

Table of Contents

<i>Réflexions initiales technologiques</i>	1
Choix du Framework : Symfony.....	1
Composants utilisés :	1
Gestion des données.....	1
Structure du code	1
Cette stratégie a permis de :	1
<i>Configuration de l'environnement de travail</i>	2
Dépôt GitHub.....	2
Outils utilisés en local	2
Déploiement en ligne.....	2
Sécurité et bonnes pratiques	2
Quelques exécutions pour setup l'environnement de travail et tester la connexion à la base de données	2
<i>Modèle conceptuel de données avec dbdiagram.io</i>	6
<i>Diagramme d'utilisation avec PlantUML</i>	6
<i>Documentation du déploiement de l'application</i>	9
Démarche globale.....	9
Étapes détaillées.....	9
Particularités.....	10

Documentation technique EcoRide

Réflexions initiales technologiques

Choix du Framework : Symfony

Symfony a été choisi pour sa robustesse, sa structure MVC claire, sa sécurité native et sa documentation complète. Il permet de gérer facilement les entités métier avec Doctrine ORM.

Composants utilisés :

- Symfony avec composer
- Doctrine pour la gestion des entités et des relations MySQL
- Twig pour les templates HTML côté front
- Bootstrap 5 + CSS personnalisé (ecoride.css) pour une interface sobre, accessible et écoresponsable
- Git / GitHub avec workflow dev > main structuré et pushes vers Heroku

Gestion des données

- Le choix pédagogique a été de ne pas utiliser DataFixtures au départ
- Toutes les données ont été insérées via requêtes SQL manuelles, afin de pratiquer le langage MySQL (CREATE, INSERT, ALTER...) avec JawsDB ou localement

Structure du code

- Routes et contrôleurs Symfony respectent les US pas à pas
- Une méthode progressive de développement a été appliquée, avec validation de chaque étape (CRUDs, sécurités, vues)
- L'utilisateur a validé les entités en respectant son propre MCD logique, partagé dans public/docs/

Cette stratégie a permis de :

- Prendre en main le framework Symfony
- Apprendre le SQL en parallèle
- Comprendre la structure MVC et le routage
- Utiliser Doctrine et composant MakerBundle
- S'exercer à la construction d'un projet complet, réaliste et cohérent

Configuration de l'environnement de travail

Dépôt GitHub

<https://github.com/tletranvn/ecoride>

Outils utilisés en local

- MacBook Air / macOS
- Visual Studio Code pour le développement
- Terminal zsh pour les commandes Git, Symfony
- MySQL pour les données relationnelles
- Sequel Ace pour gérer la base locale et pour l'accès à JawsDB (production)

Déploiement en ligne

- Hébergement via Heroku, avec push depuis Git
- Base de données en ligne via JawsDB (MySQL)
- Connexion entre Symfony et JawsDB via variable d'environnement DATABASE_URL

Sécurité et bonnes pratiques

- .env.local ignoré par Git
- Séparation claire entre branche dev (développement) et main (prod)
- Historique des merges et tests de fonctionnalités respectés avant mise en prod

Quelques exécutions pour setup l'environnement de travail et tester la connexion à la base de données

1. Créer un nouveau projet Symfony sur Desktop

Symfony new ecoride --webapp

2. Créer un fichier .env.local avec la bonne connexion MySQL (locale ou JawsDB)

3. Démarrer le serveur local Symfony :

symfony server:start

```

[tranletran@khacs-macbook-air-1 Desktop % cd ~/Desktop
[tranletran@khacs-macbook-air-1 Desktop % symfony new ecoride --webapp
* Creating a new Symfony project with Composer
* Setting up the project under Git version control
  (running git init /Users/tranletran/Desktop/ecoride)

[OK] Your project is now ready in /Users/tranletran/Desktop/ecoride

[tranletran@khacs-macbook-air-1 Desktop % cd ~/Desktop/ecoride
[tranletran@khacs-macbook-air-1 ecoride % symfony server:start

[WARNING] run "symfony server:ca:install" first if you want to run the web se
rver with TLS support, or use "--p12" or "--no-tls" to avoid this w
arning

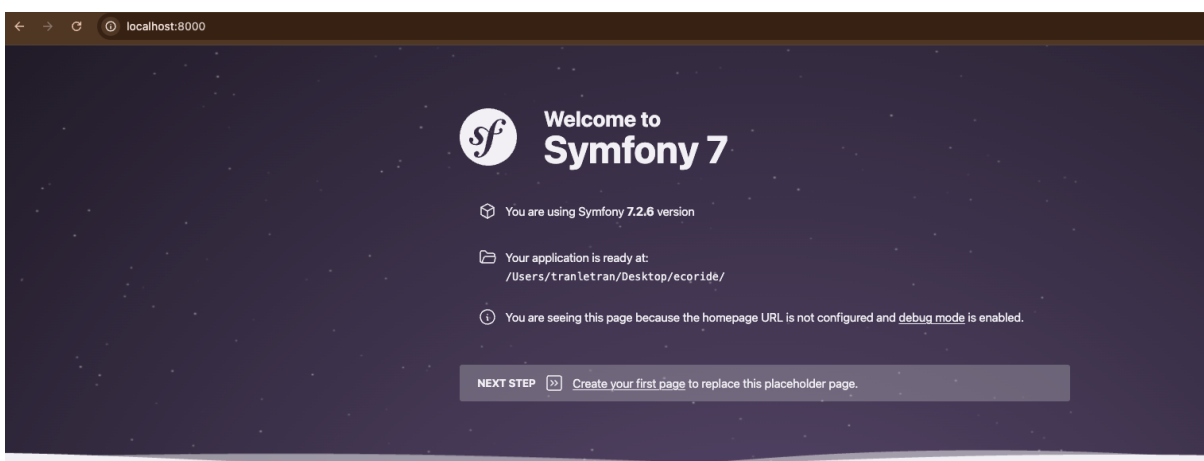
Following Web Server log file (/Users/tranletran/.symfony5/log/0128a94287c49ea5c0f8039f2b2900d6d133d1ce.1o
g)
Following PHP-FPM log file (/Users/tranletran/.symfony5/log/0128a94287c49ea5c0f8039f2b2900d6d133d1ce/53fb8
ec204547646acb3461995e4da5a54cc7575.log)

[WARNING] The local web server is optimized for local development and MUST ne
ver be used in a production setup.

[WARNING] Please note that the Symfony CLI only listens on 127.0.0.1 by defau
lt since version 5.10.3.
        You can use the --allow-all-ip or --listen-ip flags to ch
ange this behavior.

[OK] Web server listening
The Web server is using PHP FPM 8.4.7
http://127.0.0.1:8000

```



4. Créer une Base de données (exemple)

4.1 Créer l'entité User, configurer l'enregistrement d'un nouvel utilisateur avec un formulaire d'inscription.

```

● tranletran@khacs-macbook-air-1 ecoride % php bin/console make:user

The name of the security user class (e.g. User) [User]:
>

Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
>

Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:
>

Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed by some other system (e.g. a single sign-on server).
Does this app need to hash/check user passwords? (yes/no) [yes]:
>

created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml

Success!

Next Steps:
- Review your new App\Entity\User class.
- Use make:entity to add more fields to your User entity and then run make:migration.
- Create a way to authenticate! See https://symfony.com/doc/current/security.html

```

```

Your entity already exists! So let's add some new fields!

New property name (press <return> to stop adding fields):
> pseudo

Field type (enter ? to see all types) [string]:
>

Field length [255]:
> 50

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/User.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> credits

Field type (enter ? to see all types) [string]:
> integer

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/User.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> role

Field type (enter ? to see all types) [string]:
>

Field length [255]:
> 50

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/User.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> createdAt

Field type (enter ? to see all types) [datetime_immutable]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/User.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> updatedAt

Field type (enter ? to see all types) [datetime_immutable]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
> yes

updated: src/Entity/User.php

```

4.2 Migration de BDD :

Php bin/console doctrine :database :create

Php bin/console make :migrate

```

• tranletran@khacs-macbook-air-1 ecoride % php bin/console d:d:c
Created database `ecoride` for connection named default
• tranletran@khacs-macbook-air-1 ecoride % php bin/console make:migration
created: migrations/Version20250526043313.php

Success!

Review the new migration then run it with php bin/console doctrine:migrations:migrate
See https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html

```

Php bin/console doctrine :migrations :migrate

```

• tranletran@khacs-macbook-air-1 ecoride % php bin/console doctrine:migrations:migrate
WARNING! You are about to execute a migration in database "ecoride" that could result in schema changes and data loss. Are you sure you wish to continue? (yes/no) [yes]:
>
[notice] Migrating up to DoctrineMigrations\Version20250526043313
[notice] finished in 43.1ms, used 22M memory, 1 migrations executed, 2 sql queries
[OK] Successfully migrated to version: DoctrineMigrations\Version20250526043313

```

5. Test la connexion :

```

• tranletran@khacs-macbook-air-1 ecoride % mysql -u tenten -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 138
Server version: 9.3.0 Homebrew

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| data     |
| information_schema |
| laravel  |
| mysql    |
| performance_schema |
| restaurant |
| sys      |
| tenten_data |
+-----+
8 rows in set (0,038 sec)

mysql> exit;
Bye

```

The image shows a dark-themed window for configuring a database connection. At the top, there are three tabs: 'TCP/IP' (selected), 'Socket', and 'SSH'. Below the tabs, the configuration fields are as follows:

- Name:** optional
- Host:** localhost
- Username:** tenten
- Password:** masked with 12 dots
- Database:** ecoride
- Port:** 3306 (highlighted with a blue border)
- Time Zone:** Use Server Time Zone (dropdown menu)

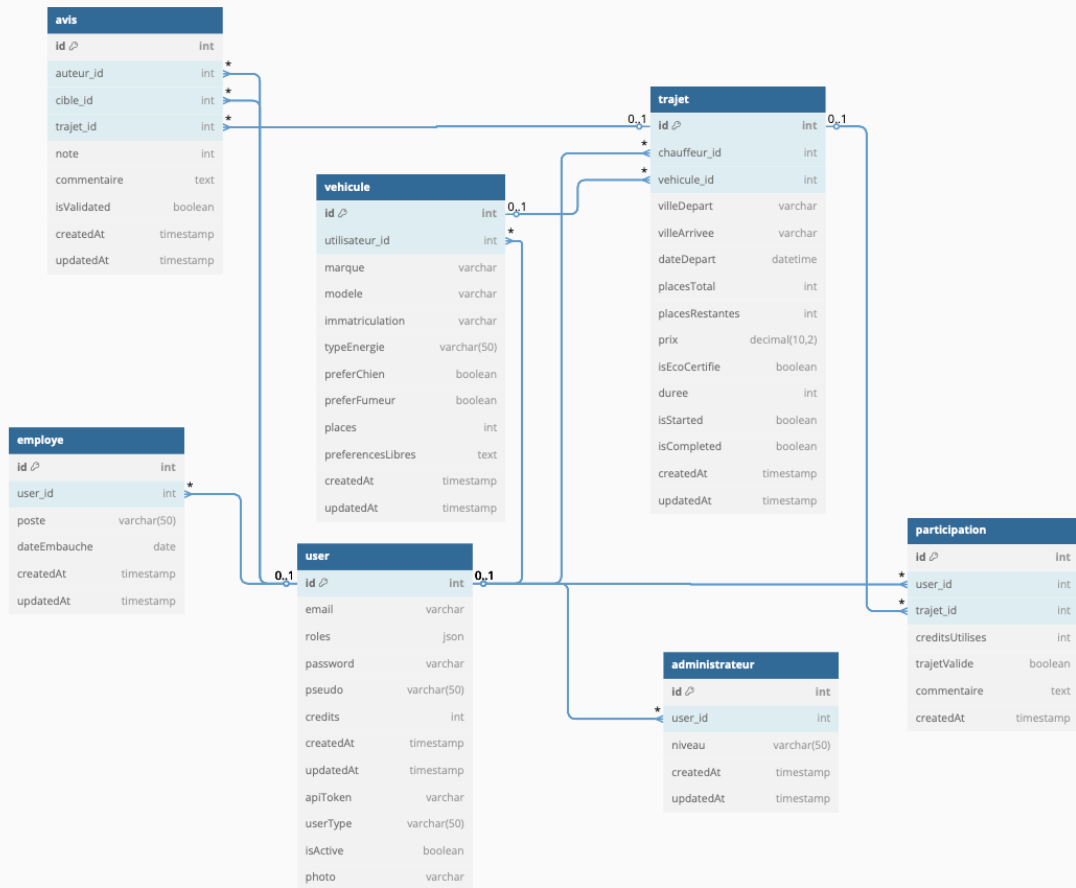
Below these fields are three unchecked checkboxes:

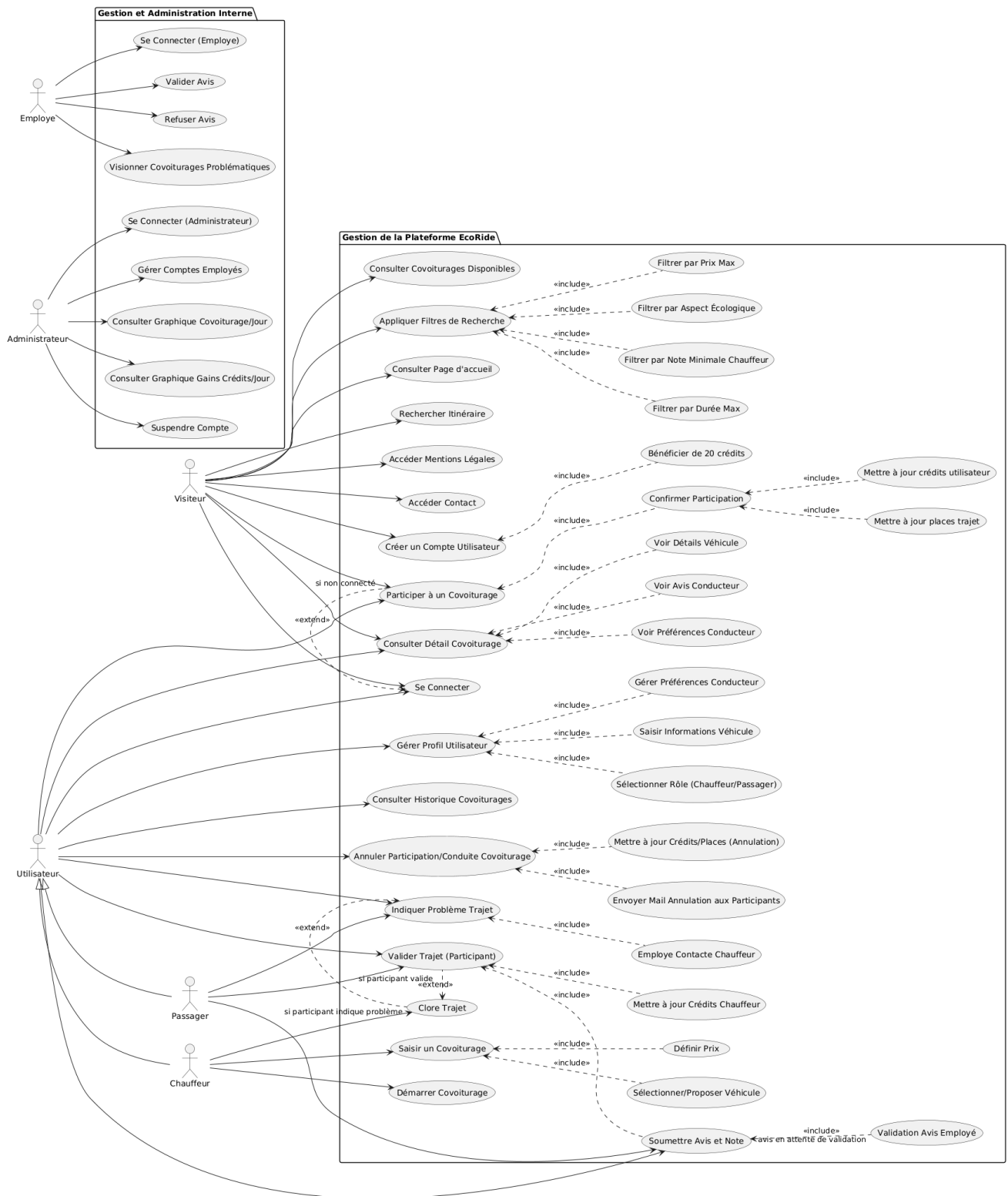
- ☐ Allow LOCAL_DATA_INFILE (insecure)
- ☐ Enable Cleartext plugin (insecure)
- ☐ Require SSL

At the bottom of the window, there is a row of buttons: a help button (question mark icon), a blue 'Connect' button, and three grey buttons: 'Add to Favorites', 'Save changes', and 'Test connection'.

Modèle conceptuel de données avec dbdiagram.io

Diagramme d'utilisation avec PlantUML





Documentation du déploiement de l'application

Démarche globale

Le déploiement de l'application EcoRide s'est fait en suivant une méthode progressive : développement local validé par étapes (chaque US est fait et testé et validé), versionnage propre sur GitHub depuis Vscodé, puis mise en ligne via Heroku.

Étapes détaillées

1. Création d'un dépôt GitHub public
2. Utilisation des branches `dev` pour le développement et `main` pour la production
3. Création un compte Heroku et connexion depuis vscode avec `heroku create`
4. Test local du projet Symfony (`symfony server:start`)
5. Merge de la branche `dev` vers `main` une fois stable
6. Push de `main` vers GitHub :

```
git push origin main
```

7. Push de `main` vers Heroku :

```
git push heroku main
```

8. Configuration `framework.yaml` et `.env.local`

```
TRUSTED_HOSTS=localhost,127.0.0.1
TRUSTED_PROXIES=127.0.0.1,REMOTE_ADDR
```

```
framework:
    secret: '%env(APP_SECRET)%'

    session:
        handler_id: null
        cookie_secure: auto

    trusted_proxies: '%env(TRUSTED_PROXIES)%'
    trusted_hosts: '%env(TRUSTED_HOSTS)%'
```

9. Connexion de l'application Symfony à JawsDB (MySQL)
10. Mise à jour manuelle de la base JawsDB si besoin

```

tranletran@khacs-macbook-air-1 ecoride % git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 6 commits.
(use "git push" to publish your local commits)
tranletran@khacs-macbook-air-1 ecoride % git merge dev
Merge made by the 'ort' strategy.
 src/Controller/ContactController.php | 18 ++++++++
 templates/contact/contact.html.twig | 11 +++++
 2 files changed, 29 insertions(+)
 create mode 100644 src/Controller/ContactController.php
 create mode 100644 templates/contact/contact.html.twig
● tranletran@khacs-macbook-air-1 ecoride % git push heroku main

```

Particularités

- Aucun usage de `doctrine:migrations` : les modifications SQL ont été faites à la main pour compréhension
- Les fichiers `.sql` nécessaires sont stockés dans `public/docs/`
- Tests de fonctionnement réalisés via Heroku live (formulaires, sessions, requêtes)

Cette démarche permet un contrôle total sur chaque étape du déploiement, et garantit une compréhension approfondie du cycle de vie complet d'une application web Symfony.