



Université Abdelmalek Essaâdi
Ecole Nationale des Sciences Appliquées de
Tanger
Année Universitaire 2023/2024



Projet d'Intégration des acquis de la première année Génie Informatique

GENIUS LAB

Mise en place d'une application web pour la gestion des Activités pour Enfants

RAPPORT TECHNIQUE

Prof Encadrant:
Prof GHAILANI Mohamed

Tuteur Encadrant :
ELHADDAD Mohamed
RAYES Abdelkoudous

:



Remerciements

Avant de décrire les parties importantes du projet, je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué de manière significative à l'élaboration et à la réalisation de ce travail. Je souhaite tout d'abord remercier chaleureusement l'ensemble du personnel d'ENSAT. Leur dévouement constant à nous préparer, mes collègues et moi, pour de telles expériences afin de nous intégrer pleinement dans le monde professionnel, mérite ma sincère reconnaissance.

Je tiens à remercier particulièrement mon professeur M. GHAILANI Mohamed pour son aide précieuse, ses vidéos et son support continu. Il est l'organisateur de ce projet et a travaillé énormément pour le succès de cette incroyable expérience professionnelle.

Mes remerciements vont également à mes tuteurs, M. Mohamed El Haddad et M. Rayes Abdelkoudous, pour leur soutien constant tout au long de la journée.



Avant-propos

Ce rapport présente notre projet d'intégration réalisé à l'ENSAT, intitulé "Genius Lab - Plateforme Web de Gestion des Activités pour Enfants". Notre équipe, composée de membres aux compétences variées, a travaillé en collaboration pour développer cette solution innovante destinée à faciliter la gestion des activités pour enfants.

Nom de l'équipe : Genius Lab

Nom	Rôle
Yassine El Garti	Chef de projet
Abdelouahab Soufiane	Équipe Front-End
Ahlam Ben Imran	Équipe Front-End
Achraf El Aidi El Idrissi	Équipe Back-End
Aymane Arib	Équipe Back-End
Salmane Koraichi	Équipe Tests et Sécurité
Fatimaezzahrae Taouil	Équipe Tests et Sécurité
Silie El Ghazi	Équipe système Linux -dockerisation
Azzman Norelimane	Équipe système Linux -dockerisation
Sarsri Aymane	Équipe BDD

Notre projet, Genius Lab, vise à développer une plateforme web permettant aux parents d'inscrire leurs enfants à diverses activités proposées par une entreprise spécialisée dans le coaching et le développement personnel pour enfants. Cette plateforme facilitera la gestion des inscriptions, des plannings et des paiements, offrant une solution efficace tant pour les parents que pour l'entreprise.

Mots clés : **Front-End, Back-End, Docker, Tests, Sécurité, BDD**

Introduction générale

Le projet Genius Lab - Plateforme Web de Gestion des Activités pour Enfants vise à simplifier l'inscription, la planification et la gestion des paiements des activités pour enfants. Cette plateforme ambitionne de fournir une solution intuitive et efficace pour les parents et les administrateurs.

Pour atteindre cet objectif, nous utilisons diverses technologies modernes. Le frontend est développé en utilisant Vue.js pour créer une interface utilisateur dynamique et réactive. Le back-end est construit avec Laravel, un framework PHP robuste, pour gérer les API RESTful et assurer une communication fluide avec la base de données. PostgreSQL est utilisé pour la gestion des données, garantissant des performances optimales et une sécurité renforcée. Docker est employé pour la conteneurisation, facilitant le déploiement et la scalabilité de l'application.

Ce rapport se concentrera principalement sur le développement de la partie front-end, incluant les tâches suivantes :

- Conception de l'interface : Créer des maquettes et des wireframes, puis développer l'interface avec Vue.js.
- Composants réutilisables : Développer des composants modulaires pour une meilleure efficacité.
- Intégration API-REST : Assurer une communication fluide entre le front-end et le back-end.

Ce rapport détaillera le processus de conception et de développement de la plateforme Genius Lab, en mettant en avant le contexte, les spécifications, les méthodologies et les technologies utilisées. L'objectif est d'améliorer la gestion des activités pour enfants et d'offrir une meilleure expérience utilisateur.

1. Partie Back-End

1.1. Environnement de Développement :

1.1.1. IDE (Environnements de Développement Intégrés) :

Visual Studio Code : Un éditeur de code source polyvalent et largement utilisé, offrant une multitude de extensions pour supporter les langages de programmation, le contrôle de version, le debugging.

PyCharm : Spécifiquement utilisé pour le développement en Python, pour le système de recommandation implémenté avec Flask.

1.1.2. Compilateurs et Interpréteurs :

PHP : Utilisé pour le backend avec Laravel. PHP est interprété directement par le serveur web (par exemple, Apache ou Nginx).

Python : Utilisé pour le système de recommandation avec Flask. Python est interprété par l'interpréteur Python, qui est configuré dans l'environnement de développement.

1.1.3. Gestion des Dépendances :

Composer : Utilisé pour gérer les dépendances PHP du projet Laravel.

1.1.4. Serveurs et Bases de Données :

XAMPP/Adminer: Environnements de serveurs locaux qui regroupent Apache, MySQL, et PHP, permettant le développement et les tests en local.

PostgreSQL/MySQL : Systèmes de gestion de bases de données relationnelles utilisés pour stocker les données de l'application.

1.1.5. Plateformes de Test et Debugging :

Postman : Utilisé pour tester les API et les endpoints, permettant de valider les requêtes et réponses.

1.2. Gestion des Contrôleurs

Dans cette section, nous allons détailler la gestion des contrôleurs utilisés dans notre projet. Les contrôleurs jouent un rôle essentiel en tant que point central pour gérer les requêtes HTTP, les opérations sur les données, et les réponses aux utilisateurs. Voici une liste des contrôleurs et leurs responsabilités dans notre projet de plateforme de gestion des activités pour enfants.

1.2.1. Liste des Contrôleurs et leurs Actions :

GET HEAD	/ ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
POST	_ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
GET HEAD	_ignition/health-check ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST	_ignition/update-config ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET HEAD	api/activites activites.index > Api\ActiviteController@index
POST	api/activites activites.store > Api\ActiviteController@store
GET HEAD	api/activites/{activite} activites.show > Api\ActiviteController@show
PUT PATCH	api/activites/{activite} activites.update > Api\ActiviteController@update
DELETE	api/activites/{activite} activites.destroy > Api\ActiviteController@destroy
GET HEAD	api/activites/{activite}/horaires activites.horaires.index > Api\ActiviteHoraireController@index
POST	api/activites/{activite}/horaires activites.horaires.store > Api\ActiviteHoraireController@store
GET HEAD	api/activites/{activite}/horaires/{horaire} activites.horaires.show > Api\ActiviteHoraireController@show
PUT PATCH	api/activites/{activite}/horaires/{horaire} activites.horaires.update > Api\ActiviteHoraireController@update
DELETE	api/activites/{activite}/horaires/{horaire} activites.horaires.destroy > Api\ActiviteHoraireController@destroy
POST	api/activites/{activite}/langues activites.langues.store > Api\ActiviteLangueController@store
GET HEAD	api/activites/{activite}/langues/{langue} activites.langues.show > Api\ActiviteLangueController@show
GET HEAD	api/activites/{activite}/modalites activites.modalites.index > Api\ActiviteModaliteController@index
POST	api/activites/{activite}/modalites activites.modalites.store > Api\ActiviteModaliteController@store
GET HEAD	api/activites/{activite}/modalites/{modalite} activites.modalites.show > Api\ActiviteModaliteController@show
PUT PATCH	api/activites/{activite}/modalites/{modalite} activites.modalites.update > Api\ActiviteModaliteController@update
DELETE	api/activites/{activite}/modalites/{modalite} activites.modalites.destroy > Api\ActiviteModaliteController@destroy
GET HEAD	api/activites/{activite}/objectifs activites.objectifs.index > Api\ObjectifController@index
POST	api/activites/{activite}/objectifs activites.objectifs.store > Api\ObjectifController@store
GET HEAD	api/activites/{activite}/objectifs/{objectif} activites.objectifs.show > Api\ObjectifController@show
PUT PATCH	api/activites/{activite}/objectifs/{objectif} activites.objectifs.update > Api\ObjectifController@update
DELETE	api/activites/{activite}/objectifs/{objectif} activites.objectifs.destroy > Api\ObjectifController@destroy
GET HEAD	api/activites/{id}/image Api\ImageUploadController@getActiviteImage
POST	api/activites/{id}/upload-image Api\ImageUploadController@uploadActiviteImage
GET HEAD	api/admin/dashboard admin.dashboard > Api\UserController@index
GET HEAD	api/admin/enable-2fa admin.enable2fa > Api\Admin2FAController@enable2FA
POST	api/admin/verify-2fa admin.verify2fa > Api\Admin2FAController@verify2FA
GET HEAD	api/animateurs animateurs.index > Api\AnimateurController@index
POST	api/animateurs animateurs.store > Api\AnimateurController@store
GET HEAD	api/animateurs/{animateur} animateurs.show > Api\AnimateurController@show
PUT PATCH	api/animateurs/{animateur} animateurs.update > Api\AnimateurController@update
DELETE	api/animateurs/{animateur} animateurs.destroy > Api\AnimateurController@destroy
GET HEAD	api/animateurs/{animateur}/dispo animateurs.dispo.index > Api\DispoAnimateurController@index

Les contrôleurs de l'API sont essentiels pour la plateforme de gestion des activités pour enfants, couvrant divers aspects fonctionnels. Parmi eux, on trouve des contrôleurs pour la gestion des activités (comme la création, la mise à jour et la suppression), la gestion des utilisateurs et des administrateurs, la gestion des services et des catégories, ainsi que la gestion des messages et des notifications. Chaque contrôleur assure des opérations spécifiques telles que la gestion des horaires et des langues associées aux activités, la gestion des disponibilités des animateurs et des enfants, ainsi que la gestion des devis et des factures. Ces contrôleurs permettent une gestion efficace et sécurisée des données et des interactions au sein de la plateforme, assurant une expérience utilisateur optimale.

1.2.2. Exemple et manipulation :

Dans cette section, nous présentons deux exemples de contrôleurs API utilisés dans notre projet de plateforme de gestion des activités pour enfants. Les contrôleurs jouent un rôle central dans la gestion des requêtes HTTP, des opérations sur les données et des réponses aux utilisateurs. Chaque exemple est détaillé ci-dessous avec des informations spécifiques sur les méthodes correspondantes et leurs responsabilités respectives.

❖ Exemple : Gestion de l'authentification à deux facteurs pour les administrateurs :

Méthode HTTP	Chemin d'accès	Contrôleur	Méthode	Description
GET / HEAD	api/admin/enable-2fa	Api\Admin2FAController	enable2FA	Activer l'authentification à deux facteurs pour un admin
POST	api/admin/verify-2fa	Api\Admin2FAController	verify2FA	Vérifier le code d'authentification à deux facteurs pour un admin

1.3. Les migrations :

Les migrations dans Laravel sont des scripts PHP utilisés pour gérer la structure de la base de données de manière incrémentielle et reproductible. Elles permettent de créer et de modifier les tables sans avoir à manipuler directement SQL, ce qui facilite la gestion de la base de données au sein de l'application.

Chaque migration correspond généralement à une modification de la base de données, comme la création d'une nouvelle table, l'ajout de colonnes, ou la modification de contraintes. Elles sont utilisées pour maintenir la cohérence de la

base de données tout au long du développement et du déploiement de l'application.

1.3.1. Création et Exécution :

Les migrations sont créées à l'aide de la commande : `php artisan make:migration`

Une fois écrites, elles peuvent être exécutées avec, `php artisan migrate`

Revenir en arrière sur les modifications du BD `php artisan migrate:rollback`

1.3.2. Exemple d'utilisation :

Pour voir toutes les migrations crée dans nos projets : `php artisan migrate:status`

```
Migration name ..... Batch / Status
2014_10_12_000000_create_users_table ..... [2] Ran
2014_10_12_100000_create_password_reset_tokens_table ..... [2] Ran
2019_08_19_000000_create_failed_jobs_table ..... [2] Ran
2019_12_14_000001_create_personal_access_tokens_table ..... [2] Ran
2024_05_01_155136_create_animateurs_table ..... [2] Ran
2024_05_01_155334_create_parentals_table ..... [2] Ran
2024_05_01_155417_create_administrateurs_table ..... [2] Ran
2024_05_01_155443_create_enfants_table ..... [2] Ran
2024_05_01_155458_create_horaires_table ..... [2] Ran
2024_05_01_155503_create_activites_table ..... [2] Ran
2024_05_01_155522_create_dispo_animateurs_table ..... [2] Ran
2024_05_01_155625_create_animers_table ..... [2] Ran
2024_05_01_155733_create_offres_table ..... [2] Ran
2024_05_01_155746_create_participers_table ..... [2] Ran
2024_05_01_155826_create_dispo_enfants_table ..... [2] Ran
2024_05_01_170411_create_contient_demandes_table ..... [2] Ran
2024_05_01_170452_create_corresponds_table ..... [2] Ran
2024_05_01_170538_create_packs_table ..... [2] Ran
2024_05_01_170559_create_factures_table ..... [2] Ran
2024_05_01_170605_create_devis_table ..... [2] Ran
2024_05_01_170846_create_activite_horaires_table ..... [2] Ran
2024_05_02_092006_create_experiences_table ..... [2] Ran
2024_05_02_092033_create_langues_table ..... [2] Ran
2024_05_02_092041_create_categories_table ..... [2] Ran
2024_05_02_093159_create_services_table ..... [2] Ran
2024_05_02_093334_create_langue_animateurs_table ..... [2] Ran
2024_05_02_103532_create_animateur_categorie_services_table ..... [2] Ran
2024_05_04_133411_create_enfant_interets_table ..... [2] Ran
2024_05_06_151914_create_reviews_table ..... [2] Ran
2024_05_06_151949_create_objectifs_table ..... [2] Ran
2024_05_07_153202_create_activite_langues_table ..... [2] Ran
2024_05_16_171258_create_offre_activites_table ..... [2] Ran
2024_05_17_005907_create_modalites_table ..... [2] Ran
2024_05_17_111155_create_activite_modalites_table ..... [2] Ran
2024_05_18_004823_create_demandes_table ..... [2] Ran
2024_05_18_170543_create_demande_activite_enfants_table ..... [2] Ran
2024_06_10_200544_create_conversations_table ..... [2] Ran
```


❖ Exemple 1 : Migration pour la table `recommendation_forms` :

La migration suivante illustre la création de la table `recommendation_forms` dans la base de données, utilisée pour stocker les préférences de recommandation associées aux utilisateurs et aux enfants.

```
<?php

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

class CreateRecommendationFormsTable extends Migration
{
    public function up()
    {
        Schema::create('recommendation_forms', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->onDelete('cascade');
            $table->foreignId('enfant_id')->constrained()->onDelete('cascade');
            $table->text('preferences');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('recommendation_forms');
    }
}
```

Objectif : Cette migration vise à créer la structure de la table `recommendation_forms` pour gérer les informations de recommandation dans l'application.

Méthode `up()` : Cette méthode est exécutée lors de l'application de la migration. Elle utilise Laravel Schéma Builder (`Schema::create`) pour définir la structure de la table `recommendation_forms` avec toutes ses colonnes et configurations associées.

Méthode `down()` : Cette méthode est utilisée pour annuler la migration. Si nécessaire, elle supprime la table `recommendation_forms` en utilisant `Schema::dropIfExists`, ce qui permet de revenir à l'état précédent avant l'application de cette migration.

Colonnes :

id : Clé primaire auto-incrémentée qui identifie de manière unique chaque enregistrement dans la table.

user_id : Clé étrangère référençant l'identifiant de l'utilisateur associé à la recommandation. La méthode `constrained()` assure que cette clé étrangère est liée à la table `users`, et `onDelete('cascade')` spécifie que si l'utilisateur est supprimé, toutes les recommandations associées seront également supprimées pour maintenir l'intégrité des données.

enfant_id : Clé étrangère référençant l'identifiant de l'enfant associé à la recommandation, avec une configuration similaire à `user_id`.

preferences : Champ de type texte qui stocke les préférences spécifiques à la recommandation.

timestamps() : Méthode qui ajoute automatiquement deux colonnes `created_at` et `updated_at`. Ces colonnes enregistrent les timestamps de création et de dernière mise à jour de chaque enregistrement, ce qui est essentiel pour suivre l'historique des modifications.

1.4. Gestion de Sanctum et Middleware :

1.4.1. Les Sanctum :

❖ Définition générale :

Sanctum est un package Laravel qui fournit une solution légère pour l'authentification des SPA (Single Page Applications), des applications mobiles et des API simples en utilisant des tokens. Sanctum permet de gérer facilement les tokens d'authentification et de sécuriser les APIs.

❖ Son rôle :

Sanctum permet aux utilisateurs de générer des tokens d'API personnels qui peuvent être utilisés pour authentifier les requêtes API. Il fournit également une authentification sans état pour les API en utilisant les tokens, ce qui est idéal pour

les applications SPA et les applications mobiles. De plus, Sanctum peut gérer les sessions de l'utilisateur pour les applications traditionnelles basées sur les sessions.

❖ Usage dans le projet (exemple) :

Dans notre projet, Sanctum est utilisé pour sécuriser les API qui nécessitent une authentification. Voici quelques exemples d'implémentation :

Authentification de l'utilisateur :

```
Route::middleware('auth:sanctum')->get('/user', function (Request $request) {  
    |     return $request->user();  
    | });
```

Cette route utilise le middleware `auth:sanctum` pour s'assurer que seules les requêtes authentifiées peuvent accéder à l'information de l'utilisateur. Lorsqu'un utilisateur authentifié envoie une requête, les détails de son profil utilisateur sont renvoyés en réponse.

Gestion des recommandations :

```
//Systeme de recomendation  
Route::middleware('auth:sanctum')->group(function () {  
    |     Route::post('/recommendations/form', [RecommendationController::class, 'store']);  
    |     Route::get('/recommendations/{enfant}', [RecommendationController::class, 'recommend']);  
    | });
```

Ces routes gèrent les recommandations. Le middleware `auth:sanctum` s'assure que seules les requêtes provenant d'utilisateurs authentifiés peuvent envoyer des recommandations ou récupérer des recommandations pour un enfant spécifique. Par exemple, un parent authentifié peut envoyer des recommandations en remplissant un formulaire, et les recommandations peuvent être récupérées pour un enfant donné.

Gestion des notifications :

```
//Systeme de notification  
Route::middleware('auth:sanctum')->group(function () {  
    |     Route::get('/notifications', [NotificationController::class, 'index']);  
    |     Route::post('/notifications/{id}/read', [NotificationController::class, 'markAsRead']);  
    |     Route::post('/send-notification', [NotificationController::class, 'sendNotification']);  
    | });
```

Ces routes gèrent le système de notifications. Le middleware `auth:sanctum` s'assure que seules les requêtes provenant d'utilisateurs authentifiés peuvent accéder aux notifications, les marquer comme lues ou envoyer de nouvelles

notifications. Cela garantit que chaque utilisateur ne peut accéder qu'à ses propres notifications.

1.4.2. Les Middleware:

❖ Définition générale :

Les middlewares sont des composants logiciels qui filtrent les requêtes HTTP entrant dans votre application. Ils permettent de traiter les requêtes avant qu'elles n'atteignent le contrôleur et de manipuler les réponses avant qu'elles ne soient renvoyées au client.

❖ Son rôle :

Les middlewares jouent un rôle crucial dans la sécurisation et la gestion des requêtes. Ils permettent de s'assurer que seules les requêtes authentifiées et autorisées accèdent aux ressources de l'application. Les middlewares peuvent également modifier les requêtes et les réponses.

❖ Usage dans le projet (exemple) :

Dans notre projet, plusieurs middlewares sont utilisés pour sécuriser les routes et gérer les permissions :

Authentification :

```
Route::middleware('auth:sanctum')->group(function () {  
    Route::get('/user', function (Request $request) {  
        return $request->user();  
    });  
});
```

Le middleware auth:api protège les routes liées à l'activation et la vérification de la double authentification (2FA) pour les administrateurs. Cela garantit que seules les requêtes provenant d'administrateurs authentifiés peuvent accéder à ces fonctionnalités.

2fa:

```
Route::middleware(['auth:api', '2fa'])->group(function () {  
    | Route::get('/admin/dashboard', [UserController::class, 'index'])->name('admin.dashboard');  
});
```

Le middleware `2fa` est utilisé en conjonction avec `auth:api` pour protéger la route du tableau de bord administratif. Après la vérification de l'authentification via un token API, le middleware `2fa` s'assure que la deuxième étape de l'authentification (2FA) est également complétée avant d'accorder l'accès au tableau de bord. un formulaire, et les recommandations peuvent être récupérées pour un enfant donné.

1.4.3. Les fonctionnalités ajoutées :

Dans cette section, nous détaillons les différentes fonctionnalités qui ont été ajoutées à notre application web. Chaque fonctionnalité est conçue pour améliorer l'interaction utilisateur, la sécurité et l'efficacité globale de l'application.

❖ Chat

Nous avons implémenté un système de chat pour faciliter la communication entre les utilisateurs. Les fonctionnalités de chat incluent :

Afficher les conversations :

```
Route::get('/conversations', [ConversationsController::class, 'index']);
```

Description : Cette route permet de récupérer la liste de toutes les conversations disponibles pour l'utilisateur connecté.

Afficher une conversation spécifique

```
Route::get('/conversations/{conversation}', [ConversationsController::class, 'show']);
```

Description : Cette route permet de récupérer les détails d'une conversation spécifique.

Créer une nouvelle conversation :

```
Route::post('/conversations', [ConversationsController::class, 'store']);
```

Description : Cette route permet de créer une nouvelle conversation

Afficher les messages d'une conversation :

```
Route::get('/conversations/{conversation}/messages', [MessagesController::class, 'index']);
```

Description : Cette route permet de récupérer la liste des messages pour une conversation donnée.

Envoyer un message dans une conversation :

```
Route::post('/conversations/{conversation}/messages', [MessagesController::class, 'store']);
```

Description : Cette route permet d'envoyer un nouveau message dans une conversation.

❖ Double authentification

Pour renforcer la sécurité des comptes administrateurs, nous avons implémenté un système de double authentification (2FA). Les fonctionnalités incluent :

```
//Double authentification
Route::middleware(['auth:api'])->group(function () {
    Route::get('/admin/enable-2fa', [Admin2FAController::class, 'enable2FA'])->name('admin.enable2fa');
    Route::post('/admin/verify-2fa', [Admin2FAController::class, 'verify2FA'])->name('admin.verify2fa');
});
Route::middleware(['auth:api', '2fa'])->group(function () {
    Route::get('/admin/dashboard', [UserController::class, 'index'])->name('admin.dashboard');
});
```

Activer la double authentification :

Description : Cette route permet d'activer la double authentification pour un administrateur.

Vérifier la double authentification:

Description : Cette route permet de vérifier le code de double authentification envoyé à l'administrateur.

❖ Système de recommandation :

Pour améliorer l'expérience utilisateur, nous avons ajouté un système de recommandations personnalisées

```
//Systeme de recommandation
Route::middleware('auth:sanctum')->group(function () {
    Route::post('/recommendations/form', [RecommendationController::class, 'store']);
    Route::get('/recommendations/{enfant}', [RecommendationController::class, 'recommend']);
});
```

Soumettre une recommandation :

Description : Cette route permet de soumettre une recommandation via un formulaire, accessible uniquement aux utilisateurs authentifiés.

Récupérer les recommandations pour un enfant :

Description : Cette route permet de récupérer les recommandations pour un enfant spécifique, accessible uniquement aux utilisateurs authentifiés.

❖ Gestion des devis et factures:

Pour faciliter la gestion administrative, nous avons ajouté des fonctionnalités pour la génération et le téléchargement des devis et factures :

```
//Upload facture et devis
Route::post('/generate-devis', [DevisController::class, 'generateDevis']);
Route::get('/download-devis/{id}', [DevisController::class, 'downloadDevis']);
Route::post('/generate-facture', [FactureController::class, 'generateFacture']);
Route::get('/download-facture/{id}', [FactureController::class, 'downloadFacture']);
```

Générer un devis et une facture :

Description : Ces deux routes permet de générer les nouveau devis et factures

Télécharger un devis et une facture:

Description : Ces deux routes permet de télécharger les nouveau devis et factures

❖ Gestion des mots de passe:

Pour améliorer la gestion des mots de passe, nous avons ajouté des fonctionnalités pour la réinitialisation et la récupération des mots de passe :

```
//Forgot et resset password
Route::post('/forgot-password', [AuthController::class, 'forgotPassword']);
Route::post('/reset-password', [AuthController::class, 'resetPassword'])->name('password.reset');
Route::post('/session/reset-token', [AuthController::class, 'storeTokenInSession']);

//Verfier email
Route::get('/password/find/{token}', [AuthController::class, 'findToken']);
Route::get('/verify-email/{id}', [UserController::class, 'verifyEmail'])->name('verification.verify');
Route::get('/verify-email/{token}', [UserController::class, 'verifyEmail'])->name('verification.verify');
```

Mot de passe oublié:

Description : Cette route permet d'envoyer un email de réinitialisation de mot de passe.

Réinitialiser le mot de passe:

Description : Cette route permet de réinitialiser le mot de passe via un lien reçu par email.

Vérification des emails:

Description : Cette route permet de stocker le token de réinitialisation dans la session utilisateur.

❖ Connexion avec les réseaux sociaux

Pour simplifier l'authentification, nous avons ajouté la possibilité de se connecter via les réseaux sociaux comme Facebook, Instagram et Google :

```
//login with fb/insta/google
Route::middleware(['web'])->group(function () {
    Route::get('login/{provider}', [AuthController::class, 'redirectToProvider']);
    Route::get('login/{provider}/callback', [AuthController::class, 'handleProviderCallback']);
});

Route::post('/login/cancel-authorization', [AuthController::class, 'cancelAuthorization']);
Route::post('/login/data-deletion', [AuthController::class, 'dataDeletion']);
```

Redirection vers le fournisseur de réseau social :

Description : Cette route redirige l'utilisateur vers le fournisseur de réseau sociale.

Gérer le callback du fournisseur de réseau social:

Description : Cette route gère le retour du fournisseur de réseau social après l'authentification.

❖ Système de notifications

Pour garder les utilisateurs informés, nous avons ajouté un système de notifications :

```
//Système de notification
Route::middleware('auth:sanctum')->group(function () {
    Route::get('/notifications', [NotificationController::class, 'index']);
    Route::post('/notifications/{id}/read', [NotificationController::class, 'markAsRead']);
    Route::post('/send-notification', [NotificationController::class, 'sendNotification']);
});
```

Afficher les notifications :

Description : Cette route permet de récupérer la liste des notifications pour l'utilisateur authentifié.

Marquer une notification comme lue :

Description : Cette route permet de marquer une notification spécifique comme lue.

Envoyer une notification :

Description : Cette route permet d'envoyer une nouvelle notification.

❖ La fonctionnalité de cache

Définition du cache

Le cache est une technologie qui permet de stocker temporairement des données pour accélérer leur accès futur. En conservant une copie des données fréquemment utilisées dans une mémoire à accès rapide, le cache réduit la nécessité de récupérer les mêmes données à partir de leur source d'origine (qui peut être plus lente, comme une base de données ou un serveur distant). Cela améliore les performances de l'application et réduit la charge sur les ressources d'arrière-plan.

Utilisation du cache dans le projet

Dans notre projet, nous avons utilisé le cache pour améliorer les performances et l'efficacité de plusieurs fonctionnalités clés. Voici quelques exemples concrets de son utilisation :

Récupération des conversations :

Pour accélérer l'accès aux conversations, nous avons mis en cache les résultats des requêtes de récupération des conversations

Description : Cette route utilise le cache pour stocker la liste des conversations pendant une heure, ce qui permet une récupération rapide des données lors des requêtes ultérieures.

Recommandations pour les enfants :

Les recommandations générées pour les enfants sont également mises en cache pour améliorer les performances de récupération.

Description : Cette route met en cache les recommandations pendant une heure, ce qui permet de fournir rapidement les recommandations aux utilisateurs sans interroger constamment la base de données.

Avantages observés

L'utilisation du cache dans ces parties du projet a permis de constater les avantages suivants :

- **Temps de réponse réduit :** Les utilisateurs constatent des temps de chargement plus rapides pour les pages de conversations, de recommandations et de profils.
- **Charge réduite sur le serveur :** Le nombre de requêtes envoyées à la base de données a diminué, ce qui a réduit la charge sur le serveur et amélioré la scalabilité de l'application.
- **Expérience utilisateur améliorée :** Les utilisateurs bénéficient d'une application plus réactive et fluide, ce qui améliore leur satisfaction et leur engagement.

2. Tests et Validation avec Postman

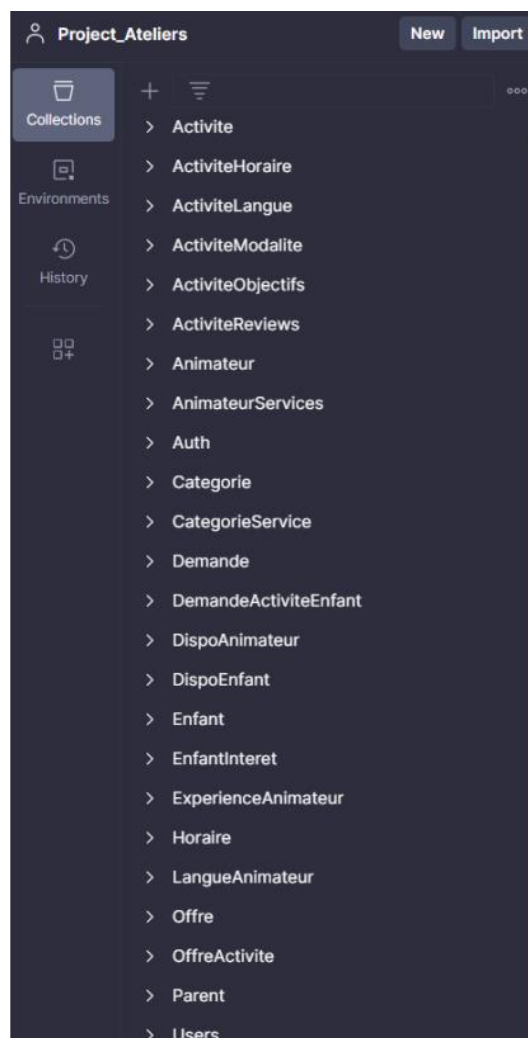
Pour valider les fonctionnalités de l'API développée, Postman est utilisé pour tester les différentes requêtes CRUD. Voici les étapes détaillées pour organiser et effectuer ces tests :

2.1. Organisation des Requêtes dans Postman :

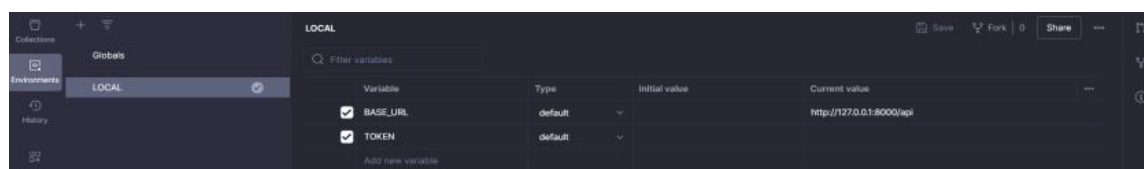
2.1.1. Collections CRUD :

Les requêtes CRUD sont organisées en collections dans Postman

2.1.2. Définition des Variables Locales :



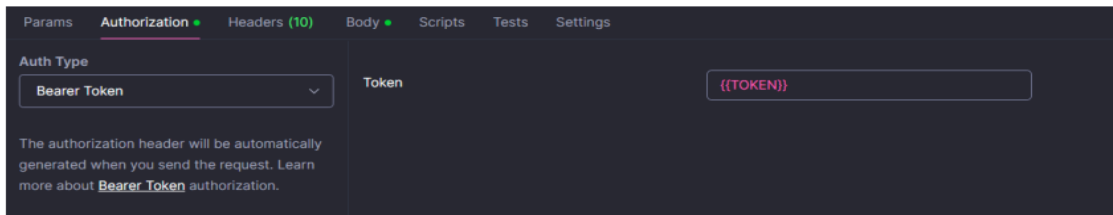
Pour faciliter les tests et éviter la répétition des mêmes valeurs, des variables locales sont définies dans l'environnement Postman. Les variables typiques incluent : `base_url` : URL de base de l'API. `Token` : Jeton d'authentification obtenu après la connexion.



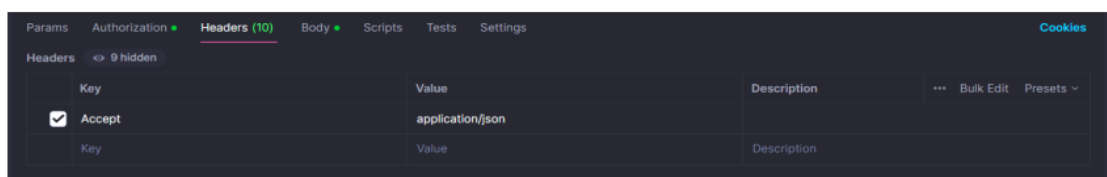
2.1.3. Exemples et Manipulation :

❖ **Effectuer une requête POST et créer une activité :**

Dans l'onglet "Authorization", sélectionnez "Bearer Token" et utilisez la variable token pour insérer le jeton de connexion.



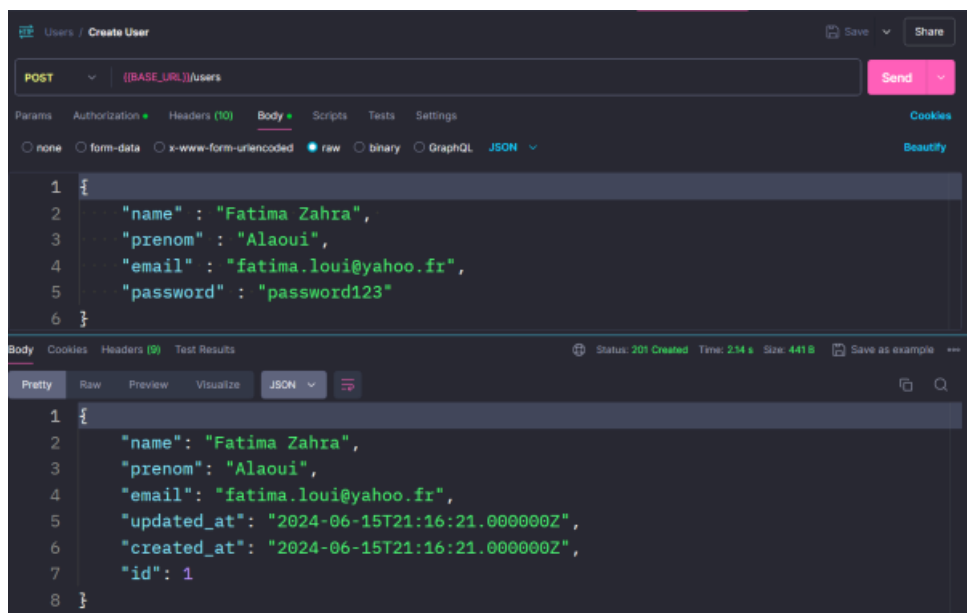
Dans l'onglet "Headers", ajoutez un en-tête pour s'assurer que le retour de la requête est au format JSON



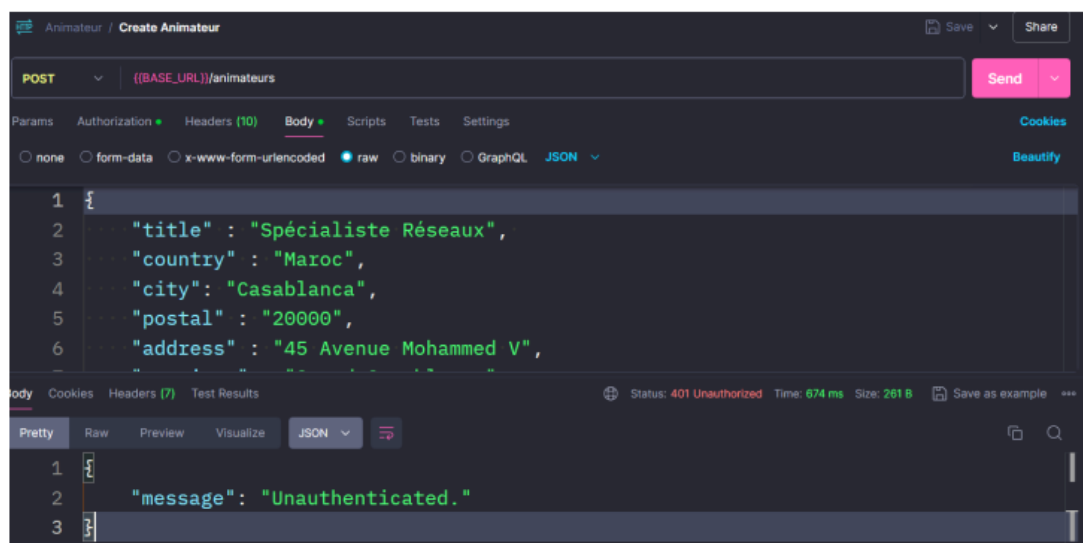
Dans l'onglet "Body", sélectionnez "raw" et choisissez "JSON" dans le menu déroulant. Ajoutez les données du formulaire en format JSON.



Envoyer la Requête (Clique sur send)



On aura cette réponse si l'animateur n'est pas authentifié



3. Partie Front-End :

Après l'achèvement de toutes les étapes de développement, nous présentons l'intégralité des pages et fonctionnalités réalisées. Cette section permet de visualiser l'ensemble du projet dans son état final.

3.2 Présentation des Pages et Fonctionnalités

3.2.1. Page d'Accueil

❖ Bannière de Bienvenue

- Une bannière visuellement attrayante présentant des images d'enfants heureux engagés dans différentes activités.
- Inclure un slogan accrocheur comme "Donner du Pouvoir aux Jeunes Esprits à travers un Apprentissage Amusant !

❖ Barre de Navigation

- Navigation claire et intuitive avec des menus déroulants pour un accès facile aux différentes sections telles que Ateliers, Laboratoires, Jeux d'Échecs, Inscription/Connexion, À Propos de Nous et Contact.

❖ Section en Vedette

- Mettre en avant les prochains ateliers ou offres spéciales avec des images de haute qualité, de brèves descriptions et des boutons d'appel à l'action proéminents.

❖ Section Témoignages

- Inclure des témoignages de parents ou d'élèves satisfaits, avec leurs photos et leurs retours sur l'impact positif des activités.

Section Ateliers

3.2.2. Autres Pages

❖ Listes Classées

- Diviser les ateliers en catégories telles que STEM (Sciences, Technologies, Ingénierie, Mathématiques), Arts Créatifs, Sports, etc.

- Chaque catégorie devrait avoir une bannière visuellement attrayante et une brève description.

❖ Cartes d'Ateliers

— Chaque carte d'atelier devrait afficher le titre de l'atelier, les objectifs, la tranche d'âge adaptée, les horaires disponibles (avec des options pour plusieurs sessions), les niveaux de prix (mensuel, trimestriel, etc.) et un bouton "Inscrivez-vous Maintenant".

❖ Filtres et Recherche

Mettre en place des filtres par tranche d'âge, horaire, prix et popularité pour aider les parents à affiner leurs choix.

— Inclure une barre de recherche pour un accès rapide aux ateliers spécifiques.

3.2.3. Inscription et Connexion

❖ Formulaire d'Inscription

— Créer un formulaire convivial avec des champs pour le nom du parent, l'email, le numéro de téléphone, l'adresse, le nom de l'enfant, l'âge, l'école et toutes les exigences particulières.

— Ajouter un captcha pour la sécurité.

❖ Page de Connexion

— Fonctionnalité de connexion sécurisée avec des options pour les parents et les administrateurs.

— Inclure des fonctionnalités "Mot de Passe Oublié" et "Se Souvenir de Moi" pour plus de commodité.

❖ Tableau de Bord Parent

— Après la connexion, les parents devraient arriver sur un tableau de bord personnalisé montrant leurs enfants inscrits, leurs sélections d'ateliers, le statut des paiements, les événements à venir et les notifications.

3.2.4. Page de Détails d'Atelier

❖ Informations Détaillées

— Inclure une description complète des objectifs de l'atelier, du programme, des matériaux fournis, du profil de l'instructeur et des témoignages des participants précédents.

❖ Horaire et Prix

— Afficher les horaires disponibles, les options de prix (avec des réductions pour plusieurs inscriptions), et les plans de paiement (mensuel, trimestriel, etc.). Ajouter un bouton "Ajouter au Panier" ou "Inscrivez-vous Maintenant".

❖ Avis et Évaluations

— Permettre aux parents de laisser des avis et des évaluations pour les ateliers auxquels ils ont participé.

— Afficher les notes globales et les retours pour aider les autres à prendre des décisions éclairées.

3.2.5. Panier et Paiement

❖ Vue d'Ensemble du Panier

— Montrer un résumé des ateliers sélectionnés dans le panier, avec des options pour supprimer des éléments, appliquer des réductions ou des codes promo, et passer à la caisse.

❖ Passerelle de Paiement Sécurisée

— Intégrer une passerelle de paiement fiable avec des options pour les cartes de crédit/débit, les portefeuilles numériques et les virements bancaires.

— Assurer le chiffrement SSL pour des transactions sécurisées.

❖ Confirmation

— Fournir une page de confirmation après un paiement réussi, avec un reçu imprimable et une confirmation par email

3.2.6. Panneau Administrateur

❖ Tableau de Bord

— Un tableau de bord centralisé pour les administrateurs avec des analyses en temps réel, des journaux d'activité utilisateur, des inscriptions en attente, et des rapports financiers.

❖ Gestion des Ateliers

— Outils pour ajouter/éditer des ateliers, définir des horaires, mettre à jour les prix, gérer les instructeurs, et suivre la popularité des ateliers.

❖ Gestion des Utilisateurs

— Capacité à gérer les comptes parentaux, voir les inscriptions, envoyer des notifications, et gérer les demandes de support client.

3.2.7. Design Réactif

❖ Optimisation Mobile

— Optimiser le site web pour les appareils mobiles avec des éléments de design réactif, des boutons adaptés au tactile, et des menus de navigation faciles.

❖ Compatibilité Multi-Navigateur

— Tester le site web sur plusieurs navigateurs (Chrome, Firefox, Safari, Edge) pour assurer des performances et une mise en page cohérentes.

3.3. Palette de Couleurs

3.3.1. Couleur Principale

Cette couleur est apaisante et souvent associée à la confiance, ce qui est important pour les parents envisageant d'inscrire leurs enfants à des activités.

3.3.2. Couleurs Secondaires

Fournit un fond doux et neutre, créant une atmosphère chaleureuse et accueillante pour le contenu.

Symbolise la créativité et l'imagination, adapté pour stimuler l'expression artistique ou l'innovation.

Évoque la tranquillité et la clarté, idéal pour promouvoir la pleine conscience ou les activités paisibles.

3.3.3. Couleurs Neutres

Utilisé pour le texte et les éléments essentiels, assurant la lisibilité et la hiérarchie visuelle.

Fournit un contraste pour le texte et les éléments essentiels, améliorant la lisibilité et la hiérarchie visuelle.

3.4. Outils & technologies

3.4.1. Vue.js

Vue.js est un Framework JavaScript progressif utilisé pour créer des interfaces utilisateur. Il permet de développer des applications web dynamiques et réactives avec une courbe d'apprentissage douce. Vue.js est particulièrement apprécié pour sa flexibilité et son intégration facile avec d'autres bibliothèques et projets existants.

Installation : npm install vue

3.4.2. Axios

Axios est une bibliothèque JavaScript utilisée pour faire des requêtes HTTP depuis le navigateur. Elle est largement utilisée pour interagir avec les API et récupérer ou envoyer des données. Axios offre une API simple et intuitive, facilitant l'intégration des appels API dans les applications Vue.js.

Installation : npm install axios

3.4.3. Vuex

Vuex est un gestionnaire d'état pour les applications Vue.js. Il centralise l'état des composants, permettant une gestion efficace et prévisible des données partagées. Vuex est essentiel pour les applications de grande taille où la gestion de l'état peut devenir complexe.

Installation : npm install Vuex

3.4.4. Tailwind CSS

Tailwind CSS est un Framework CSS utilitaire qui permet de créer des interfaces utilisateur personnalisées rapidement et efficacement. Il fournit une collection de classes utilitaires qui peuvent être combinées pour créer des designs sans écrire de CSS personnalisé. Tailwind CSS est particulièrement apprécié pour sa flexibilité et sa capacité à réduire le temps de développement.

Installation : npm install tailwindcss

3.4.5. Mapbox

Mapbox est une plateforme de cartographie utilisée pour créer et intégrer des cartes interactives dans les applications web et mobiles. Elle offre des outils puissants pour la visualisation de données géo spatiales et est utilisée dans diverses applications nécessitant des cartes personnalisables.

Installation : `npm install mapboxgl`

4. Test côté Backend:

5.1. Qu'est-ce que les tests ?

Les tests et l'assurance qualité (QA) sont des processus essentiels dans le développement logiciel visant à garantir que les applications fonctionnent comme prévu et répondent aux attentes des utilisateurs. Les tests consistent à examiner les différentes fonctionnalités du logiciel pour identifier et corriger les erreurs éventuelles.

L'assurance qualité, quant à elle, est un ensemble de pratiques et de méthodologies visant à améliorer le processus global de développement afin de produire des logiciels de haute qualité.

Les tests logiciels sont une sous-catégorie de l'assurance qualité qui consiste à exécuter le programme et à vérifier son comportement et ses fonctionnalités dans un environnement contrôlé. Les tests unitaires, les tests d'intégration, les tests système et les tests d'acceptation utilisateur ne sont que quelques-uns des types de tests utilisés dans le processus de test, qui vise à trouver et corriger les erreurs, les problèmes et les comportements inattendus.

5.2. Importance dans le Cycle DevOps

L'importance des tests et de l'assurance qualité dans le cycle DevOps ne peut être sous-estimée. Dans un environnement DevOps, où le développement et les opérations sont intégrés, les tests permettent de détecter les erreurs le plus tôt possible, réduisant ainsi le coût et le temps de correction.

Une bonne assurance qualité garantit que chaque étape du développement est réalisée selon des standards élevés, ce qui augmente la fiabilité et la performance du produit final.

En assurant une vérification continue et rigoureuse, les tests et la QA jouent un rôle crucial dans la livraison d'un produit sans bugs, répondant ainsi aux exigences des utilisateurs et offrant une expérience utilisateur optimale. Ils contribuent également à une meilleure collaboration entre les équipes de développement et d'opérations, facilitant ainsi une livraison continue et rapide des mises à jour et nouvelles fonctionnalités.

5.3. Les types de tests :

Il existe de nombreux types de tests, et à chaque nouvelle phase, vous en découvrirez de nouveaux. Ici, nous couvrirons les principaux types de tests et comment en tirer le meilleur parti.

5.3.1. Tests Fonctionnels

Les tests fonctionnels vérifient que le logiciel fonctionne comme prévu et se déroulent généralement dans cet ordre :

❖ Tests Unitaires :

Vérifient chaque unité ou composant individuellement pour s'assurer de leur bon fonctionnement.

❖ Tests d'Intégration :

Vérifient l'interaction entre différents modules pour identifier les problèmes de communication.

❖ Tests Système :

Évaluent le système complet pour vérifier qu'il répond aux exigences spécifiées.

❖ Tests d'Acceptation :

Valident que le système répond aux critères d'acceptation de l'utilisateur ou du client, prêt pour la mise en production.

5.3.2. Tests Non Fonctionnels

Les tests non fonctionnels évaluent les aspects du logiciel liés à la performance et à l'utilisation globale :

❖ Tests de Sécurité:

Vérifient la protection contre les attaques et les vulnérabilités pour assurer la sécurité des données.

❖ Tests de Performance :

Évaluent la vitesse, la réactivité et la stabilité sous différentes charges de travail.

❖ Tests d'Utilisabilité :

Examinent la facilité d'utilisation et l'expérience utilisateur pour garantir un logiciel intuitif.

❖ Tests de Compatibilité :

Vérifient le bon fonctionnement sur différentes plateformes, navigateurs et configurations matérielles pour une expérience utilisateur cohérente.

5.3.3. Test côté backend

❖ Introduction au Monde des Tests dans Laravel

Les tests jouent un rôle crucial dans le développement d'applications web robustes et fiables.

Dans le contexte de Laravel, un framework PHP moderne et puissant, les tests sont intégrés de manière native grâce à PHPUnit, offrant ainsi une infrastructure solide pour garantir la qualité du code tout au long du cycle de développement.

❖ Pourquoi les Tests sont Importants

Les tests permettent de vérifier que chaque partie de l'application fonctionne comme prévu.

Voici quelques raisons pour lesquelles ils sont essentiels :

Fiabilité du Code : Les tests automatisés réduisent les erreurs humaines en identifiant les problèmes avant qu'ils n'atteignent la production.

Régression : Ils protègent contre les régressions en s'assurant que les modifications ne cassent pas les fonctionnalités existantes.

Maintenance facilitée : Ils facilitent la maintenance en permettant de détecter rapidement les impacts des changements de code.

Confiance : Ils offrent une confiance accrue dans la stabilité et la sécurité de l'application.

5.3.4. Types de Tests dans Laravel

Laravel supporte plusieurs types de tests :

❖ Tests Unitaires :

Ils vérifient le comportement d'une unité de code (comme une méthode ou une fonction) de manière isolée, sans dépendre d'autres parties de l'application ou de services externes.

❖ Tests fonctionnels :

Ils simulent des interactions d'utilisateur réelles avec l'application, en testant des fonctionnalités complètes comme les routes, les contrôleurs et les requêtes HTTP.

❖ Tests d'Intégration :

Ils vérifient comment différentes parties de l'application interagissent ensemble pour assurer leur bonne coordination.

❖ Tests de Fumée (Smoke Tests) :

Ils vérifient les fonctionnalités principales et critiques de l'application pour s'assurer qu'elles sont opérationnelles.

5.4. PHPUnit et Laravel

5.4.1. Définition

Laravel utilise PHPUnit, un framework de test PHP standard, pour l'écriture et l'exécution des tests. Il fournit des fonctionnalités robustes telles que les assertions pour vérifier les résultats attendus, la gestion des dépendances, et la configuration flexible pour adapter les tests à vos besoins spécifiques.

5.4.2. Structure des Tests dans Laravel

Les tests dans Laravel suivent une structure organisée :

❖ Répertoire tests :

Contient tous les fichiers de test de votre application.

❖ Tests par Classes :

Les tests sont regroupés par fonctionnalité ou composant, souvent en utilisant des classes de test dédiées.

❖ Utilisation de Faker :

Laravel inclut Faker pour générer des données aléatoires réalistes, facilitant ainsi la création de données de test.

5.5. Prérequis et Installation

5.5.1. Prérequis

Avant de commencer l'installation, assurez-vous que votre environnement de développement répond aux prérequis suivants :

- PHP (version recommandée)
- Composer installé
- Serveur de base de données (MySQL, SQLite, etc.)

5.5.2. Tests Fonctionnels du Contrôleur ParentalController :

❖ Étapes pour Créer les Tests Fonctionnels dans Laravel

Tout d'abord, créez un fichier de test dans le répertoire tests/Feature de votre application Laravel. Vous pouvez nommer ce fichier comme suit : Cela générera un fichier ParentalControllerTest.php dans le répertoire tests/Feature.

Utilisez les classes et les traits nécessaires pour configurer votre environnement de test. Assurez-vous d'importer les modèles nécessaires (Parental et User), ainsi que les classes de base de PHPUnit et de Laravel.

Maintenant, vous pouvez commencer à écrire vos tests pour les différentes fonctionnalités du contrôleur ParentalController.

Pour exécuter tous les tests définis dans ce fichier et vérifier leur bon fonctionnement : Cela exécutera PHPUnit qui parcourra tous les fichiers de tests dans le répertoire tests et fournira un rapport sur les résultats des tests, y compris les succès, les échecs éventuels et les erreurs rencontrées.

En suivant ces étapes, vous pouvez écrire et exécuter efficacement des tests fonctionnels pour le contrôleur ParentalController de votre application Laravel, assurant ainsi la qualité et la fiabilité des fonctionnalités que vous développez.

Dans le contexte de Laravel, les utilitaires Faker et RefreshDatabase sont des outils essentiels pour simplifier le processus de développement et de test. Voici comment ils sont utilisés et comment ils facilitent la création de tests plus efficaces :

5.5.3. Utilisation de Faker

Faker est une bibliothèque PHP qui permet de générer des données aléatoires réalistes.

Dans le contexte de Laravel, Faker est intégré nativement et est souvent utilisé dans les tests pour créer des données de manière aléatoire et cohérente. Cela est particulièrement utile pour les tests fonctionnels où il est nécessaire de simuler des données variées sans avoir à les saisir manuellement.

5.5.4. Utilisation de RefreshDatabase

RefreshDatabase est un trait fourni par Laravel qui permet de réinitialiser la base de données avant l'exécution des tests. Cela signifie que chaque fois que vous lancez vos tests, la base de données est vidée et migrée vers un état initial connu. Cela garantit que chaque test commence dans un environnement de base de données propre et cohérent, sans dépendances des tests précédents ou d'anciennes données résiduelles.

Lorsque vous exécutez vos tests avec RefreshDatabase, Laravel s'assure que

- Avant chaque test, la base de données est réinitialisée.
- Toutes les migrations sont exécutées pour mettre à jour le schéma de la base de données.
- Les données de test sont insérées à partir de zéro, ce qui crée un environnement cohérent et prévisible pour chaque test.

5.5.5. Avantages

- **Facilité de Développement** : Faker simplifie la création de données aléatoires réalistes, ce qui accélère le processus de développement et de test.
- **Consistance des Tests** : RefreshDatabase garantit que chaque test s'exécute dans un environnement isolé et propre, évitant les effets secondaires entre les tests.
- **Réduction des Erreurs** : En utilisant ces outils, vous minimisez les erreurs humaines liées à la création de données de test manuellement et à la gestion de la base de données.

En combinant Faker et RefreshDatabase, vous pouvez créer des tests robustes et fiables pour votre application Laravel, en assurant une meilleure qualité du code et une efficacité accrue dans le processus de développement. Pour que notre code fonctionne correctement avec les factories dans Laravel, nous devons suivre plusieurs étapes clés pour assurer la création et la

validation cohérente des données de test. Voici comment nous pouvons accomplir cela :

❖ **Création de la Factory ParentalFactory :**

Nous devons d'abord définir une factory pour le modèle Parental. Cette factory est responsable de générer des données aléatoires cohérentes pour chaque instance de Parental que nous voulons utiliser dans nos tests.

Définition des Attributs par Défaut :

À l'intérieur de notre factory, dans la méthode `definition()`, nous spécifions comment chaque attribut du modèle Parental doit être généré. Par exemple, nous utilisons des méthodes fournies par Faker pour créer des valeurs réalistes telles que des pays, des villes, des codes postaux, des adresses, des numéros de téléphone, des dates de naissance, etc. De plus, nous associons chaque Parental à un utilisateur existant à l'aide de `User::factory()` pour garantir des relations correctes entre les entités de notre application.

❖ **Utilisation des Factories dans les Tests :**

Une fois que la factory est définie, nous l'utilisons dans nos tests fonctionnels. Par exemple, dans notre test `it_can_create_a_parental()`, nous appelons `Parental::factory()->make()->toArray()` pour générer un tableau de données représentant un Parental. Ce tableau est ensuite utilisé pour simuler une requête JSON POST vers notre API (`postJson('/api/parental', $parentalData)`). Nous validons ensuite que l'API retourne un statut HTTP 201 (Créé) et que les données sont bien enregistrées dans la base de données avec `assertDatabaseHas()`.

En suivant ces étapes méthodiquement, nous nous assurons que nos tests sont robustes, fiables et reflètent les conditions réelles d'utilisation de notre application. Cela garantit également que nos données de test sont cohérentes et que nos tests fonctionnels peuvent être exécutés de manière isolée et répétable, contribuant ainsi à la qualité globale de notre code dans le développement de notre application Laravel.

Maintenant que votre factory `ParentalFactory` est définie, vous pouvez l'utiliser dans vos tests pour créer des données de manière simple et efficace.

Par exemple, dans votre fichier de test `ParentalControllerTest.php` :
Comme montré dans le code déjà partagé.

❖ Explication :

- `Parental::factory()->make()` : Cette méthode crée une instance de `Parental` avec des valeurs aléatoires générées par la factory `ParentalFactory`. `make()` crée l'instance mais ne la persiste pas en base de données.
- `toArray()` : Transforme l'objet `Parental` en tableau, ce qui est nécessaire pour passer les données à `postJson()` pour créer une nouvelle instance via une requête JSON.
- `assertDatabaseHas()` : Méthode de PHPUnit qui vérifie si les données spécifiées existent dans la base de données après l'exécution de la requête.

5. Test côté Frontend

5.6. Introduction :

5.6.1. Qu'est-ce que les tests frontend?

Les tests frontend vérifient la partie visible d'une application web, c'est-à-dire l'interface utilisateur. Ils s'assurent que tous les éléments visuels et fonctionnels du site web ou de l'application fonctionnent comme prévu. Cela inclut :

- Interface Utilisateur (UI) : Vérification des boutons, menus, formulaires, animations et autres composants graphiques.
- Performance : S'assurer que l'application se charge rapidement et fonctionne de manière fluide sur différents navigateurs et appareils.
- Expérience Utilisateur (UX) : Vérifier que la navigation est intuitive et que toutes les fonctionnalités sont accessibles et utilisables sans erreurs.

Les outils comme Playwright sont souvent utilisés pour automatiser ces tests, simulant les interactions de l'utilisateur et générant des rapports détaillés.

5.7. Introduction aux Tests E2E et Tests Unitaires

5.7.1. Tests End-to-End (E2E)

Les tests E2E vérifient le bon fonctionnement d'une application du début à la fin, en simulant le comportement d'un utilisateur réel. Ils couvrent tout le flux de travail, du front-end au backend, et détectent les problèmes d'intégration et de performance. En reproduisant les actions des utilisateurs réels, ils garantissent une expérience utilisateur cohérente et fiable.

5.7.2. Tests Unitaires

Les tests unitaires vérifient le bon fonctionnement d'unités individuelles de code, comme des fonctions ou des méthodes, de manière isolée. Ils sont essentiels pour garantir la robustesse du code et faciliter la maintenance, permettant de détecter rapidement les régressions et assurant que chaque composant fonctionne comme prévu.

5.7.3. Introduction à Playwright

Playwright est un outil de test automatisé développé par Microsoft. Il permet aux développeurs et aux testeurs de contrôler les navigateurs web

(Chromium, Firefox, WebKit) à travers un ensemble d'API. Playwright offre une exécution de tests rapide et fiable, permettant de simuler des interactions complexes avec les applications web, et s'assure que celles-ci fonctionnent correctement sur différentes plateformes et configurations de navigateur.

5.7.4. Importance des Tests E2E

Les tests E2E sont essentiels pour s'assurer que toutes les parties de l'application fonctionnent correctement ensemble dans un environnement réel. Ils permettent de

- Détecter les problèmes d'intégration : Vérifient que les différentes parties de l'application interagissent correctement, identifiant et résolvant les problèmes d'intégration.
- Assurer la qualité de l'expérience utilisateur : Simulent les actions des utilisateurs pour garantir que l'application répond aux attentes et besoins des utilisateurs finaux.
- Réduire les risques de régressions : Vérifient que les nouvelles fonctionnalités ou modifications n'ont pas introduit de bugs dans le système, maintenant la stabilité et la fiabilité de l'application.

5.7.5. Pourquoi Tester les Pages avec des Programmes au Lieu de Manuellement ?

- Gain de temps et efficacité : Exécution rapide de centaines de scénarios de test, contrairement aux tests manuels plus lents et fastidieux.
- Fiabilité et précision: Élimination des erreurs humaines, garantissant des résultats cohérents et une couverture de test exhaustive.
- Facilité de maintenance : Tests automatisés réutilisables et facilement mis à jour en cas de modifications du code, améliorant la gestion des versions et la continuité des tests.
- Couverture étendue: Test de cas d'utilisation multiples et complexes, augmentant la couverture de test et la fiabilité de l'application.
- Détection précoce des bugs : Intégration des tests automatisés dans le cycle de développement (CI/CD), permettant de détecter et corriger les bugs rapidement avant la production.

5.8. Methodologie de travail :

Dans notre méthodologie de travail, nous commençons par prendre le code du front-end via un ticket Jira. Ensuite, nous procédons à une vérification visuelle et répertorions toutes les fonctionnalités à tester. Nous effectuons d'abord des tests manuels pour identifier les problèmes évidents. Une fois cette étape terminée, nous passons à l'automatisation des tests en utilisant Playwright. Cette automatisation nous permet d'obtenir des rapports détaillés et précis que nous transmettons ensuite à l'équipe de développement pour qu'elle puisse corriger les éventuelles anomalies détectées. Cette approche permet d'assurer une couverture de test complète et efficace, tout en garantissant que les problèmes soient identifiés et résolus rapidement.

5.8.1. Installation Et Configuration:

Pour installer Playwright , suivez ces étapes :

❖ Clonage du Projet :

Clonez le projet depuis votre référentiel Git :

Créez une autre branche dédiée à Playwright basée sur la branche frontend afin que l'environnement de test soit un peu différent de l'environnement de développement.

Pour installer Playwright on exécute la commande suivante : `Npx playwright install`

❖ Configuration de l'Environnement :

Afin de travailler efficacement avec Playwright, il est nécessaire de configurer correctement l'environnement de test. Cela comprend la configuration des navigateurs à utiliser, la possibilité de conserver des vidéos des simulations, ainsi que la définition d'un site web ou d'un serveur local pour les tests.

Dans le fichier ``playwright.config.js``, nous spécifions tous les paramètres relatifs à Playwright, tels que les navigateurs à utiliser pour les simulations, les options pour conserver les enregistrements vidéo des tests, et les configurations spécifiques pour les différents environnements de test.

De plus, il est important de changer le type de module dans le fichier ``package.json`` de "module" à "commonjs" pour permettre la compatibilité avec les modules CommonJS. Les modules CommonJS utilisent ``require()``

pour charger les modules et ``module.exports`` pour exposer des fonctionnalités, tandis que les modules ECMAScript utilisent ``export`` et ``import`` pour gérer les dépendances.

Cette configuration permet de structurer le projet de manière à ce que les tests soient écrits dans un dossier dédié, séparé des composants principaux de l'application. Chaque fichier de test doit être nommé avec le suffixe ``.spec.js`` pour indiquer qu'il s'agit d'un fichier de test Playwright.

Alors on les séparer en mettant dans une branche nos tests et dans l'autre le code frontend

5.8.2. Interface Graphique Playwright :

Lors de l'exécution des tests, les résultats s'affichent de manière détaillée, indiquant le nom du fichier testé, les différents tests exécutés avec une icône de succès ou d'échec à côté, le nombre de tests réussis et échoués, ainsi que les navigateurs utilisés pour les tests. En cas d'échec, il est possible de cliquer sur les tests échoués pour voir pourquoi ils ont échoué et déterminer si l'erreur provient du test lui-même ou de l'application.

Cette méthodologie permet d'organiser les résultats de test de manière claire et de les partager avec l'équipe de développement pour une résolution rapide des problèmes.

5.8.3. Responsivité :

La responsivité est importante car elle fait partie de l'expérience utilisateur. Une personne n'utilisera pas un site web à moins qu'il soit simple et esthétiquement plaisant, et l'une des choses importantes à cet égard est la responsivité, car chacun de nous utilise différents types de navigateurs tels que Apple, Windows, Chrome, Firefox, Apple, Android, etc. Il est donc nécessaire de vérifier la visibilité pour chacun d'entre eux.

Pour ce faire, nous pouvons adopter deux approches : soit le faire manuellement, soit donner à Playwright le pouvoir de lancer les tests sur tous les différents navigateurs. La puissance de Playwright réside ici, car il n'est pas nécessaire d'avoir tous les navigateurs pour tester le code sur chacun d'entre eux, ce qui permet d'atteindre deux objectifs en une seule action.

Dans le but d'automatiser les tests, nous pouvons simplement lancer les tests et vérifier la responsivité ainsi que d'autres aspects sur tous les navigateurs différents.

Comme vous pouvez le voir dans cette image, nous pourrions adopter l'approche normale et redimensionner les fenêtres encore et encore.

5.8.4. Test Manuel :

Ou bien nous pourrions simplement l'automatiser. Par exemple, le code ouvre simplement le navigateur sur la page, et comme vous pouvez le voir sur les résultats, nous avons tous les navigateurs différents avec leurs tailles d'écran, etc.

5.9. Conclusion

Les tests Laravel couvrent un large éventail de scénarios, incluant les tests unitaires pour vérifier le bon fonctionnement des composants individuels, les tests d'intégration pour s'assurer que les différentes parties de l'application fonctionnent bien ensemble, et les tests fonctionnels pour vérifier le bon fonctionnement des flux de travail complets. En intégrant ces tests dans notre processus de développement, nous pouvons automatiser efficacement ces vérifications, ce qui permet de maintenir la qualité de notre application tout en réduisant le temps et les efforts nécessaires. Cela garantit une application fiable et performante pour nos utilisateurs. De même, les tests avec Playwright peuvent couvrir divers scénarios, tels que les tests de responsivité pour vérifier l'affichage sur différents appareils, les tests de navigation pour s'assurer que les liens mènent aux bonnes pages, et les tests de fonctionnalités pour vérifier le bon fonctionnement des boutons et autres éléments interactifs. En intégrant Playwright dans notre processus de développement, nous pouvons également automatiser ces tests de manière efficace, assurant ainsi une qualité constante tout en minimisant le temps et les efforts requis. Cela aboutit à une application fiable et performante pour nos utilisateurs.

6. Partie Git :

6.1. C'est quoi le contrôle de version :

Le contrôle de version aide les développeurs à suivre et à gérer les modifications apportées au code d'un projet logiciel. Au fur et à mesure qu'un projet logiciel prend de l'ampleur, le contrôle de version devient essentiel. Prenez WordPress...

6.2. Le rôle de contrôle de version en cycle Devops:

Le contrôle de version est essentiel pour DevOps, car il prend en charge les principaux objectifs de rapidité, de qualité et de collaboration. Les développeurs peuvent bénéficier du contrôle de version en ayant accès à l'ensemble de l'historique et de l'état du code, ce qui leur permet de travailler plus rapidement et plus efficacement. La qualité et la fiabilité peuvent être améliorées en testant et en révisant le code avant de le fusionner avec la branche principale, ainsi qu'en résolvant plus facilement les conflits et les bogues. En outre, le contrôle de version permet aux développeurs de mieux collaborer et communiquer en partageant et en synchronisant le code avec d'autres membres de l'équipe, ainsi qu'en utilisant des messages de validation et des demandes de tirage pour documenter et discuter des modifications. Il y'en a de divers Logiciel , Mais on utilise git

6.2.1. Pourquoi Git?

❖ C'est quoi Git?

Git est un système de contrôle de version distribué (VCS). Il permet à plusieurs utilisateurs de suivre les modifications du code source durant le développement logiciel, en conservant un historique des changements de code et en assurant la traçabilité. C'est comme un journal numérique partagé pour le code informatique, permettant à de nombreuses personnes d'écrire dedans tout en gardant une trace de toutes les modifications effectuées au fil du temps.

❖ Pourquoi Git?

Open source : Git est un logiciel libre, ce qui permet aux développeurs de l'examiner, de modifier ou d'améliorer le code source. Populaire : Largement adopté par la communauté des développeurs, Git est le système de contrôle de

version le plus utilisé dans le monde.

Performances : Exceptionnellement efficace pour gérer de grands dépôts de source grâce à sa conception efficace des données.

Sécurité : Intègre des fonctionnalités robustes de sécurité pour maintenir l'intégrité du code source et gérer l'accès.

Basé sur les fonctionnalités : Permet une gestion de projet flexible où chaque nouvelle fonctionnalité est développée dans des branches séparées. **Basées sur les rôles** : Supporte la définition de lignes de code basées sur les rôles, permettant des contrôles d'accès flexibles.

Flux multiples : Supporte plusieurs workflows, tels que Gitflow, facilitant ainsi le développement simultané de diverses fonctionnalités.

Sauvegardes multiples : Offre la possibilité de multiples sauvegardes, chaque clone agissant comme une sauvegarde complète avec toute l'historique.

6.3. Github :

Pour travailler sur le même code depuis différents endroits, nous avons besoin d'une plateforme, et rien n'est mieux adapté que GitHub à cet effet.

6.3.1. Qu'est-ce que GitHub ?

GitHub est un site web et un service cloud qui assiste les développeurs dans le stockage et la gestion de leur code. Il permet également de suivre et de contrôler les modifications apportées à ce dernier. Pour bien comprendre ce qu'est GitHub, il est crucial de connaître deux concepts fondamentaux liés à ce service : Le contrôle de version Git Philosophie Open Source: GitHub est également un espace de collaboration où les développeurs du monde entier peuvent participer à des projets professionnels comme open source. Ils peuvent utiliser et améliorer le code d'autres, le tout animé par une philosophie de l'open source.

6.3.2. Comment créer un dépôt ?

Créer un compte GitHub : Rendez-vous sur GitHub et inscrivez-vous. Suivez les instructions pour compléter la création de votre compte.

Créer un nouveau dépôt : Une fois connecté, cliquez sur le bouton "New Repository". Donnez un nom à votre dépôt, ajoutez une description si nécessaire. Pour des raisons de sécurité, sélectionnez "Private" pour rendre le dépôt privé. Cochez les options pour ajouter un fichier README et un fichier .gitignore

6.3.3. Création des branches :

Pour chaque membre de l'équipe, créez une branche personnelle ou bien le je le crée moi-même : **git checkout -b nom-de-branche** Cela permet à chaque membre de se familiariser avec GitHub.

❖ Travail collaboratif sur les branches :

Une fois les membres à l'aise, passez à une organisation par équipe avec une branche principale par équipe et des branches spécifiques pour chaque fonctionnalité.

Utilisez les branches de la manière suivante :

- ✓ main : Branche principale pour la production.
- ✓ Premaster : Branche avant la production.
- ✓ Frontend-Branch : Branche de développement frontend pour intégrer les nouvelles pages.
- ✓ Backend-global : Branche de développement backend pour intégrer les nouvelles fonctionnalités, la sécurité et la création des tests avec le responsable des tests backend.
- ✓ feature/nouvelle-fonctionnalité : Branche pour le développement de nouvelles fonctionnalités spécifiques.
- ✓ dev_env : Branche pour les responsables d'administration système.
- ✓ Playwright : Branche pour les tests E2E.

6.4. Intégration des Pratiques DevOps

6.4.1. Principes DevOps:

Pour comprendre la méthodologie DevOps dans le contexte de l'utilisation de Git, GitHub et GitHub Actions, il est important de voir comment ces outils soutiennent et renforcent les principes du DevOps :

❖ **Collaboration et Contrôle de Version :**

Répertoire Partagé : Git et GitHub offrent une plateforme centralisée permettant aux équipes de collaborer sur du code. Cela favorise la transparence et permet à plusieurs développeurs de travailler simultanément sur la même base de code.

❖ **Contrôle de Version :**

Les capacités de branching et de merging de Git permettent aux équipes de gérer efficacement différentes versions du code. GitHub améliore cela en fournissant une interface graphique et des outils pour la revue de code, le suivi des problèmes et la gestion de projet.

❖ **Intégration Continue et Déploiement Continue (CI/CD) :**

Workflows Automatisés : GitHub Actions automatise le processus de construction, de test et de déploiement du code. Cela est conforme aux principes CI/CD qui visent à automatiser les tâches répétitives pour obtenir une livraison de logiciel plus rapide et plus fiable.

Intégration avec les Outils : GitHub Actions s'intègre parfaitement avec divers outils et services, permettant aux équipes de personnaliser les workflows en fonction de leurs besoins spécifiques en matière de développement et de déploiement.

6.4.2. Pratiques DevOps Prises en Charge par Git et GitHub :

❖ **Communication et Collaboration :**

Suivi des Problèmes : Les Issues GitHub facilitent la communication sur les tâches, les bogues et les fonctionnalités entre les membres de l'équipe. Cela aide à planifier et à prioriser efficacement le travail.

Pull Requests : La fonctionnalité de pull request de GitHub encourage la revue de code et la collaboration. Elle favorise une culture de partage des connaissances et de maintien de la qualité du code grâce aux retours et aux discussions.

❖ **Automatisation et Surveillance :**

GitHub Actions : Automatise des tâches telles que les tests, la construction et le déploiement des applications. Cela réduit l'intervention manuelle, accélère les boucles de feedback et améliore la fiabilité globale du déploiement.

Surveillance : Les GitHub Actions peuvent être configurées pour inclure des tâches de surveillance après le déploiement, telles que des vérifications de santé et des métriques de performance. Cela garantit que les applications restent stables et performantes en environnement de production.

❖ **Infrastructure as Code (IaC) :**

Git pour la Gestion de Configuration : Git peut gérer les configurations d'infrastructure et les scripts en tant que code. Associé à GitHub, les équipes peuvent contrôler la version des changements d'infrastructure et collaborer sur la provision et la gestion des ressources de manière reproductible.

6.4.3. Conclusion :

Dans le domaine du développement logiciel moderne, le trio Git, GitHub et GitHub Actions représente des piliers d'efficacité, de collaboration et d'automatisation. Ces outils révolutionnent la manière dont les équipes gèrent le code, collaborent de manière transparente et automatisent les flux de travail, favorisant ainsi l'innovation et la fiabilité des projets logiciels.

7. Administration des Systèmes

7.1. Configuration DNS

7.1.1. Fonctionnement du DNS

Le DNS (Domain Name System) traduit les noms de domaine en adresses IP pour permettre aux navigateurs d'accéder aux sites web. Voici les étapes principales :

Requête DNS : Le navigateur envoie une requête pour trouver l'adresse IP associée à un nom de domaine.

Résolution récursive : Le résolveur DNS interroge plusieurs niveaux de serveurs pour obtenir l'adresse IP.

Réponse DNS : L'adresse IP trouvée est renvoyée au navigateur pour accéder au site web.

7.1.2. Tester le Fonctionnement DNS

Pour tester le DNS, utilisez des commandes comme `nslookup` ou `dig` pour vérifier la résolution des noms de domaine.

❖ Configuration de NGINX

NGINX est un serveur web performant utilisé pour :

Serveur web : Servir des pages web en réponse aux requêtes des utilisateurs.

Proxy inverse : Agir comme intermédiaire entre les clients et les serveurs backend, améliorant la sécurité et la gestion des requêtes.

Équilibrage de charge : Répartir les requêtes entre plusieurs serveurs pour optimiser les ressources et assurer la disponibilité.

Pour configurer NGINX, suivez les étapes suivantes :

Installation de NGINX : Installez NGINX sur votre serveur en utilisant les commandes appropriées pour votre distribution (par exemple, `apt-get install nginx` pour Ubuntu).

Configuration des Fichiers : Modifiez les fichiers de configuration situés généralement dans `/etc/nginx/nginx.conf` ou dans le dossier `sites-available` pour définir les serveurs virtuels et les paramètres spécifiques de votre application.

Test de la Configuration : Utilisez la commande `nginx -t` pour tester la validité des fichiers de configuration.

Redémarrage de NGINX : Appliquez les modifications en redémarrant NGINX avec `systemctl restart nginx`.

7.1.3. Configuration de POSTFIX

❖ Fonctionnalités de POSTFIX

POSTFIX est un serveur de messagerie utilisé pour envoyer et recevoir des emails. Il peut être configuré pour rediriger les emails et activer la validation en deux étapes.

Pour tester POSTFIX, envoyez un email de test à une boîte de réception et vérifiez qu'il a bien été reçu.

Installation de POSTFIX : Installez POSTFIX en utilisant les commandes appropriées (`apt-get install postfix` pour Ubuntu).

Configuration de la Redirection : Modifiez le fichier de configuration pour rediriger les emails et activer les fonctionnalités requises.

Envoi d'un Email de Test : Utilisez un client de messagerie ou une commande de terminal (`sendmail` par exemple) pour envoyer un email de test.

❖ Sécurité des Serveurs

Pour protéger les serveurs contre les accès non autorisés, un pare-feu tel que UFW (Uncomplicated Firewall) est configuré. UFW permet de contrôler précisément les connexions entrantes et sortantes, en autorisant uniquement le trafic nécessaire, comme SSH pour l'administration à distance et HTTP/HTTPS pour les services web.

Installation de UFW : Installez UFW si nécessaire (`apt-get install ufw` pour Ubuntu).

Configuration des Règles : Définissez les règles pour autoriser ou bloquer le trafic (`ufw allow 22` pour SSH, `ufw allow 80` pour HTTP, `ufw allow 443` pour HTTPS).

Activation du Pare-feu : Activez UFW avec la commande `ufw enable`.

❖ SSH et SSL

SSH : Utilisation de clés SSH pour une authentification sécurisée, renforçant la protection contre les attaques par force brute.

Génération des Clés SSH : Utilisez `ssh-keygen` pour générer une paire de clés SSH.

Configuration du Serveur : Ajoutez la clé publique au fichier `~/.ssh/authorized_keys` sur le serveur.

Connexion Sécurisée : Connectez-vous au serveur en utilisant la clé privée.

SSL : Les certificats SSL, obtenus via Let's Encrypt, chiffrent les communications entre les clients et les serveurs, garantissant la sécurité des données échangées.

Obtention des Certificats : Utilisez Certbot pour obtenir et installer les certificats SSL (`apt-get install certbot python3-certbot-nginx`).

Configuration de NGINX pour SSL : Modifiez la configuration de NGINX pour inclure les directives SSL.

Renouvellement Automatique : Configurez le renouvellement automatique des certificats avec Cron.

❖ Cloud AWS Provider

Création d'une Instance Cloud : Utilisez l'interface AWS ou les outils en ligne de commande pour créer une nouvelle instance EC2.

Configuration de NGINX : Configurez NGINX pour servir des applications web sur des instances AWS en suivant les étapes mentionnées précédemment.

Configuration DNS : Configurez les serveurs DNS pour la résolution des noms de domaine publics en utilisant Route 53 ou un autre service DNS.

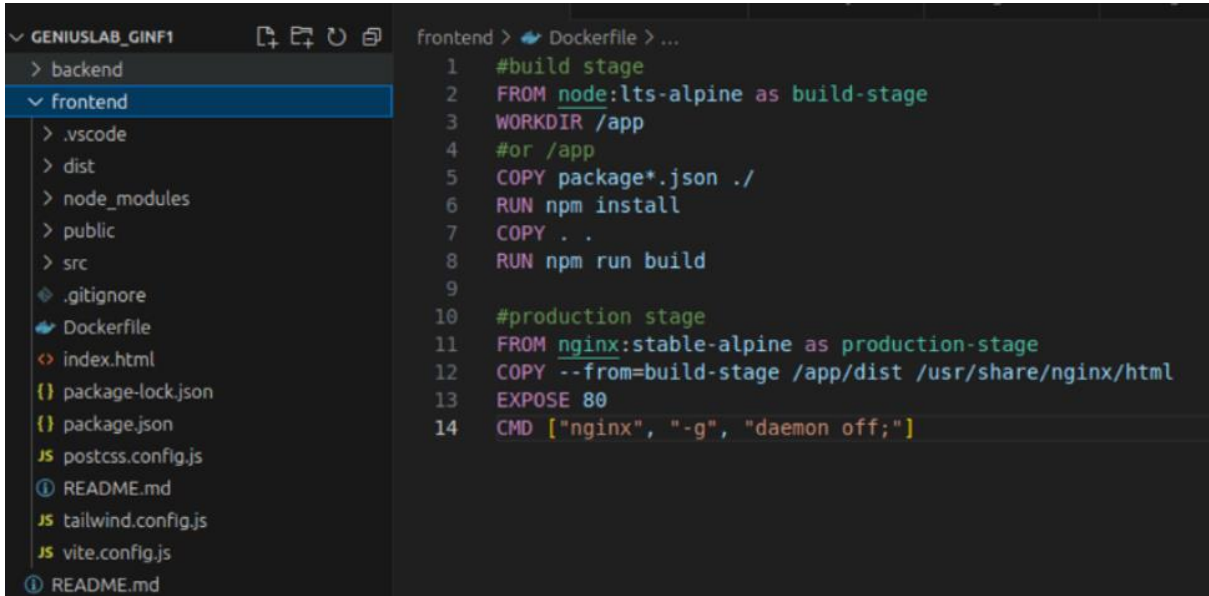
Nom de Domaine Public : Associez un nom de domaine public à votre instance EC2 pour rendre les services accessibles sur Internet.

7.2. Conteneurisation

7.2.1. Dockerisation d'un Projet Laravel/Vue.js

❖ Dockerisation Frontend

Création du Dockerfile : Créez un Dockerfile pour l'application frontend utilisant Vue.js.



```
1 #build stage
2 FROM node:lts-alpine as build-stage
3 WORKDIR /app
4 #or /app
5 COPY package*.json ./
6 RUN npm install
7 COPY . .
8 RUN npm run build
9
10 #production stage
11 FROM nginx:stable-alpine as production-stage
12 COPY --from=build-stage /app/dist /usr/share/nginx/html
13 EXPOSE 80
14 CMD ["nginx", "-g", "daemon off;"]
```

Construction de l'Image Docker : Utilisez la commande `docker build -t frontend:latest .` pour construire l'image Docker.

Test Local : Testez l'application localement en utilisant `npm run serve`.

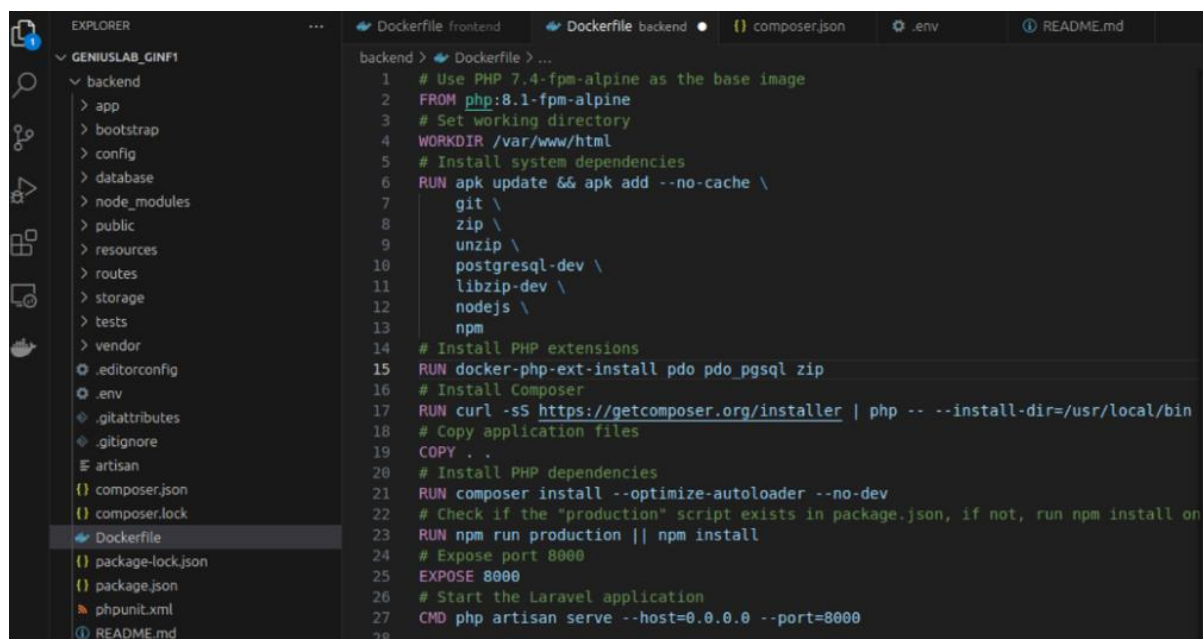
Exécution du Conteneur : Lancez le conteneur Docker avec `docker run -p 8080:8080 frontend:latest`.

Push sur Docker Hub : Connectez-vous à Docker Hub et poussez l'image avec `docker push your_dockerhub_username/frontend:latest`.

❖ Dockerization Backend

Création du Dockerfile : Créez un Dockerfile pour l'application backend

utilisant Laravel.



Construction de l'Image Docker : Utilisez la commande `docker build -t backend:latest .` pour construire l'image Docker.

Test Local : Testez l'application localement en utilisant Composer.

Exécution du Conteneur : Lancez le conteneur Docker avec `docker run -p 9000:9000 backend:latest`.

Push sur Docker Hub : Connectez-vous à Docker Hub et poussez l'image avec `docker push your_dockerhub_username/backend:latest`.

7.2.2. Déploiement Automatique CI/CD

Le workflow CI/CD inclut des étapes automatisées pour construire et pousser les images Docker vers Docker Hub, décrites dans un fichier YAML. Ce fichier définit les étapes suivantes :

Récupération du Code Source : Utilisation de GitHub Actions pour récupérer le code source.

Construction de l'Image Docker : Construction de l'image Docker à partir du Dockerfile.

Authentification Docker Hub : Authentification auprès de Docker Hub pour permettre le push de l'image.

Push de l'Image : Poussée de l'image vers le registre Docker Hub.

