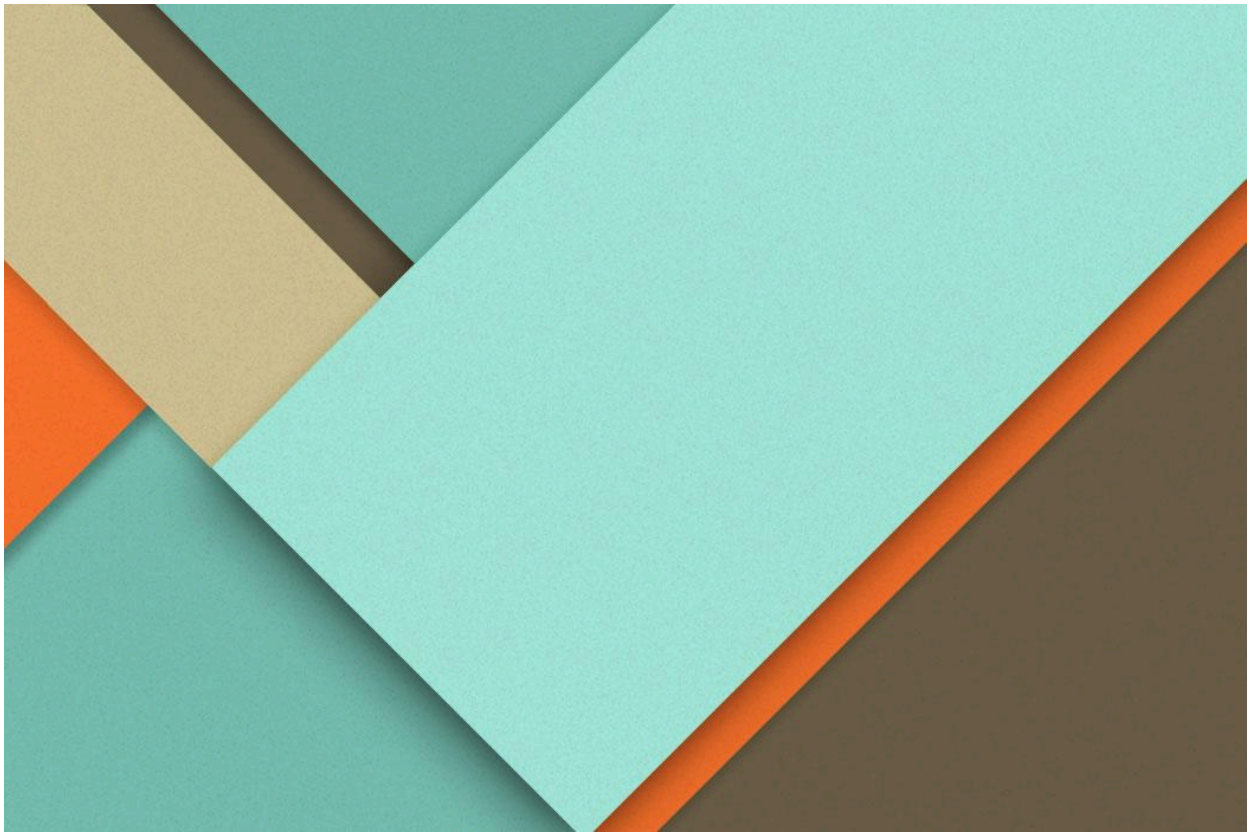


ApiCovoiturage

03/03/2024

Miodrag Mihajlovic
CDA - Greta Bretagne Sud
Vannes

Introduction

Ce document technique est destiné à fournir une compréhension détaillée de l'application que nous avons développée. L'application est construite en utilisant le langage de programmation PHP et le framework Symfony. Elle utilise également le bundle LexikJWTAuthenticationBundle pour l'authentification, le bundle Doctrine pour la gestion de la base de données, et le bundle NelmioApiDoc pour la documentation de l'API. L'objectif principal de cette application est de fournir une API robuste et sécurisée. Cette API permet aux utilisateurs d'interagir avec l'application de manière efficace et sécurisée. Les détails spécifiques de l'API, y compris les routes disponibles et les exemples de requêtes et de réponses, seront discutés plus en détail dans les sections suivantes de ce document. Ce document couvrira également les détails de la configuration de l'application, les fonctionnalités de l'application, le guide d'utilisation de l'API, les tests, le déploiement de l'application, et la maintenance et le support. Nous espérons que ce document vous aidera à comprendre notre application de manière plus approfondie et à l'utiliser efficacement.

Description de l'application

L'application est une API appelée ApiCovoiturage. Elle est construite en utilisant le langage de programmation PHP et le gestionnaire de dépendances Composer. Le framework principal utilisé est Symfony, qui fournit une structure solide et flexible pour le développement de l'application. L'application utilise plusieurs bundles pour fournir des fonctionnalités spécifiques. Le bundle LexikJWTAuthenticationBundle est utilisé pour l'authentification, permettant aux utilisateurs de se connecter de manière sécurisée à l'application. Le bundle Doctrine est utilisé pour la gestion de la base de données, facilitant l'interaction avec les données stockées. Le bundle NelmioApiDoc est utilisé pour la documentation de l'API, fournissant une interface utilisateur pour visualiser et interagir avec l'API. L'application expose plusieurs routes sous le chemin /api, à l'exception de /api/doc qui est réservé pour la documentation de l'API. Une de ces routes est /api/login_check, qui est utilisée pour l'authentification des utilisateurs. La version actuelle de l'API est 1.0.0, comme indiqué dans la configuration de NelmioApiDoc. Dans les sections suivantes de ce document, nous détaillerons davantage l'architecture de l'application, la configuration de l'API, les fonctionnalités de l'application, le guide d'utilisation de l'API, les tests, le déploiement de l'application, et la maintenance et le support.

Architecture de l'application

L'architecture de l’application ApiCovoiturage est basée sur le modèle MVC (Modèle-Vue-Contrôleur) qui est couramment utilisé dans le développement web. Ce modèle divise l'application en trois composants interconnectés, ce qui permet une séparation des préoccupations.

Langages de programmation utilisés

L'application est principalement écrite en PHP, un langage de programmation largement utilisé pour le développement web. PHP est utilisé pour la logique métier de l'application, y compris le traitement des requêtes HTTP, l'interaction avec la base de données, et la génération des réponses HTTP.

Diagrammes

Dans cette section, nous allons présenter les différents diagrammes qui représentent l'architecture et la structure de l’application.

Diagramme de Classe - modéle

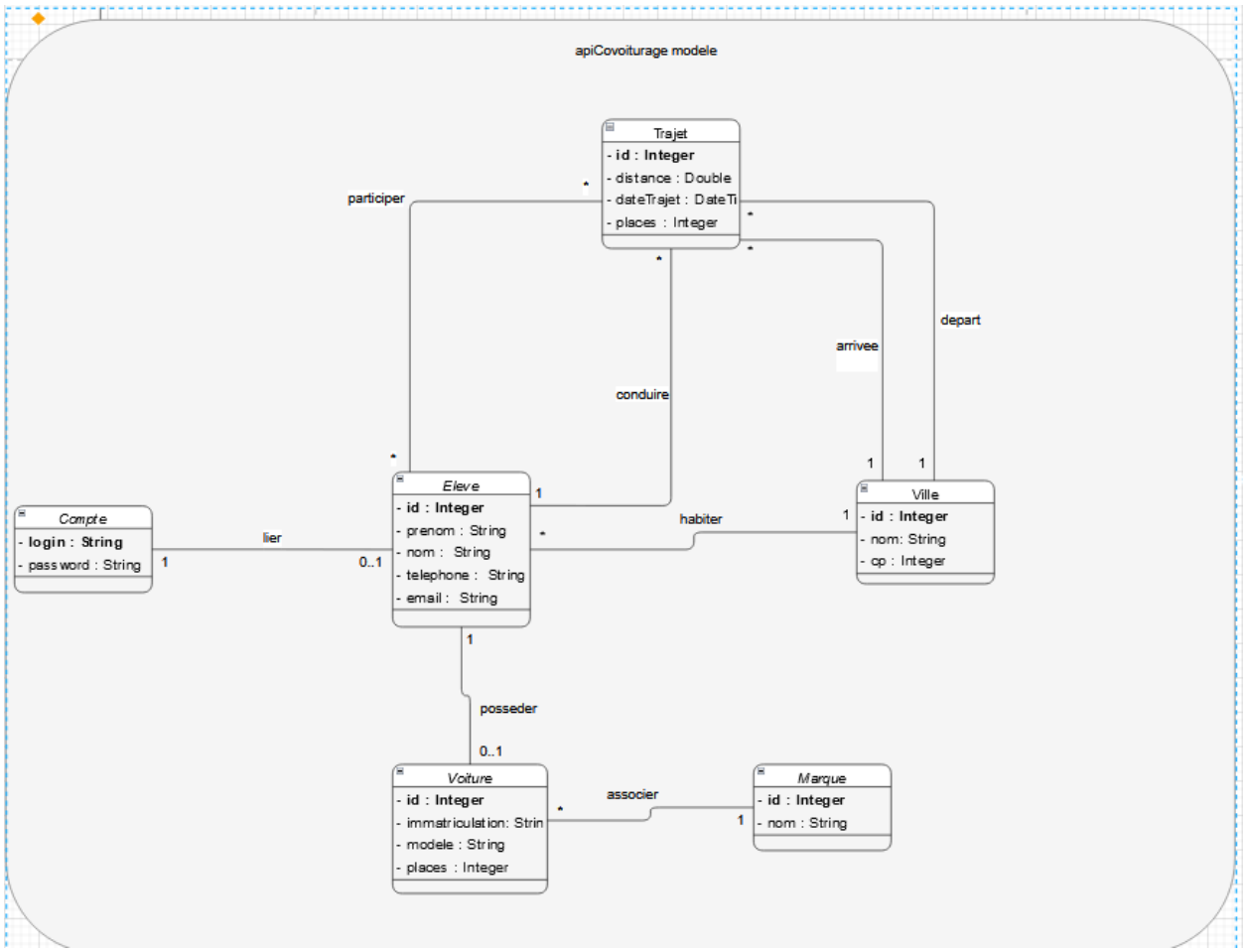
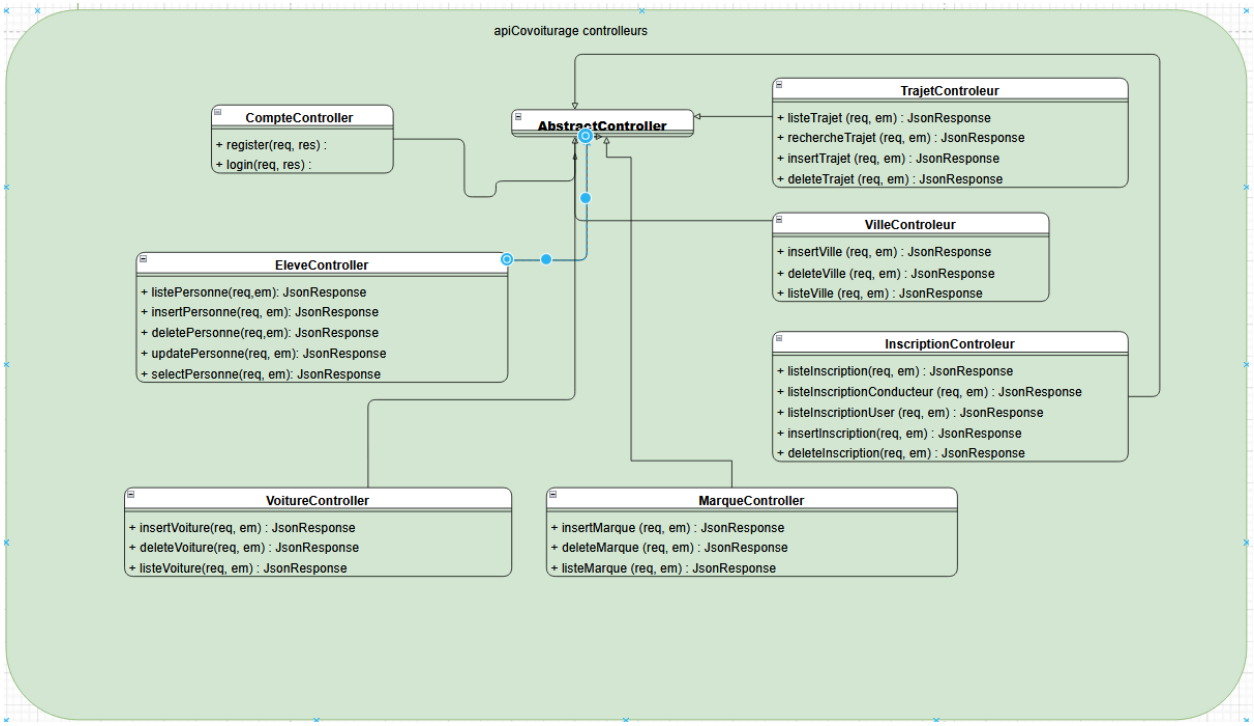
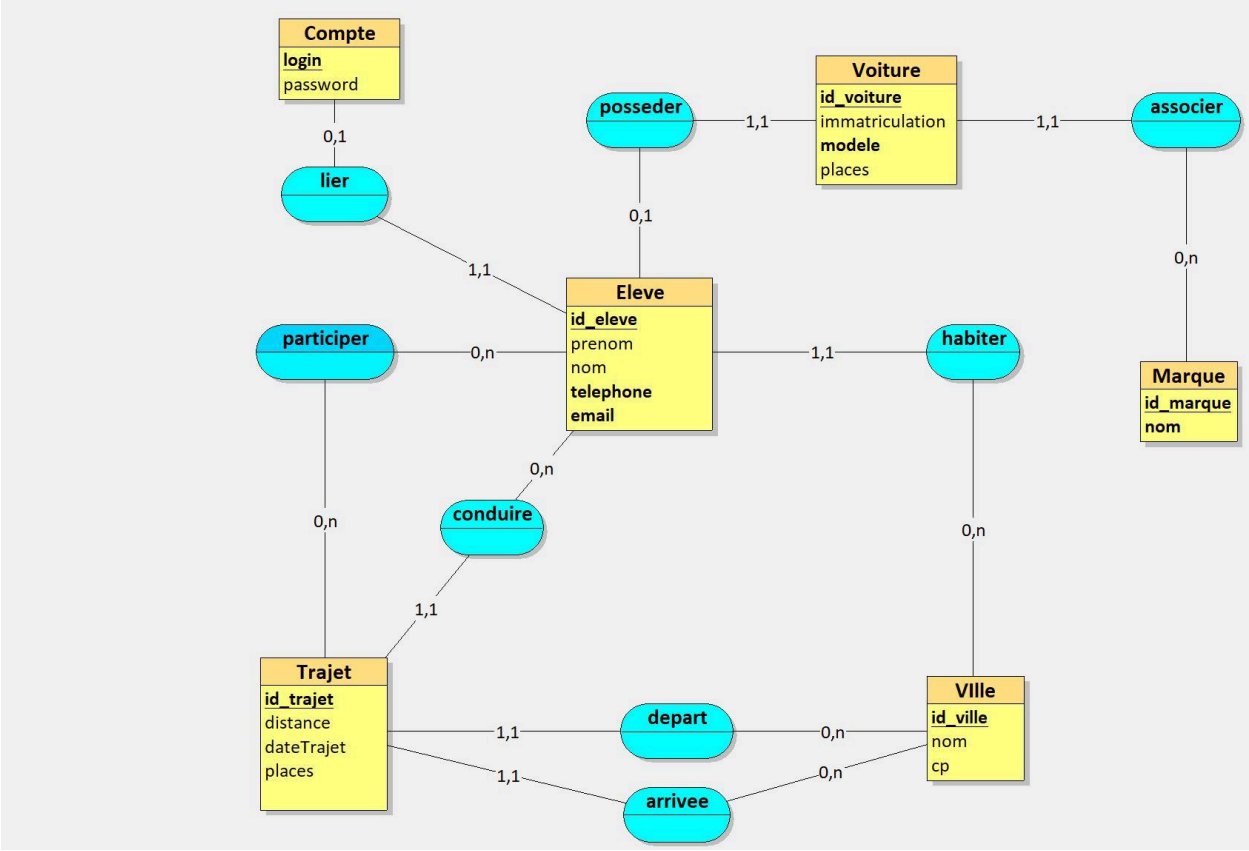


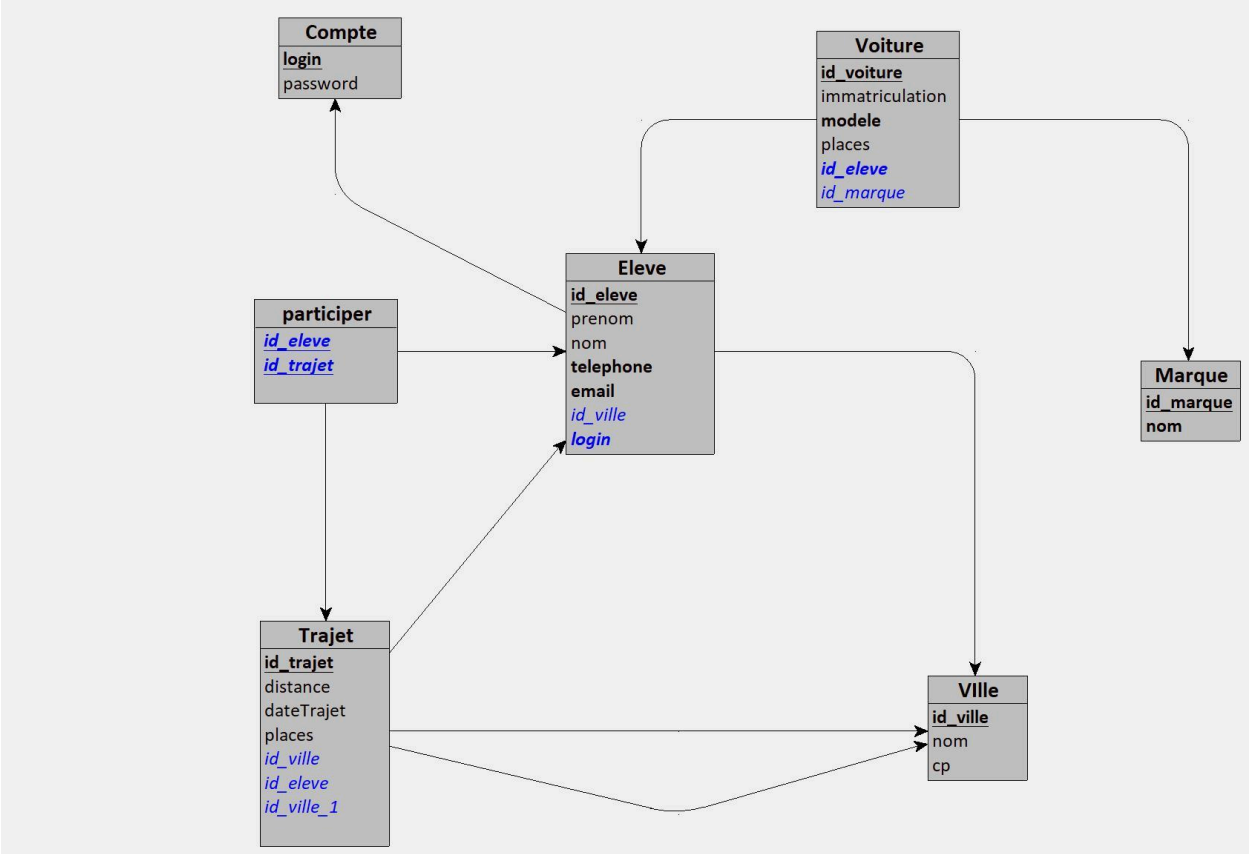
Diagramme de Classe - controlleurs



MCD


Compte
login
password
0,1
lier
1,1
posseder
1,1
Voiture
id_voiture
immatriculation
modele
places
1,1
associer
0,1
0,n
Eleve
id_eleve
prenom
nom
telephone
email
participer
0,n
1,1
habiter
Marque
id_marque
nom
0,n
conduire
0,n
0,n
1,1
Trajet
id_trajet
distance
dateTrajet
places
1,1
depart
0,n
VIlle
id_ville
nom
cp
1,1
arrivee
0,n

MLD


Compte
login
password
Voiture
id_voiture
immatriculation
modele
places
id_eleve
id_marque
Eleve
id_eleve
prenom
nom
telephone
email
id_ville
login
participer
id_eleve
id_trajet
Marque
id_marque
nom
Trajet
id_trajet
distance
dateTrajet
places
id_ville
id_eleve
id_ville_1
VIlle
id_ville
nom
cp

Configuration de l'application

La configuration de l'application ApiCovoiturage est gérée à travers plusieurs fichiers situés dans le répertoire config/ du projet.

Configuration de l'API

La configuration de l'API est définie dans le fichier config/packages/nelmio_api_doc.yaml. Ce fichier contient des informations sur l'API, y compris son titre, sa description, et sa version. Il définit également les zones de l'API qui doivent être documentées. Actuellement, toutes les routes sous /api sont documentées, à l'exception de /api/doc.

```
nelmio_api_doc:
   documentation:
       info:
           title: ApiCovoiturage
           description: API over HTTP
           version: 1.0.0

   areas: # to filter documented areas
       path_patterns:
           - ^/api(?!/doc$) # Accepts routes under /api except
/api/doc
```

Configuration de NelmioApiDoc

La configuration de NelmioApiDoc est gérée dans le fichier AppKernel.php. Ce fichier enregistre le bundle NelmioApiDoc dans l'application.

```
public function registerBundles(): iterable
{
    $bundles = [
        // ...

        new NelmioApiDocBundle(),
    ];

    return $bundles;
}
```

Configuration des Bundles

La configuration des bundles utilisés par l'application est définie dans le fichier config/bundles.php. Ce fichier retourne un tableau associatif où les clés sont les noms des classes des bundles et les valeurs sont les environnements dans lesquels ces bundles sont activés.

```
return [
    Symfony\Bundle\FrameworkBundle\FrameworkBundle::class => ['all' => true],
    Symfony\Bundle\SecurityBundle\SecurityBundle::class => ['all' => true],

Lexik\Bundle\JWTAuthenticationBundle\LexikJWTAuthenticationBundle::class => ['all' => true],
    Symfony\Bundle\MakerBundle\MakerBundle::class => ['dev' => true],
```

```
    Doctrine\Bundle\DoctrineBundle\DoctrineBundle::class => ['all' =>
true],
    Doctrine\Bundle\MigrationsBundle\DoctrineMigrationsBundle::class
=> ['all' => true],
    Nelmio\ApiDocBundle\NelmioApiDocBundle::class => ['all' => true],
    Symfony\Bundle\TwigBundle\TwigBundle::class => ['all' => true],
    Twig\Extra\TwigExtraBundle\TwigExtraBundle::class => ['all' =>
true],
];
```

Fonctionnalités de l'application

ApiCovoiturage offre une variété de fonctionnalités pour faciliter l'interaction avec l'API. Voici une liste des principales fonctionnalités de l'application :

1. Authentification : L'application utilise le bundle LexikJWTAuthenticationBundle pour gérer l'authentification des utilisateurs. Les utilisateurs peuvent se connecter en envoyant une requête POST à /api/login avec leurs identifiants. En cas de succès, ils reçoivent un token JWT qu'ils peuvent utiliser pour s'authentifier sur les routes protégées.
2. Gestion des utilisateurs : L'application permet la création, la lecture, la mise à jour et la suppression des utilisateurs. Ces opérations sont réalisées via des requêtes HTTP aux routes appropriées.
3. Gestion des Voitures : L'application permet la gestion des voitures. Les utilisateurs peuvent ajouter, modifier, et supprimer leur voiture. Chaque voiture a des attributs tels que la marque, le modèle, et le nombre de places disponibles.
4. Gestion des Trajets : L'application offre également la possibilité de gérer des trajets. Les utilisateurs peuvent créer des trajets, les modifier, les visualiser et les supprimer. Un trajet comprend des informations telles que le point de départ, la destination, la date et l'heure du trajet, et la voiture utilisée pour le trajet.
5. Gestion des Inscriptions aux Trajets par les Élèves : Enfin, l'application permet aux élèves de s'inscrire à des trajets. Les élèves peuvent visualiser les trajets disponibles et s'inscrire à ceux qui correspondent à leurs besoins. Dans les sections suivantes de ce document, nous détaillerons davantage le guide d'utilisation de l'API, les tests, le déploiement de l'application, et la maintenance et le support.
6. Documentation de l'API : L'application utilise le bundle NelmioApiDoc pour fournir une documentation interactive de l'API. Les utilisateurs peuvent accéder à cette documentation en visitant /api/doc.

7. Gestion des erreurs : L'application gère les erreurs de manière robuste, en renvoyant des réponses HTTP appropriées en cas d'erreur. Par exemple, si un utilisateur tente d'accéder à une route protégée sans s'authentifier, l'application renvoie une réponse avec le code de statut 401 (Non autorisé).
8. Sécurité : L'application utilise plusieurs mesures de sécurité pour protéger les données des utilisateurs. Par exemple, toutes les données sensibles sont cryptées avant d'être stockées, et l'application utilise des tokens JWT pour l'authentification, ce qui évite de stocker les mots de passe des utilisateurs.

Guide d'utilisation de l'API

1. Authentification

   L'authentification est gérée par le bundle LexikJWTAuthenticationBundle. Pour vous authentifier, vous devez envoyer une requête POST à /api/login_check avec vos identifiants. En cas de succès, vous recevrez un token JWT que vous devrez inclure dans l'en-tête Authorization de vos requêtes ultérieures pour accéder aux routes protégées. Exemple de requête d'authentification :

   ```
   127.0.0.1:8000/api/login/{login},{password}
   ```

2. Routes disponibles

   Voici quelques-unes des routes disponibles dans notre API :

   /api/login: Utilisée pour l'authentification des utilisateurs.

   /api/register : Utilisée pour la registration a l’application

   /api/listPersonne : Utilisée pour la visualisation des eleves

Tests

Les tests de notre application sont effectués à l'aide de SonarQube et GitLab. SonarQube est un outil d'analyse statique de code qui nous aide à maintenir la qualité du code en détectant les bugs, les vulnérabilités et les problèmes de code. GitLab, d'autre part, est utilisé pour l'intégration continue (CI) et le déploiement continu (CD), ce qui nous permet de vérifier que notre application fonctionne correctement à chaque commit.

SonarQube

Nous utilisons SonarQube pour analyser notre code source et détecter les problèmes potentiels. SonarQube vérifie la qualité du code en se basant sur des règles prédéfinies pour chaque langage de programmation. Il fournit également des métriques sur la couverture de code, la complexité cyclomatique, les duplications de code, et plus encore. La configuration de SonarQube est définie dans le fichier .gitlab-ci.yml de notre projet. Nous avons une étape spécifique sonarqube-check qui exécute l'analyse SonarQube.

GitLab

GitLab est utilisé pour l'intégration continue (CI) de notre application. À chaque commit, GitLab exécute un pipeline qui comprend l'analyse SonarQube et d'autres tâches, comme la construction de l'application et l'exécution des tests. Nous avons également une étape sonarqube-vulnerability-report dans notre pipeline GitLab qui génère un rapport de vulnérabilités à partir des résultats de l'analyse SonarQube. Dans les sections suivantes de ce document, nous détaillerons davantage le déploiement de l'application, et la maintenance et le support.

Conclusion

Application ApiCovoiturage est une solution robuste et flexible pour la gestion du covoiturage entre les élèves. Grâce à une API bien conçue et sécurisée, les utilisateurs peuvent facilement gérer les voitures, les trajets et les inscriptions aux trajets. L'application est construite en utilisant le langage de programmation PHP et le framework Symfony, avec le support de plusieurs bundles pour fournir des fonctionnalités spécifiques. L'authentification est gérée par le bundle LexikJWTAuthenticationBundle, la gestion de la base de données est facilitée par le bundle Doctrine, et la documentation de l'API est fournie par le bundle NelmioApiDoc. Les tests de l'application sont effectués à l'aide de SonarQube et GitLab, garantissant ainsi la qualité du code et le bon fonctionnement de l'application. Le déploiement de l'application est également automatisé grâce à GitLab. En somme, ApiCovoiturage est une solution complète pour la gestion du covoiturage entre les élèves, offrant une interface facile à utiliser et une architecture solide et sécurisée.