Rapport de stage des 7 semaines

Introduction du rapport de stage :

Dans le cadre de ma deuxième année de BTS SIO (Services Informatiques aux Organisations), spécialité SLAM (Solutions Logicielles et Applications Métier) , j'ai eu l'opportunité d'effectuer un stage au sein de l'entreprise Chevalierb.fr .

Ce stage, d'une durée de 7 semaines, avait pour objectif de mettre en pratique les compétences acquises au cours de ma formation, notamment en développement d'applications, gestion de bases de données et conception de solutions logicielles adaptées aux besoins d'une entreprise.

Au sein de « Chevalierb.fr », j'ai travaillé sur un projet Symfony visant à optimiser la gestion des événements et des utilisateurs, impliquant des fonctionnalités comme l'authentification, la gestion des rôles et la participation aux événements. Ce rapport détaillera les différentes étapes de mon travail, les technologies utilisées ainsi que les compétences développées tout au long de cette expérience.

Structure du projet :

- **Fichiers de configuration :**
  - .env, .env.dev, .env.test → Contiennent les variables d'environnement.
  - composer.json, composer.lock → Définissent les dépendances du projet Symfony.
  - symfony.lock → Enregistre les versions des paquets installés.
  - phpunit.xml.dist → Configuration des tests unitaires avec PHPUnit.
  - importmap.php → Probablement utilisé pour gérer les imports JS (si Symfony UX est utilisé).
  - compose.yaml, compose.override.yaml → Indiquent une potentielle utilisation de Docker.
- **Documentation :**
  - README.md → Contient sûrement des instructions sur l'utilisation du projet.
  - looping 1.pdf → Un fichier PDF, à voir son contenu.
- **Dossiers Git :**

- .git/ → Indique que le projet est sous gestion de version avec Git.

📌 Analyse des dépendances Symfony :

✅ **Dépendances principales :**

- **Doctrine ORM & DBAL** → Gestion de la base de données avec Doctrine.
- **Symfony Form & Validator** → Gestion des formulaires et validation des données.
- **Symfony Security Bundle** → Gestion de l'authentification et de l'autorisation.
- **Mailer & Notifier** → Envoi d'e-mails et notifications.
- **Twig & Asset Mapper** → Utilisation de Twig pour les templates et gestion des assets.
- **Symfony UX (Stimulus & Turbo)** → Amélioration de l'expérience utilisateur avec des interactions front-end dynamiques.

Dépendances de développement :

- **PHPUnit** → Pour les tests unitaires.
- **Symfony Maker Bundle** → Génération rapide de code (entités, contrôleurs, etc.).
- **Web Profiler Bundle** → Outil de debugging et monitoring des requêtes.

Analyse du code source (src/)

Contrôleurs (Controller/)

Ces fichiers définissent les différentes routes et logiques métier de l'application :

- ConnexionController.php → Gestion de l'authentification des utilisateurs.
- EntrepriseController.php → Gestion des entreprises.
- EvenementController.php → Gestion des événements.
- HomeController.php → Page d'accueil.
- InscriptionController.php → Gestion des inscriptions.
- InvitesController.php & InvitesFormController.php → Gestion des invitations.
- UserController.php → Gestion des utilisateurs.

🔜 **L'application gère des utilisateurs, des entreprises et des événements, avec un système d'invitation.**

Entités (Entity/)

Ces fichiers représentent les tables de la base de données :

- Entreprise.php → Modélisation des entreprises.
- EvenementOR.php → Modélisation des événements.
- Roles.php → Gestion des rôles utilisateurs.
- User.php → Gestion des utilisateurs.

Analyse de la base de données

1.Tableaux principaux

entreprise(Gestion des entreprises)

- Contenu des informations sur les entreprises : nom_entreprise, adresse, contact, etc.
- Liée à la table uservia entreprise_id, ce qui signifie qu'un utilisateur appartient à une entreprise.

user(Gestion des utilisateurs)

- Stocke les informations des utilisateurs : email, password, nom, prenom, telephone, etc.
- Liée à entreprisevia entreprise_id(un utilisateur appartient à une entreprise).
- Champ invited_by_idpour suivre qui a invité l'utilisateur.
- is_presentet is_present_mangerindiquer leur présence à des événements et s'ils vont manger.

roles& user_roles(Gestion des rôles)

- rolesstocke les rôles disponibles (id, name).
- user_rolesest une table pivot qui associe des rôles aux utilisateurs (user_id, roles_id).

---

2 Gestion des événements

evenement_or(Tableau des événements)

- Stockez les détails d'un événement : date_evenement, heure_evenement, type, adresse_postale, etc.
- is_presentet is_present_mangerpour indiquer la présence des participants.

evenement_participants(Participants aux événements)

- Tableau de relation entre useret evenement_or:

- evenement_or_id: l'événement concerné.
- user_id: l'utilisateur participant.

➡ **Permet de savoir quels utilisateurs assistent à quels événements.**

Autres tables

messenger_messages(Système de messages)

- Stockez les messages envoyés via Symfony Messenger (utile pour l'envoi asynchrone de notifications).
- Gère le traitement différé avec created_at, available_at, delivered_at.

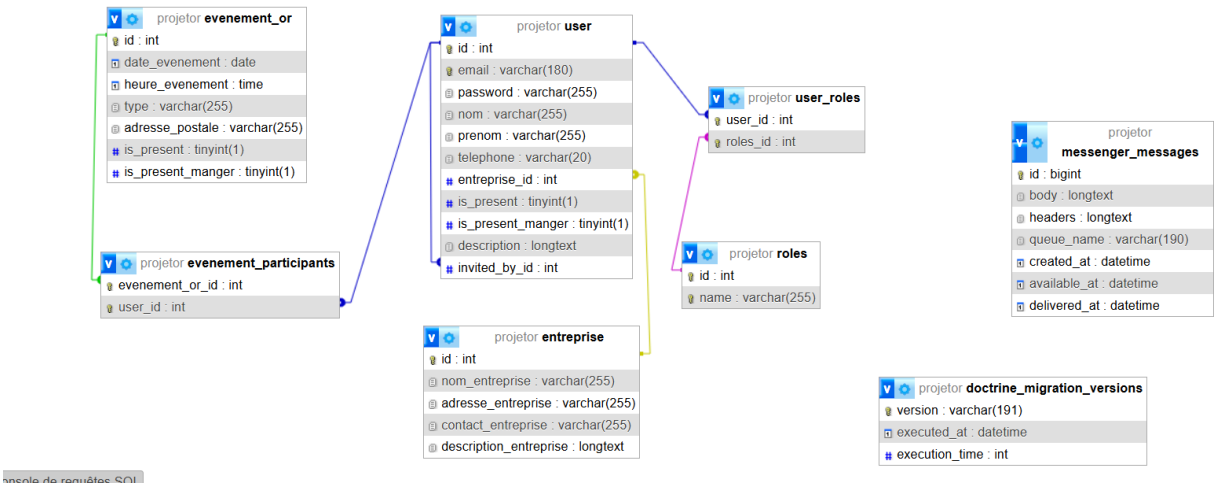
doctrine_migration_versions(Gestion des migrations)

- Suivi des versions de migration de la base de données.

Fonctionnalités possibles basées sur ce BDD

✅ **Gestion des entreprises** → Lier les utilisateurs à une entreprise.
✅ **Gestion des utilisateurs et rôles** → Authentification et autorisation.
✅ **Système d'événements** → Organisation d'événements et suivi des participants.
✅ **Système d'invitations** → Suivi des invités via invited_by_id.
✅ **Notifications/messages** → Utilisation de Symfony Messenger pour la communication.

➡ **La base de données est structurée autour des utilisateurs, des rôles, des entreprises et des événements.**



Repositories (Repository/)

Chaque entité a son repository dédié pour interagir avec la base de données :

- EntrepriseRepository.php
- EvenementORRepository.php
- RolesRepository.php
- UserRepository.php

Sécurité (Security/)

- UserAuthenticator.php → Définit la logique d'authentification des utilisateurs.

Autres fichiers

- EventListener/SessionInitializerListener.php → Un écouteur d'événements pour initialiser certaines sessions.
- Form/InviteType.php → Définition d'un formulaire pour gérer les invitations.

Résumé global du projet

✅ **Type d'application** : Une plateforme de gestion des entreprises et événements avec utilisateurs, rôles et invitations.
✅ **Technologies clés** : Symfony 7.2, Doctrine ORM, Twig, Formulaires, Sécurité (authentification), UX (Stimulus, Turbo).
✅ **Fonctionnalités principales** :

- Authentification et rôles utilisateurs.
- Gestion des entreprises et événements.
- Système d'invitations et d'inscription.
- Sécurité et interaction dynamique avec Symfony UX.