sur tous les supports.

CSS3

Cascading Style Sheets



```
<style scoped>

  .logo img {
    width: 250px;
  }

  /* HEADER */
  #headerOne {
    background-color: var(--bg);
    box-shadow: 0 6px 16px var(--shadow);
    border-bottom: 1px solid #ccc;
    position: sticky;
    top: 0;
    z-index: 1000;
  }

  #navOne {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem 3rem;
    background-color: #rgb(240, 240, 240);
  }

  /* LOGO */
  .logo img {
    width: 200px;
  }

  /* MENU PRINCIPAL */
  .menu {
    list-style: none;
    display: flex;
    gap: 2.5rem;
    margin: 0;
    padding: 0;
  }
</style>
```

CSS3

```
<style scoped>

  .logo img {
    width: 250px;
  }

  /* HEADER */
  #headerOne {
    background-color: var(--bg);
    box-shadow: 0 6px 16px var(--shadow);
    border-bottom: 1px solid ■#ccc;
    position: sticky;
    top: 0;
    z-index: 1000;
  }

  #navOne {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem 3rem;
    background-color: ■rgb(240, 240, 240);
  }

  /* LOGO */
  .logo img {
    width: 200px;
  }

  /* MENU PRINCIPAL */
  .menu {
    list-style: none;
    display: flex;
    gap: 2.5rem;
    margin: 0;
    padding: 0;
  }

</style>
```

```
<style scoped>

  .logo img {
    width: 250px;
  }

  /* HEADER */
  #headerOne {
    background-color: var(--bg);
    box-shadow: 0 6px 16px var(--shadow);
    border-bottom: 1px solid ■#ccc;
    position: sticky;
    top: 0;
    z-index: 1000;
  }

  #navOne {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem 3rem;
    background-color: ■rgb(240, 240, 240);
  }

  /* LOGO */
  .logo img {
    width: 200px;
  }

  /* MENU PRINCIPAL */
  .menu {
    list-style: none;
    display: flex;
    gap: 2.5rem;
    margin: 0;
    padding: 0;
  }

</style>
```

```
<style scoped>

  .logo img {
    width: 250px;
  }

  /* HEADER */
  #headerOne {
    background-color: var(--bg);
    box-shadow: 0 6px 16px var(--shadow);
    border-bottom: 1px solid ■#ccc;
    position: sticky;
    top: 0;
    z-index: 1000;
  }

  #navOne {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem 3rem;
    background-color: ■rgb(240, 240, 240);
  }

  /* LOGO */
  .logo img {
    width: 200px;
  }

  /* MENU PRINCIPAL */
  .menu {
    list-style: none;
    display: flex;
    gap: 2.5rem;
    margin: 0;
    padding: 0;
  }

</style>
```

Conception Réactive (Responsive Design)

L'un des objectifs clés était d'améliorer l'expérience utilisateur et de moderniser le site. Cela passe nécessairement par le **Responsive Design** (design adaptatif).

Principe :

Le site s'adapte parfaitement à toutes les tailles d'écran (mobiles, tablettes, ordinateurs)

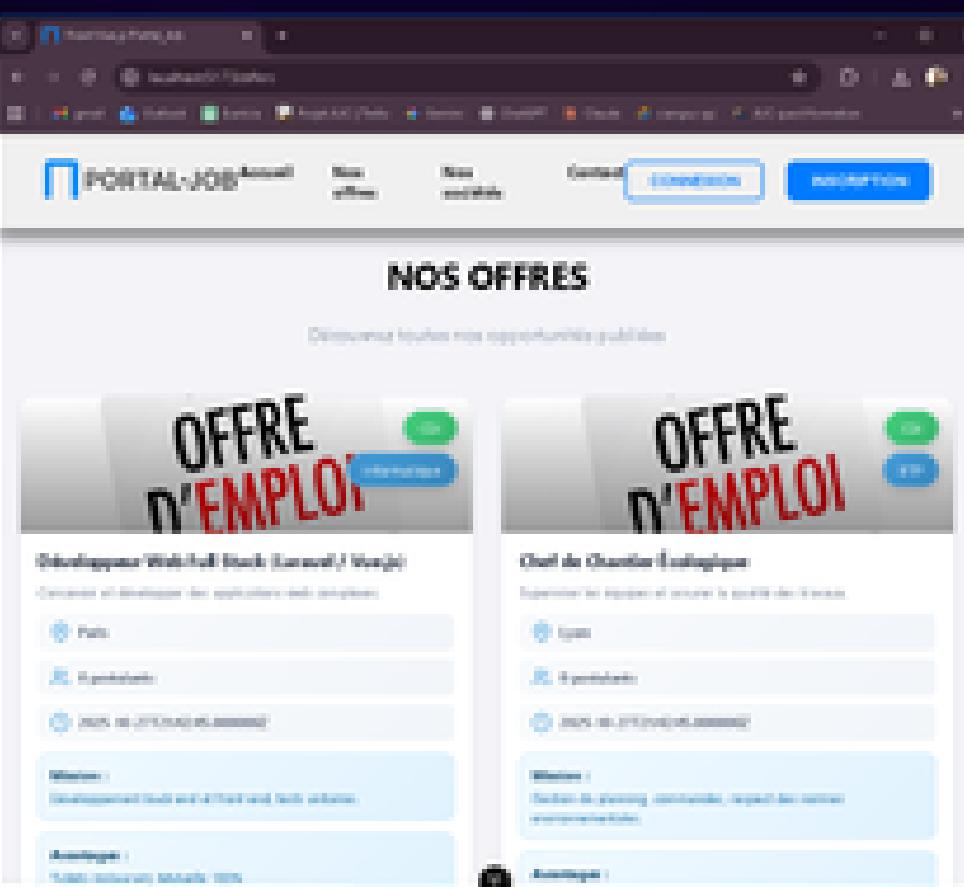
Mise en œuvre :

J'ai utilisé le CSS3 natif et implémenté les Media Queries pour garantir une interface conviviale et intuitive pour tous les utilisateurs

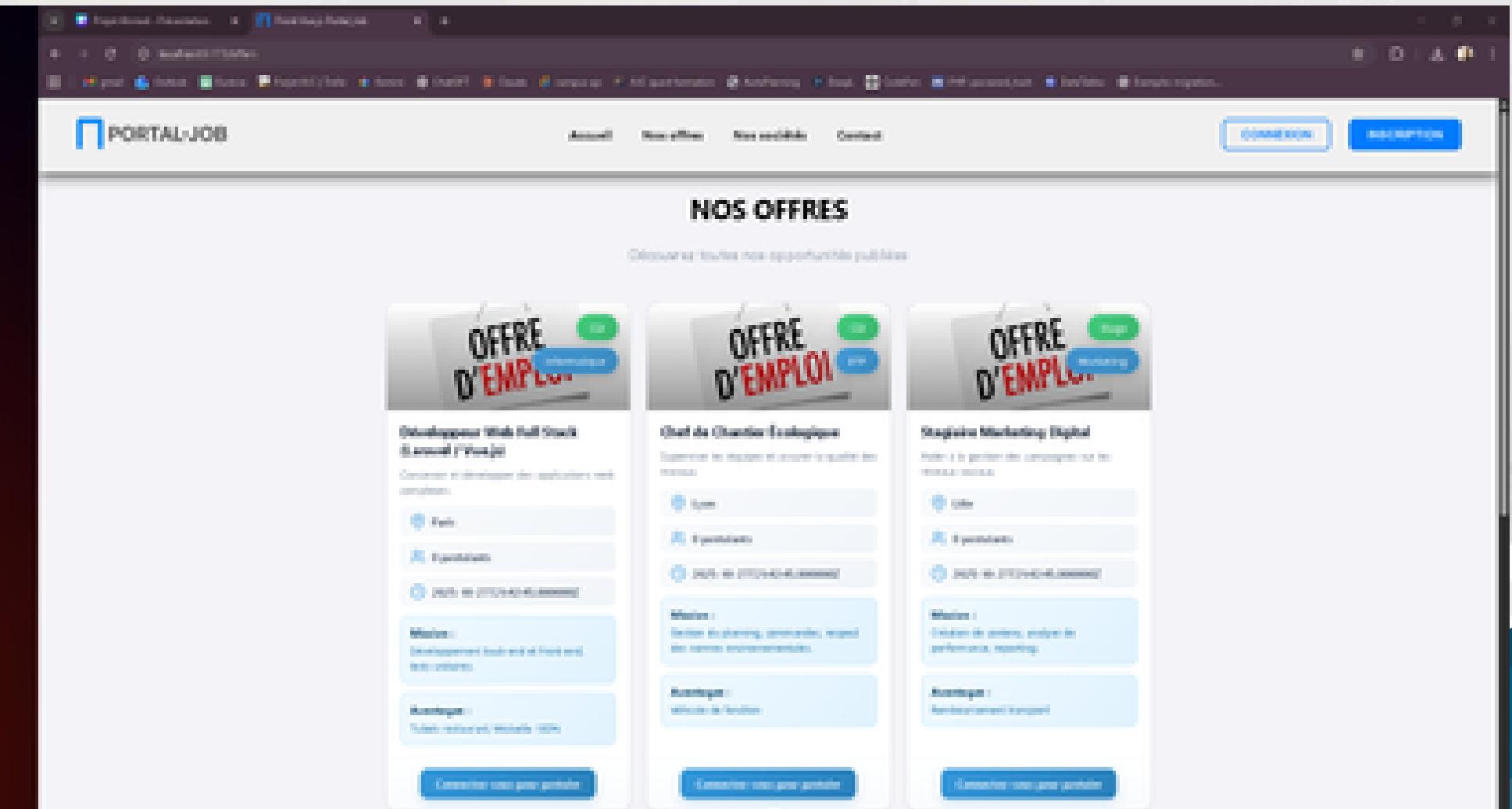
Bénéfice :

Cette approche assure une expérience utilisateur optimale sur tous les supports, ce qui est crucial pour un site d'offres d'emploi.

D'un côté, vous votez le site sous une fenêtre retrecie



De l'autre, le site plein écran

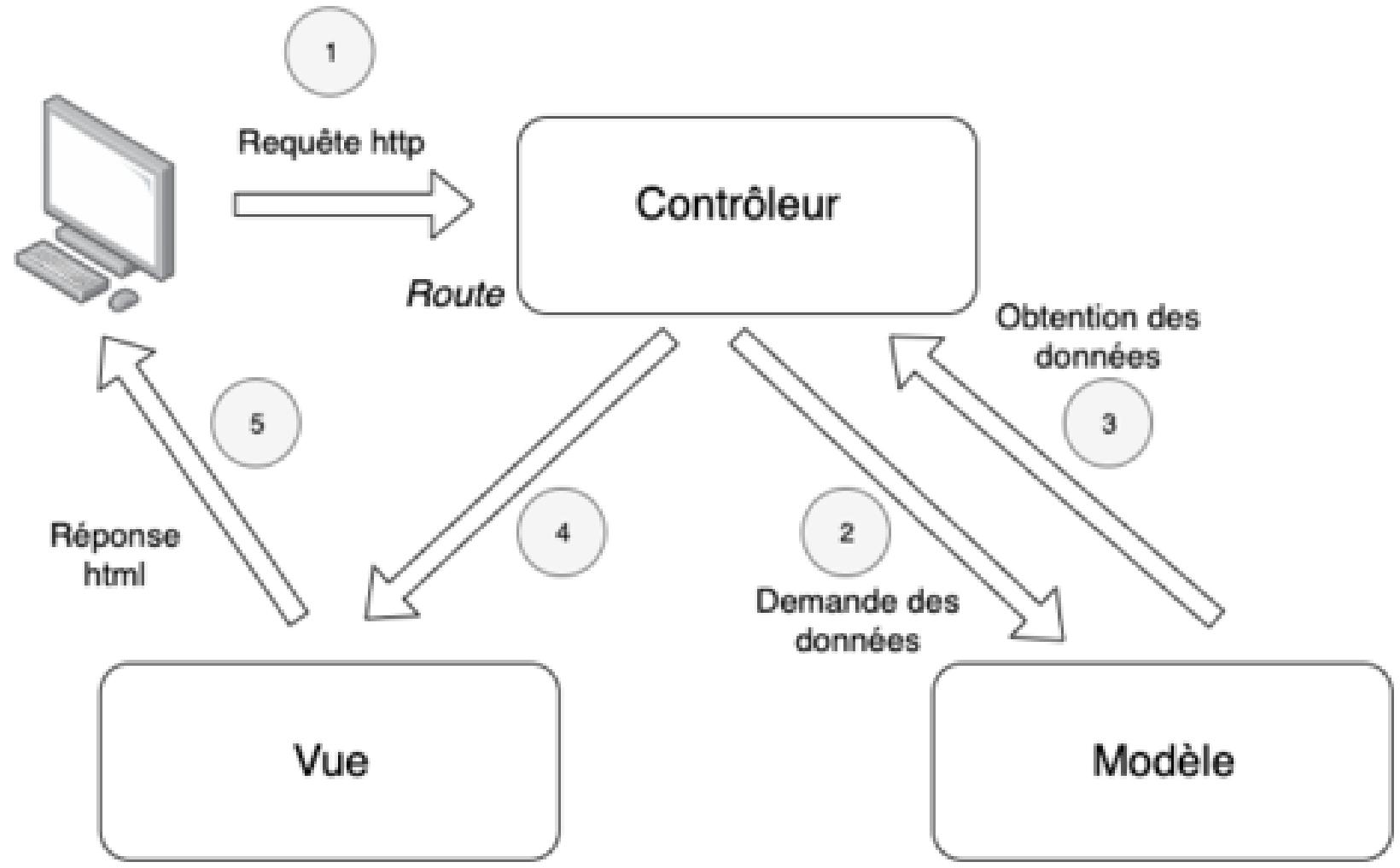


Historique du projet

Du procédurale à l'MVC

Au cours du développement de mon projet, j'ai relevé le défi de le transformer d'une approche procédurale à un modèle basé sur **MVC / programmation orientée objet (POO)** que ça soit pour le **front (Vue.js)** ou le **back (Laravel)**.

Cette transition m'a permis de structurer le code de manière plus modulaire et maintenable. J'ai séparé clairement les responsabilités de l'application en couches distinctes.



1. La Requête (Route) :
L'utilisateur interagit avec l'application, ce qui déclenche une requête HTTP. Cette requête est captée par le Contrôleur.

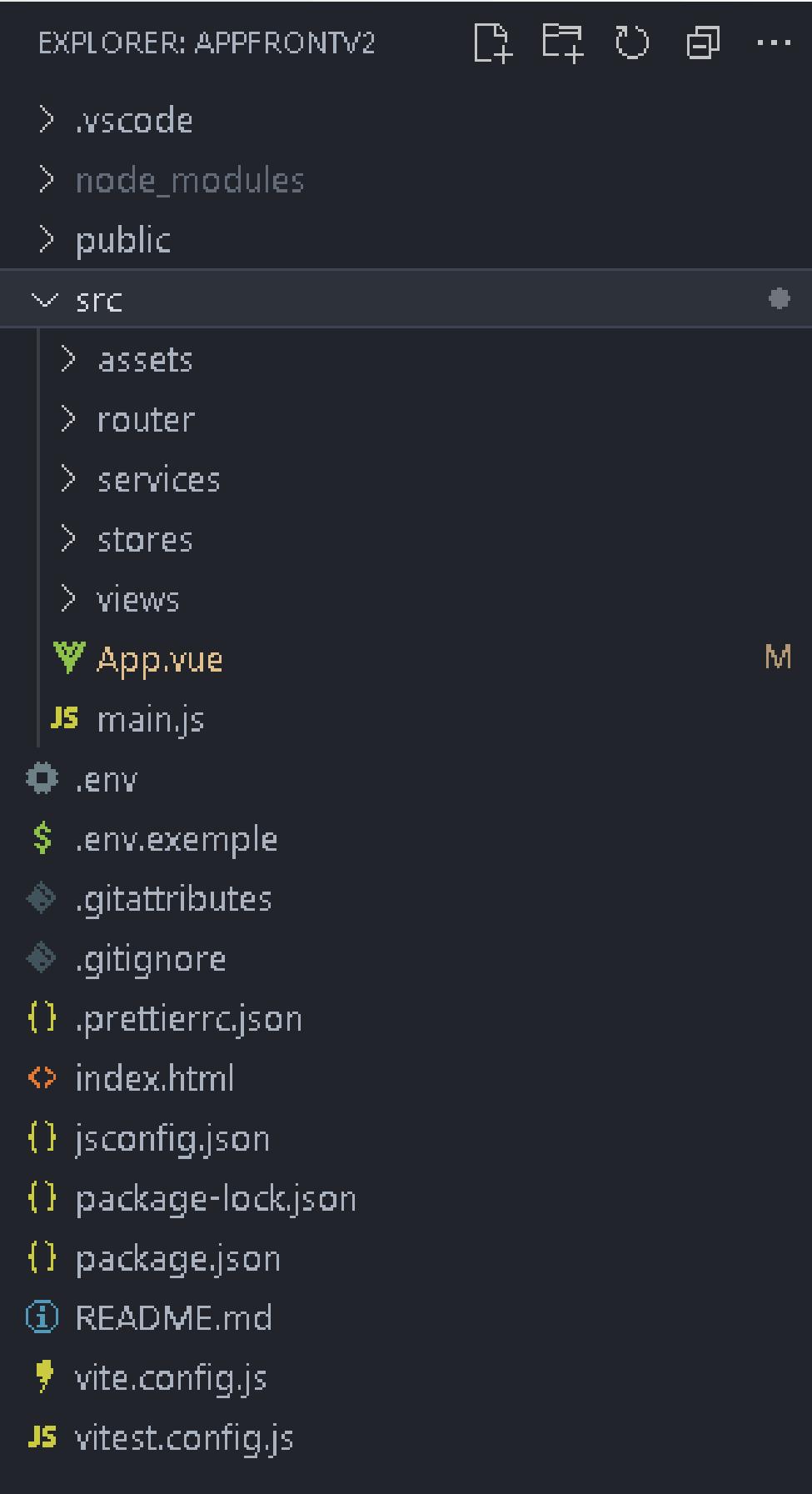
2. Le Contrôleur Orchestre :
Le Contrôleur (la "colle" de l'application) reçoit la requête HTTP et détermine l'action à réaliser. Il demande les données nécessaires au Modèle.

4. La Vue est Préparée :

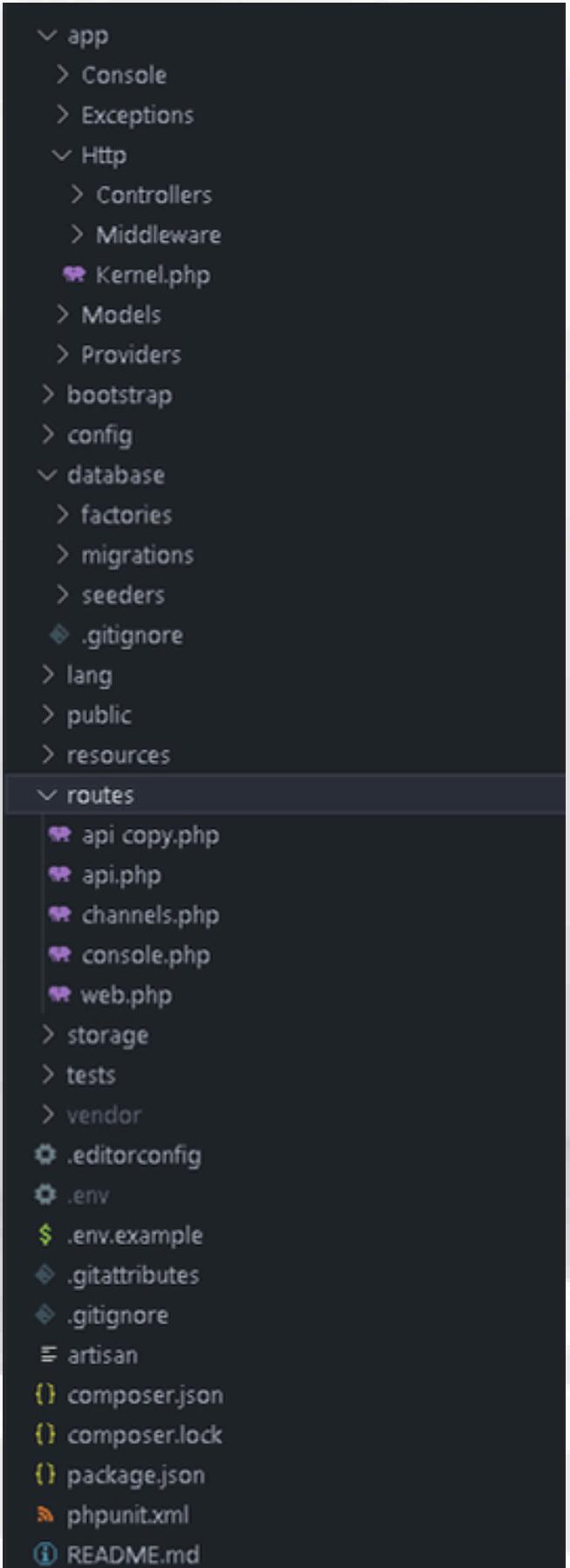
Le Contrôleur transmet les données formatées à la Vue (la couche de présentation).
La Vue est responsable de générer l'interface utilisateur (HTML).

5. La Réponse est Servie :

La Vue retourne la Réponse HTML complète qui est affichée à l'utilisateur selon la requête initiale.



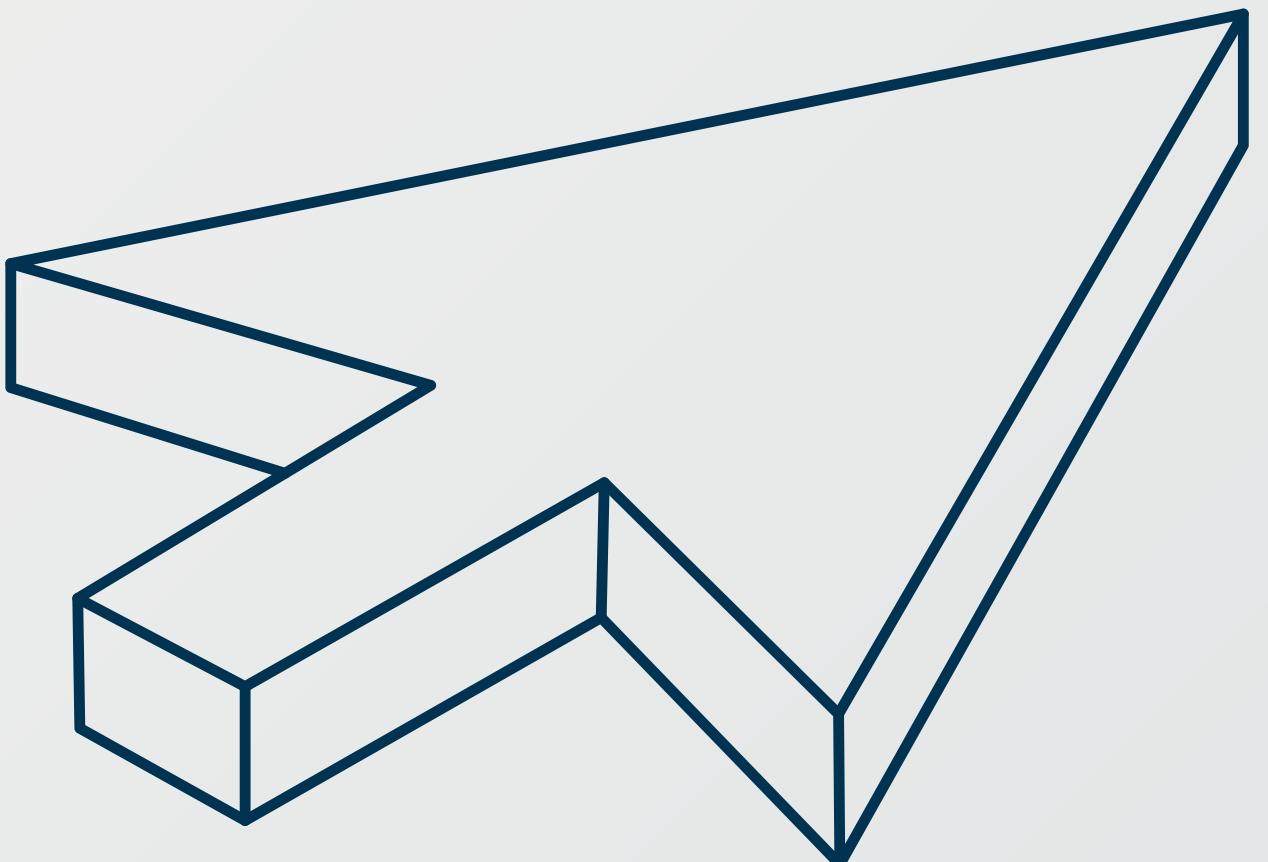
Front Vue.js



Back Laravel

III -

PARTIE BACKEND



La base de données

Avant-propos

Nous débuterons par la création d'un dictionnaire de données exhaustif, suivi la schématisation visuelle de la base de données pour définir ses structures et ses relations clés.

Cette approche méthodique garantit une base de données cohérente et efficace pour le système de gestion de [Portal_Job](#).

Schématisation de la BDD

MLD Modèle Logique de Données

Pour schématiser visuellement ma base de données et capturer ses relations et structures clés, j'ai utilisé un outil de modélisation de bases de données qui offrent des diagrammes : 'nom de l'outil'

LUNA Modeler a permis la schématisation visuelle pour définir les structures et les relations clés de la BDD

Je vais utiliser **LUNA Modeler** ([plan gratuit](#) / [linux](#))

USERS	
id	integer NN
nom	varchar(191)
prenom	varchar(191)
email	varchar(191)
password	varchar(191)
role	varchar(191)
telephone	varchar(191)
ville	varchar(191)
code_postal	varchar(191)
cv_pdf	varchar(191)
qualification	varchar(191)
preference	varchar(191)
disponibilite	boolean
photo	varchar(191)
created_at	timestamp
updated_at	timestamp
company_id	integer NN

REQUESTS	
id	integer NN
title	varchar(191)
description	varchar(191)
type	varchar(191)
status	varchar(191)
created_at	timestamp
updated_at	timestamp
user_id	integer

COMPANYS	
id	integer NN
name	varchar(191)
logo	varchar(191)
number_of_employees	integer
industry	varchar(191)
address	text
latitude	double precision(8,2)
longitude	double precision(8,2)
description	text
email_company	varchar(191)
n_siret	varchar(191)
status	varchar(191)
created_at	timestamp
updated_at	timestamp

FAVORIS	
id	integer NN
user_id	integer
offer_id	integer
created_at	timestamp
updated_at	timestamp

OFFERS	
id	integer NN
title	varchar(191)
description	text
mission	text
location	varchar(191)
category	varchar(191)
employment_type_id	boolean
technologies_used	text
benefits	text
participants_count	integer
image_url	varchar(191)
created_at	timestamp
updated_at	timestamp
id_company	integer

EMPLOYMENT_TYPE	
id	integer NN
name	varchar(191)
created_at	timestamp
updated_at	timestamp

Cardinalité	Définition courte
0-N	Peut avoir plusieurs, optionnel
1-N	Un lié à plusieurs obligatoirement
0-1	Zéro ou une seule fois
1-0	Relation obligatoire d'un côté
1-1	Un seul avec un seul
N-N	Plusieurs liés à plusieurs
0-0	Aucun lien possible (vide)

PhpMyAdmin

Bienvenue dans phpMyAdmin

Langue (Language)

Français - French

Connexion

Utilisateur : root

Mot de passe :

Choix du serveur : MySQL

Connexion

PhpMyAdmin est une application web gratuite qui fournit une interface utilisateur graphique pour gérer les bases de données MySQL et MariaDB via un navigateur.

Il permet aux développeurs et aux administrateurs d'effectuer facilement toutes les opérations de gestion (création, modification, suppression de tables et de données CRUD) sans avoir à utiliser de lignes de commande.

phpMyAdmin

Serveur courant : MySQL

Récentes Préférées

Nouvelle base de données

+ cars_db

+ contact

+ gestioncommandes

+ information_schema

- job_portal

+ job_portal

- job_portal_laravel

+ Nouvelle table

+ companys

+ employment_type

+ failed_jobs

+ favoris

+ migrations

+ offers

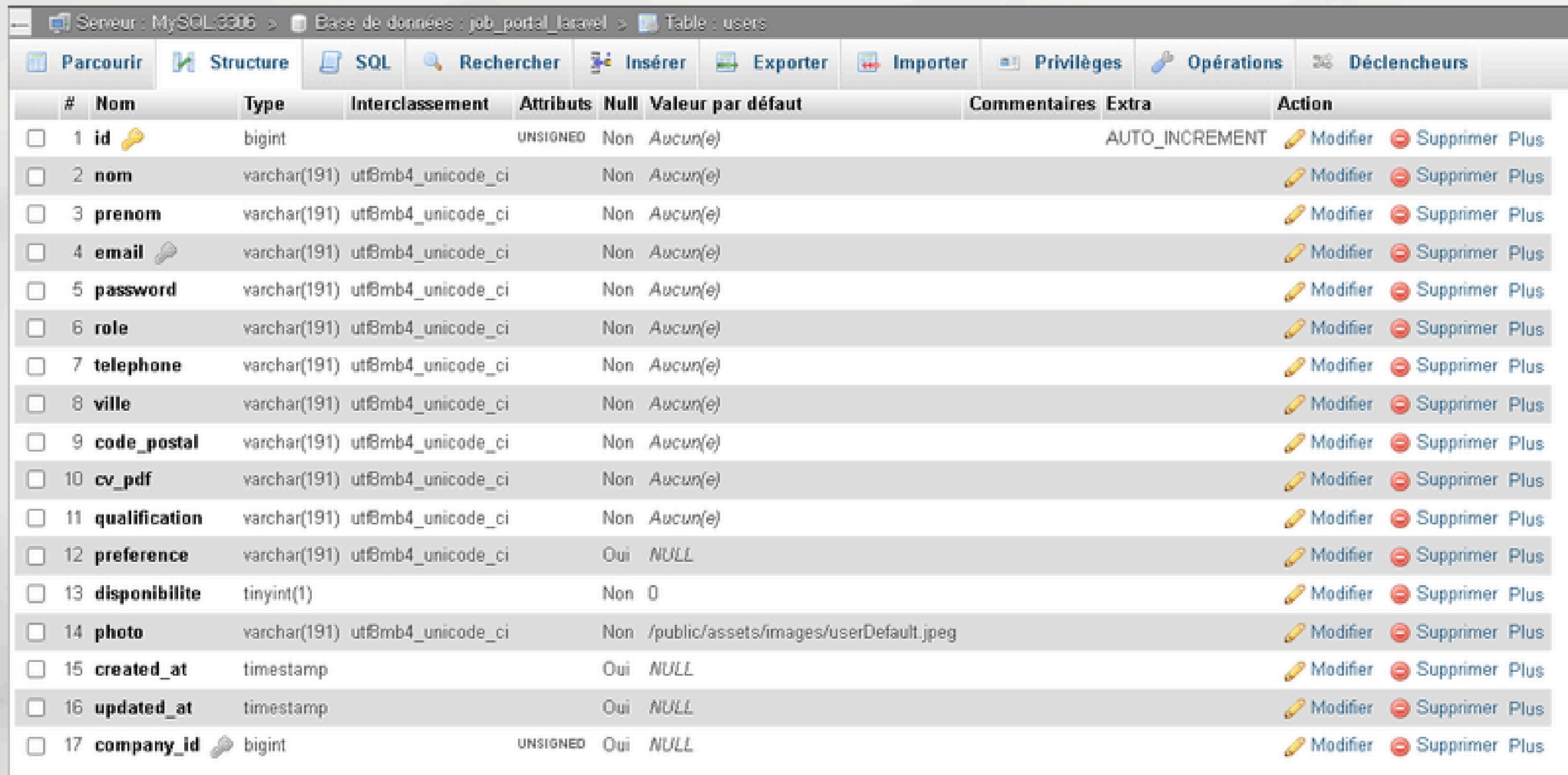
+ password_resets

+ personal_access_tokens

+ requests

+ users

Dans une base de données relationnelle, une **table** est l'élément structurel principal, organisant les données en **colonnes** (représentant les attributs ou champs) et en **lignes**. Un **tuple** est simplement le terme technique pour désigner une **ligne unique** dans cette table, représentant un enregistrement complet et spécifique (par exemple, les informations complètes d'un seul employé ou d'une seule offre d'emploi).



The screenshot shows the 'Structure' tab of the MySQL Workbench interface for the 'users' table. The table has 17 columns:

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id 🛡	bigint		UNSIGNED	Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
2	nom	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
3	prenom	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
4	email 🔐	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
5	password	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
6	role	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
7	telephone	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
8	ville	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
9	code_postal	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
10	cv_pdf	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
11	qualification	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
12	preference	varchar(191)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
13	disponibilite	tinyint(1)			Non	0			Modifier Supprimer Plus
14	photo	varchar(191)	utf8mb4_unicode_ci		Non	/public/assets/images/userDefault.jpeg			Modifier Supprimer Plus
15	created_at	timestamp			Oui	NULL			Modifier Supprimer Plus
16	updated_at	timestamp			Oui	NULL			Modifier Supprimer Plus
17	company_id 🛡	bigint		UNSIGNED	Oui	NULL			Modifier Supprimer Plus

La connexion à la base de données

Au lieu d'utiliser une fonction `getPDO()` dans un fichier `database.php`,

la connexion à la base de données est gérée nativement par `Laravel`.

Le framework utilise les `variables d'environnement` définies dans le fichier `.env` (`DB_CONNECTION`, `DB_HOST`, etc.) pour établir automatiquement la connexion sécurisée via `Eloquent` (l'ORM de `Laravel`).

Fichier : `.env` contient

```
DB_CONNECTION=mysql  
DB_HOST=127.0.0.1  
DB_PORT=3306  
DB_DATABASE=job_portal_laravel  
DB_USERNAME=root  
DB_PASSWORD=
```

Développement du CRUD

Avant-propos

Explorons la mise en place d'un système **CRUD (**Create, Read, Update, Delete**) en utilisant la **Programmation Orientée Objet (POO)**.**

Ce système permet de gérer les opérations de base sur les données tout en assurant un code clair, réutilisable et maintenable.

Pour illustrer cela, nous utiliserons un modèle de classe représentant les entités clés de notre base de données, chaque classe possédant les méthodes nécessaires à chaque opération **CRUD.**

A présent, présentons les opérations de **CRUD via le modèle **MVC**, pour chacune de ses méthodes, une explication est fournie.**

(CRUD) CREATE

Les opérations de Create (offers) :

La méthode **CREATE** est la première étape du **CRUD**, permettant d'insérer de nouvelles données (**un nouveau tuple**) dans la base de données (par exemple, ajouter une nouvelle offre d'emploi). Elle prend en entrée les informations fournies par l'utilisateur et les prépare de manière sécurisée avant de les enregistrer dans la table correspondante.

```
public function addOffer(Request $requestParam)
{
    $validatedData = $requestParam->validate([
        'title'          => 'required|string|max:255',
        'description'   => 'required|string|max:2000',
        'mission'        => 'required|string|max:255',
        'location'       => 'required|string|max:255',
        'category'       => 'required|string|max:255',
        'employment_type_id' => 'required|integer|exists:employment_type,id',
        'technologies_used' => 'required|string|max:255',
        'benefits'        => 'required|string|max:255',
        'image_url'       => 'required|file|mimes:jpeg,png,jpg,webp|max:2048',
        'id_company'      => 'required|integer|exists:companys,id',
    ], [...]);
}

try {
    $validatedData['image_url'] = $requestParam->file('image_url')->store('photo_offer', 'public');

    $offer = Offer::create([
        'title'          => $validatedData['title'],
        'description'   => $validatedData['description'],
        'mission'        => $validatedData['mission'],
        'location'       => $validatedData['location'],
        'category'       => $validatedData['category'],
        'employment_type_id' => $validatedData['employment_type_id'],
        'technologies_used' => $validatedData['technologies_used'],
        'benefits'        => $validatedData['benefits'],
        'image_url'       => $validatedData['image_url'],
        'id_company'      => $validatedData['id_company'],
    ]);
}

return response()->json([
    'success' => true,
    'message' => 'Offre créée avec succès.',
    'data' => $offer,
], 201);
} catch (\Exception $e) {
    return response()->json([
        'message' => 'Échec de l\'ajout de l\'offre.',
        'error'   => $e->getMessage(),
    ], 500);
}
```

(CRUD) READ

Les opérations de Read (offers) :

La méthode **READ est essentielle pour récupérer les données existantes stockées dans la base de données afin de les afficher à l'utilisateur (par exemple, lister toutes les offres d'emploi). Elle peut être utilisée pour obtenir tous les enregistrements ou pour filtrer et afficher un seul tuple spécifique sur la base d'un identifiant ou d'un critère de recherche.**

```
public function getOffer()
{
    $data = Offer::select(
        'id',
        'title',
        'description',
        'mission',
        'location',
        'category',
        'employment_type_id', // dans le front on a donc appelé {{ offer.employment_type.name }}
        'technologies_used',
        'benefits',
        'participants_count',
        'image_url',
        'created_at',
        'updated_at',
        'id_company',
    )->with('employment_type')->get();

    return response()->json([
        'success' => true,
        'data' => $data
    ], 200); // code reponse 200 pour success
}
```

(CRUD) UPDATE

Les opérations de Update (offers) :

La méthode UPDATE permet de modifier les données existantes dans un enregistrement spécifique de la base de données (par exemple, changer la description d'une offre d'emploi déjà publiée).

Pour garantir la cohérence, elle nécessite généralement l'identifiant de l'enregistrement à cibler, ainsi que les nouvelles valeurs à appliquer aux champs.

```
public function updateOffer(Request $requestParam, $id)
{
    $offer = Offer::find($id);

    if (!$offer) {
        return response()->json([
            'success' => false,
            'message' => 'Offre non trouvée',
        ], 404);
    }

    $validatedData = $requestParam->validate([
        'title'          => 'sometimes|string|max:255',
        'description'   => 'sometimes|string|max:2000',
        'mission'        => 'sometimes|string|max:255',
        'location'       => 'sometimes|string|max:255',
        'category'       => 'sometimes|string|max:255',
        // Ajouter 'integer' et 'exists' pour la clé étrangère afin de faire le lien avec la table employment_type
        'employment_type_id' => 'sometimes|integer|exists:employment_type,id',
        'technologies_used' => 'sometimes|string|max:255',
        'benefits'        => 'sometimes|string|max:255',
        'image_url'       => 'sometimes|file|mimes:jpeg,png,jpg,webp|max:2048',
    ], [
        ...
    ]);

    $offer->update($validatedData);

    return response()->json([
        'success' => true,
        'message' => 'Offre mise à jour avec succès',
        'data' => $offer
    ], 200);
}
```

(CRUD) DELETE

Les opérations de Delete (offers) :

La méthode **DELETE est utilisée pour supprimer définitivement un ou plusieurs enregistrements de la base de données (par exemple, retirer une offre d'emploi qui n'est plus valide). C'est l'opération la plus sensible, car elle requiert une identification précise du tuple à l'aide de son identifiant unique pour éviter toute perte de données accidentelle.**

```
public function deleteOffer($id)
{
    $offer = Offer::find($id);

    if (!$offer) {
        return response()->json([
            'success' => false,
            'message' => 'Offre non trouvée, impossible de la supprimer',
        ], 404);
    }

    try {
        $offer->delete();

        return response()->json([
            'success' => true,
            'message' => 'Offre supprimée avec succès.'
        ], 200);
    } catch (\Exception $e) {
        return response()->json([
            'message' => 'Echec de la suppression de l\'offre',
            'error'   => $e->getMessage()
        ], 500);
    }
}
```

Postman

Avant-propos

Postman est un environnement de développement API qui agit comme un client essentiel pour tester directement les points d'accès (endpoints**) de votre API Backend. Il permet d'envoyer des requêtes HTTP (telles que GET, POST, PUT, DELETE) et d'inspecter les réponses du serveur pour valider immédiatement le fonctionnement des opérations **CRUD**. L'utilisation de Postman est cruciale pour garantir la fiabilité et la conformité de l'API avant son intégration avec le Frontend.**

```
// tous les rôles non connecté !!!
Route::middleware(['guest'])->group(
    function () {
        Route::post('/login', [AuthController::class, 'login']); // connexion
        Route::post('/addUser', [CompanyController::class, 'addUser']); // inscription

        Route::get('/count', [UserController::class, 'getCount']); // affichage page d'entrée
        Route::get('/allOffer', [OfferController::class, 'getOffer']);
        Route::get('/offerById/{id}', [OfferController::class, 'getOfferById']);

        Route::get('/allCompany', [CompanyController::class, 'getCompany']);
        Route::get('/companyById/{id}', [CompanyController::class, 'getCompanyById']);
        Route::post('/addCompany', [CompanyController::class, 'addCompany']);
    }
);

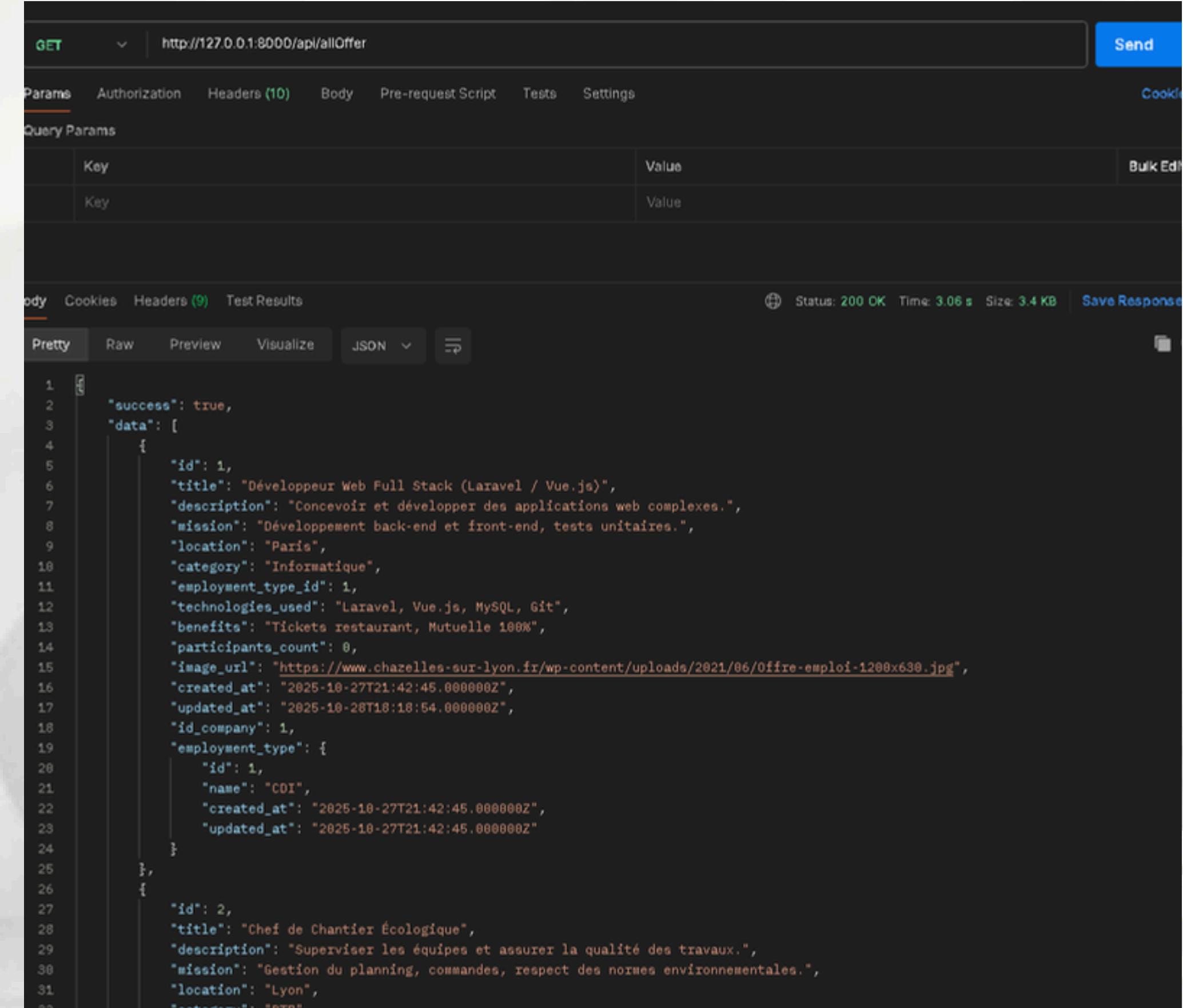
Route::middleware(['auth:sanctum', 'role:admin'])->group(
    function () {
        Route::get('/allUser', [UserController::class, 'getUser']);
        Route::get('/userByRole/{role}', [UserController::class, 'getUserByRole']);
        Route::post('/deleteUser/{id}', [UserController::class, 'deleteUser']);

        Route::get('/allRequest', [RequestController::class, 'getRequest']);
    }
);
```

Flux Postman - API

Postman envoie une Requête HTTP (ex: **GET, POST, PUT, DELETE**) vers une URL spécifique (**endpoint**) de votre **API**.

L'API Backend reçoit la requête, interagit avec la base de données (**CRUD**), puis renvoie une Réponse HTTP contenant le statut (ex: **200 OK**) et les données demandées au client (**Postman**).



The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:8000/api/allOffer`. The response body is a JSON object containing two job offers:

```
1  {
2     "success": true,
3     "data": [
4         {
5             "id": 1,
6             "title": "Développeur Web Full Stack (Laravel / Vue.js)",
7             "description": "Concevoir et développer des applications web complexes.",
8             "mission": "Développement back-end et front-end, tests unitaires.",
9             "location": "Paris",
10            "category": "Informatique",
11            "employment_type_id": 1,
12            "technologies_used": "Laravel, Vue.js, MySQL, Git",
13            "benefits": "Tickets restaurant, Mutuelle 100%",
14            "participants_count": 8,
15            "image_url": "https://www.chazelles-sur-lyon.fr/wp-content/uploads/2021/06/Offre-emploi-1200x630.jpg",
16            "created_at": "2025-10-27T21:42:45.000000Z",
17            "updated_at": "2025-10-28T18:18:54.000000Z",
18            "id_company": 1,
19            "employment_type": {
20                "id": 1,
21                "name": "CDI",
22                "created_at": "2025-10-27T21:42:45.000000Z",
23                "updated_at": "2025-10-27T21:42:45.000000Z"
24            },
25        },
26        {
27            "id": 2,
28            "title": "Chef de Chantier écologique",
29            "description": "Superviser les équipes et assurer la qualité des travaux.",
30            "mission": "Gestion du planning, commandes, respect des normes environnementales.",
31            "location": "Lyon",
32            "category": "Construction"
33        }
34    ]
35 }
```

Composants BackEnd

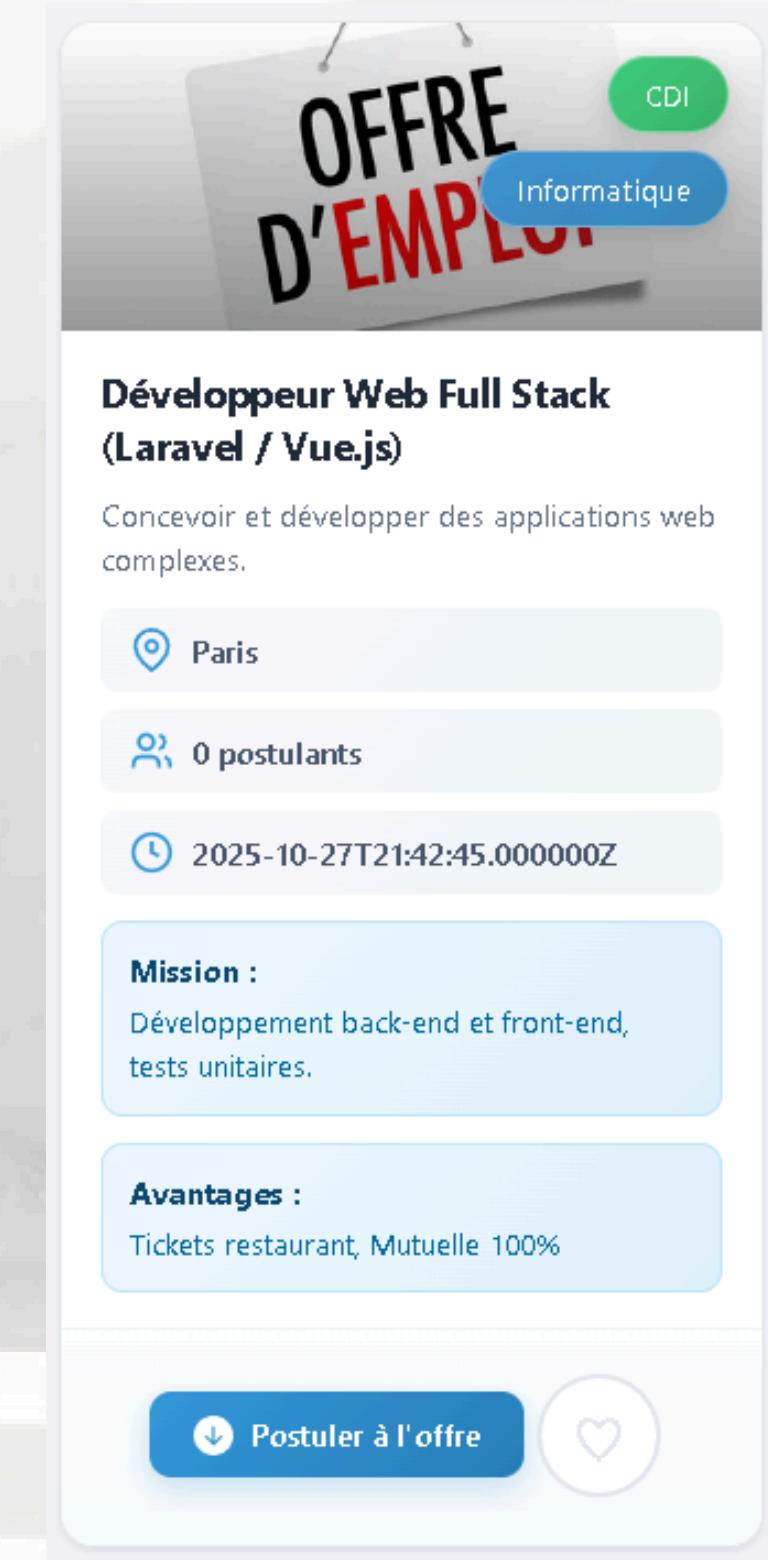
Filtrage des Offres/Sociétés

Pour améliorer l'expérience utilisateur du site en ligne, je vais mettre en place un système simple de filtrage des offres/sociétés.

Cette fonctionnalité permet aux utilisateurs de filtrer par catégorie (ex: contrat, durée ...), rendant la navigation et la recherche de produits plus efficaces

Système de favoris

Le **Système de Favoris** permet aux candidats d'enregistrer et de suivre facilement les offres d'emploi qui les intéressent, sans avoir à postuler immédiatement. Techniquelement, cette fonctionnalité nécessite l'établissement d'une **relation spécifique** entre l'entité Utilisateur (**Candidat**) et l'entité **Offre d'Emploi** dans la base de données. Elle améliore significativement l'expérience utilisateur en offrant un **tableau de bord personnalisé** pour gérer et retrouver rapidement les opportunités.



PhpMailer

Gestion des Emails

J'ai choisi d'intégrer la librairie **PHPMailer** dans mon contrôleur Laravel pour gérer l'envoi des emails de contact, car elle me permet une configuration **SMTP** plus fiable et sécurisée, cruciale pour l'utilisation de services comme Gmail.

Je gère la soumission des données depuis le front-end Vue.js qui appelle une **API** vers ce contrôleur, ce qui me garantit une séparation des préoccupations nette entre la présentation et le traitement serveur.

Mon code s'assure de la sécurité en validant les données, en utilisant les variables d'environnement (.env) pour les identifiants SMTP, et en assurant la robustesse grâce à la gestion des erreurs avec le bloc try-catch.

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=seghiriahmed9@gmail.com
MAIL_PASSWORD= # Le mot de passe de 16 caractères !
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=seghiriahmed9@gmail.com # Doit correspondre à MAIL_USERNAME pour Gmail
MAIL_FROM_NAME="${APP_NAME}"
```

```

class ContactController extends Controller
{
    public function submitContact(Request $request)
    {
        // on valide les des données du form contact.vue
        $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|email|max:255',
            'subject' => 'required|string|max:255',
            'message' => 'required|string',
        ]);

        // on récup ensuite les des données validées
        $name = $request->input('name');
        $fromEmail = $request->input('email');
        $subject = $request->input('subject');
        $messageContent = $request->input('message');

        // Configuration de PHPMailer pour l'envoi
        $mail = new PHPMailer(true);

        try {
            // Configuration SMTP
            $mail->isSMTP();
            $mail->Host = env('MAIL_HOST');
            $mail->SMTPOAuth = true;
            $mail->Username = env('MAIL_USERNAME');
            $mail->Password = env('MAIL_PASSWORD');
            $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
            $mail->Port = env('MAIL_PORT');
            $mail->CharSet = 'UTF-8';

            // Expéditeur technique : Doit correspondre à l'utilisateur SMTP (pour l'authentification)
            // DOIT être MAIL_USERNAME pour Gmail
            $auth_email = env('MAIL_USERNAME');
            $mail->setFrom($auth_email, 'Formulaire Contact Portal_Job');

            // Destinataire : Votre adresse e-mail personnelle/d'administration
            $mail->addAddress('seghiriahmed9@gmail.com');

            // Adresse de Réponse : L'e-mail de l'utilisateur. C'est ici que vous répondez.
            $mail->addReplyTo($fromEmail, $name);

            // Contenu du message
            $mail->isHTML(false);
            $mail->Subject = "Nouveau message de contact : " . $subject;
            $mail->Body = "De: {$name} ({$fromEmail})\n\nSujet: {$subject}\n\nMessage:\n{$messageContent}";

            $mail->send();

            // Succès
            return response()->json(['message' => 'Votre message a été envoyé avec succès !'], 200);
        } catch (Exception $e) {
            // Échec
            // Loggez l'erreur pour la déboguer
            Log::error("Erreur d'envoi PHPMailer: " . $mail->ErrorInfo);

            return response()->json(['error' => "Erreur lors de l'envoi du mail. Veuillez réessayer."], 500);
        }
    }
}

```

\$request->validate(...) : Validation immédiate et sécurité des données.

\$mail = new PHPMailer(true); : Initialisation du moteur d'envoi d'emails.

\$mail->isSMTP(); : Utilisation du protocole SMTP standard.

\$mail->Host = env('MAIL_HOST'); : Configuration des identifiants via variables .env

\$mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS; : Connexion sécurisée TLS chiffrée.

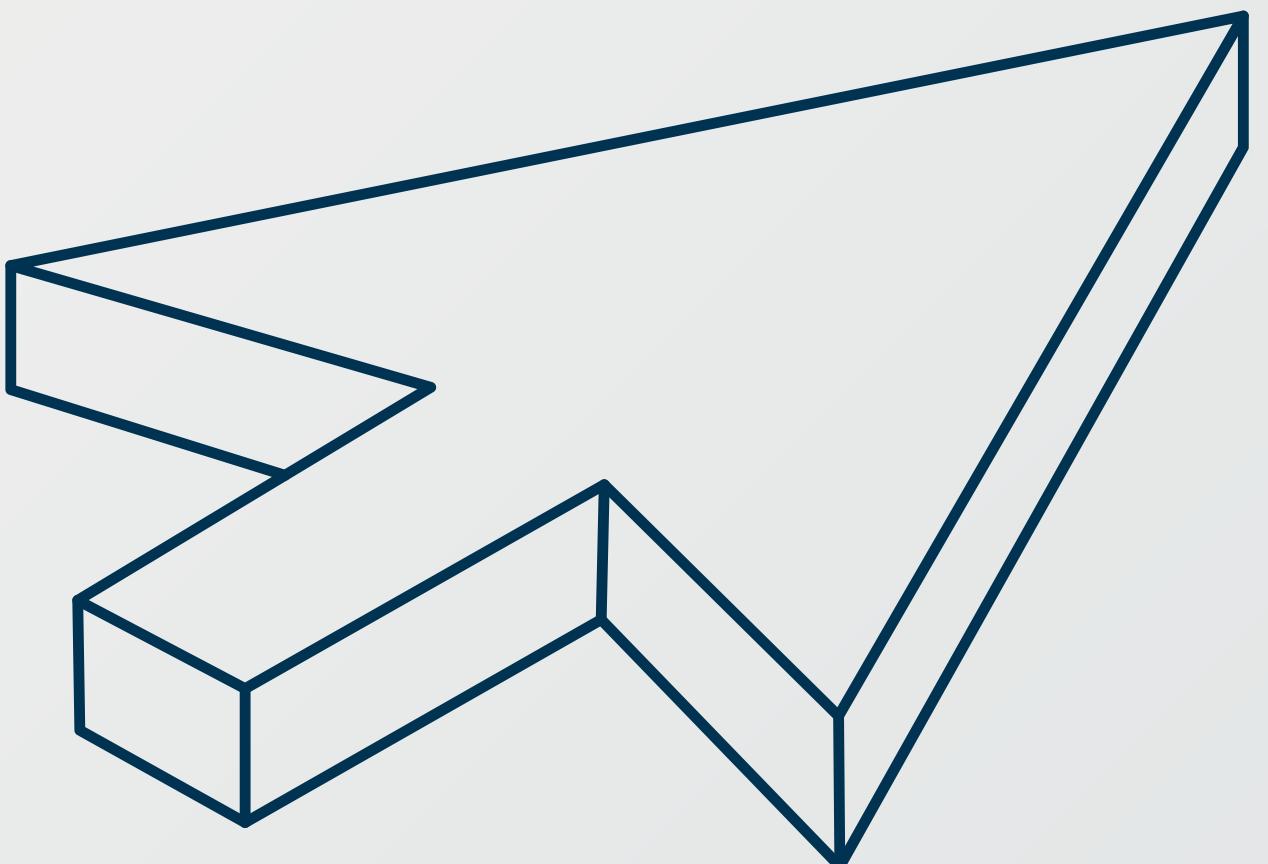
\$mail->addReplyTo(\$fromEmail, \$name); : Réponse directe à l'utilisateur facilitée.

\$mail->send(); : Exécution de l'envoi de l'email

try { ... } catch (Exception \$e) { ... } : Gestion robuste des erreurs d'envoi.

IV -

CONCLUSION



Bilan du Projet Portal_Job

Ce projet représente une étape significative dans la maîtrise des technologies full-stack, notamment avec le framework **Laravel** pour le **Backend**. Malgré les défis rencontrés, notamment l'optimisation des requêtes complexes, j'ai réussi à implémenter l'intégralité des **Fonctionnalités clés** pour une plateforme d'emploi fonctionnelle ou alors cela sera mis en place d'ici peu.

Le développement de **Portal_job** est la preuve concrète de ma persévérance, de ma capacité à structurer un projet complet et à appliquer les meilleures pratiques de développement.

Perspectives d'Évolution

Le projet **Portal_job** a un fort potentiel d'évolution. Les prochaines étapes incluent l'intégration d'un **système d'intelligence artificielle** pour des recommandations d'offres plus pertinentes et l'ajout d'une **messagerie interne** pour faciliter la communication directe entre candidats et recruteurs.

Je prévois également l'optimisation des performances du Frontend pour garantir un temps de chargement minimal.

Conclusion

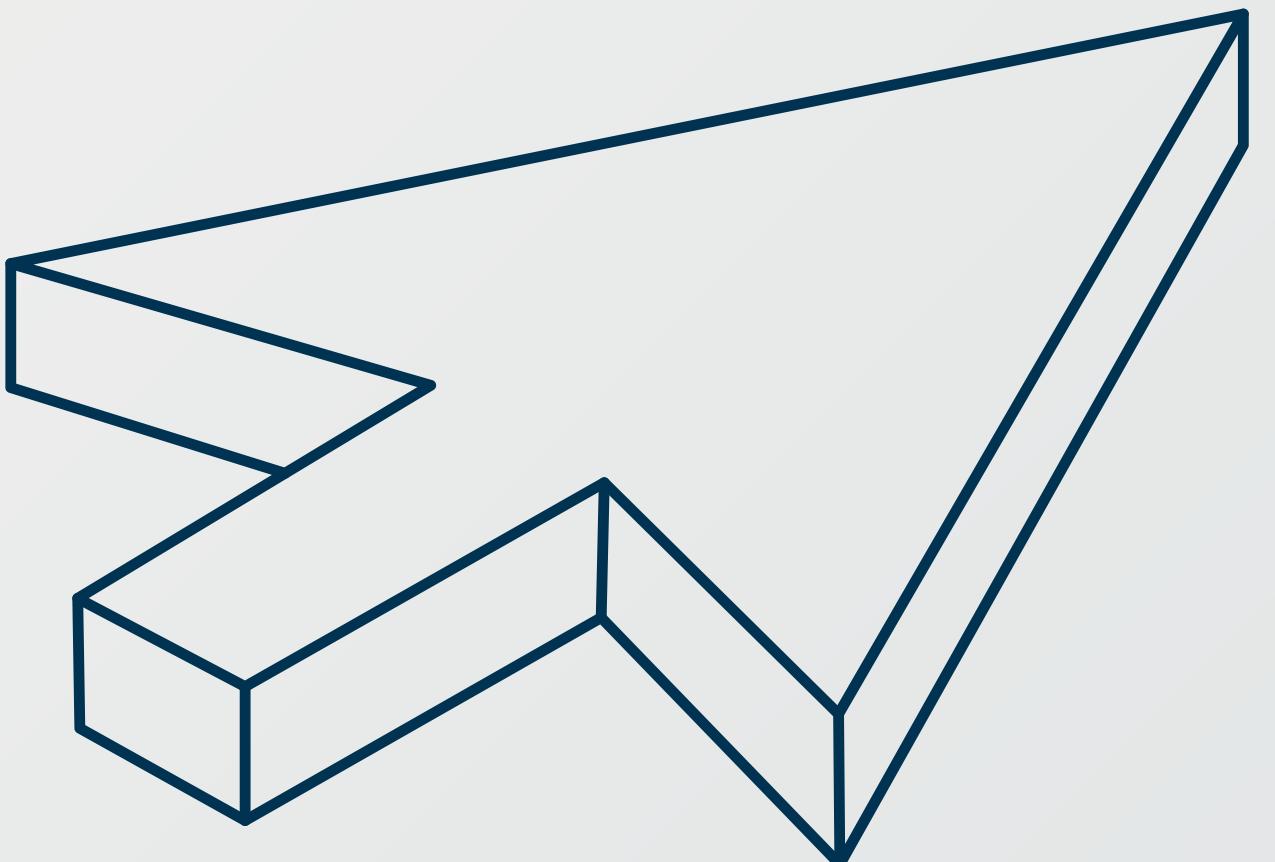
En conclusion ,

La refonte de [Portal_job](#) est un succès qui démontre ma capacité à concevoir et développer une plateforme complète, sécurisée et orientée utilisateur, du Frontend au Backend.

Ce projet m'a permis de solidifier mes compétences en Laravel, POO et gestion de base de données, et je suis prêt à appliquer cette expertise aux défis futurs du développement web.

V -

ANNEXES



Fichier : api.php (racine du projet back)

```
// Rôles : company, admin
Route::middleware(['auth:sanctum', 'role:company,admin'])->group(
    function () {
        // CORRECTION 1 : Remplacer Route::post par Route::get pour la lecture
        Route::get('/userById/{id}', [UserController::class, 'getUserById']);

        Route::post('/userUpdate/{id}', [UserController::class, 'updateUser']);

        Route::post('/addOffer', [OfferController::class, 'addOffer']);
        Route::post('/offerUpdate/{id}', [OfferController::class, 'updateOffer']);
        Route::delete('/deleteOffer/{id}', [OfferController::class, 'deleteOffer']);

        Route::post('/companyUpdate/{id}', [CompanyController::class, 'updateCompany']);
        Route::delete('/deleteCompany/{id}', [CompanyController::class, 'deleteCompany']);

        Route::get('/requestById/{id}', [RequestController::class, 'getRequestById']);
        Route::post('/requestUpdate/{id}', [RequestController::class, 'updateRequest']);
        Route::delete('/deleteRequest/{id}', [RequestController::class, 'deleteRequest']);
    }
);

// Rôles : candidat, admin
Route::middleware(['auth:sanctum', 'role:candidat,admin'])->group(
    function () {
        // CORRECTION 2 : Remplacer Route::post par Route::get pour la lecture
        Route::get('/userById/{id}', [UserController::class, 'getUserById']);

        Route::post('/userUpdate/{id}', [UserController::class, 'updateUser']);

        Route::get('/requestById/{id}', [RequestController::class, 'getRequestById']);
        Route::post('/requestUpdate/{id}', [RequestController::class, 'updateRequest']);
        Route::delete('/deleteRequest/{id}', [RequestController::class, 'deleteRequest']);

        Route::get('/favorisById/{id}', [FavorisController::class, 'getFavorisById']);
        Route::post('/addFavoris', [FavorisController::class, 'addFavoris']);
        Route::delete('/deleteFavoris/{id}', [FavorisController::class, 'deleteFavoris']);
    }
);
```

Import de mes controllers qui eux s'occupent du CRUD

```
<?php

use App\Http\Controllers\AuthController;
use App\Http\Controllers\OfferController;
use App\Http\Controllers\FavorisController;
use App\Http\Controllers\RequestController;
use App\Http\Controllers\UserController;
use App\Http\Controllers\CompanyController;
use Illuminate\Support\Facades\Route;
use Illuminate\Http\Request;
```

Les Routes qui s'occupent de la communication avec la bdd les controllers et donc dans l'url

Fichier : index.js (projet front)

```
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'homepage',
      component: HomeView,
    },
    {
      path: '/SignIn',
      name: 'connexion',
      component: Connexion,
    },
    {
      path: '/SignUp/addUser/',
      name: 'inscription',
      component: Inscription,
    },
    {
      path: '/SignIn/passwordForget',
      name: 'motDePasse',
      component: Password,
    },
    {
      path: '/SignIn/applyCompany',
      name: 'ajoutSociete',
      component: ApplyCompany,
    },
    {
      path: '/Home',
      name: 'Accueil',
      component: Accueil,
    },
    {
      path: '/Home/apply',
      name: "Postuler à l'offre",
      component: Apply,
    },
  ],
})
```

history : Définit la manière dont le routeur gère l'historique de navigation dans le navigateur, ici en utilisant le mode HTML5 History ([createWebHistory](#)).

path : Spécifie le **chemin URL exact** auquel cette route doit correspondre.

name : Fournit un **nom unique** pour cette route, permettant une navigation facile et programmatique sans avoir besoin de l'**URL complète**.

component : Indique la vue **Vue.js** (le composant) qui doit être chargée et rendue lorsque le **path** est activé.

Import de mes Vue qui eux s'occupent de faire appel à mon html css js



```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'
import Connexion from '../views/SignIn.vue'
import Inscription from '../views/SignUp.vue'
import Password from '../views/PasswordForget.vue'
import ApplyCompany from '../views/ApplyCompany.vue'
import Accueil from '../views/Home.vue'
import Apply from '../views/ApplyOffer.vue'
import Offers from '../views/Offers.vue'
import Companys from '../views/Companys.vue'
import CompanyDetails from '../views/CompanyDetails.vue'
import myRequest from '../views/MyRequest.vue'
import Request from '../views/Request.vue'
import DashboardCompany from '../views/DashboardCompany.vue'
import DashboardAdmin from '../views/DashboardAdmin.vue'
import Favoris from '../views/Favoris.vue'
import Profil from '../views/Profil.vue'
import Contact from '../views/Contact.vue'
import OffersCompany from '../views/OffersCompany.vue'
import UpdateMyRequest from '../views/UpdateMyRequest.vue'
import AddMyRequest from '../views/AddMyRequest.vue'
import UpdateOfferById from '../views/UpdateOffer.vue'
import UpdateCompanyByID from '../views/UpdateCompany.vue'
import UpdateProfil from '../views/UpdateProfil.vue'
```

