

Mini-Projet – Visites Tuteurs

Durée : ~5h – Backend & Frontend Symfony + API Platform

Objectif : Réaliser une micro-application permettant à un tuteur de gérer ses étudiants et leurs visites.

1. Contexte du projet

Dans la continuité du TP précédent sur API Platform, vous allez créer une application destinée aux tuteurs qui suivent les étudiants en alternance ou en stage.

Cette application permettra à un tuteur :

- de **se connecter**,
- de **se déconnecter**,
- de **gérer les étudiants** qu'il suit,
- de **planifier des visites**,
- de **rédiger un compte-rendu**,
- de **visualiser l'historique**.

⚠ Les notions abordées dans ce mini-projet seront évaluées au Contrôle Terminal (CT).

2. Modèle de données attendu

2.1. Entité Tuteur

Vous l'avez déjà créée lors du TP précédent.

Elle représente l'utilisateur de votre application.

Champs obligatoires :

- nom
- prenom
- email
- telephone

Relations :

- Un tuteur → plusieurs étudiants
- Un tuteur → plusieurs visites

2.1 bis – Crédation d'un tuteur via l'API (obligatoire)

Avant d'utiliser l'application, vous devez **créer un ou plusieurs tuteurs via l'API Platform**.

Via le Swagger de API Platform :

1. Aller sur **POST /api/tuteurs**
2. Créer un tuteur avec au minimum :

```
{  
    "nom": "Durand",  
    "prenom": "Alice",  
    "email": "alice.durand@example.com",  
    "telephone": "0601020304"  
}
```

3. Ce tuteur servira pour la connexion.

Cette étape est obligatoire : vous devez pouvoir créer un tuteur à tout moment via votre API REST.

2.2. Entité Etudiant

Créer l'entité : `symfony console make:entity Etudiant`

Champs :

- nom
- prénom
- formation

Relation :

- tuteur : ManyToOne (obligatoire)

Exposez l'entité avec API Platform.

2.3. Entité Visite

Champs :

- date (datetime_immutable)
- commentaire (string)
- compteRendu (text, nullable)

- statut (string : “prévue”, “réalisée”, “annulée”)
- tuteur : ManyToOne (obligatoire)
- etudiant : ManyToOne (obligatoire)

Ajoutez des contraintes de validation.

3. Connexion et déconnexion du tuteur

Vous allez mettre en place un **login simplifié**, sans mot de passe, pour comprendre le fonctionnement d'une session Symfony.

3.1. Page /login (obligatoire)

Fonctionnalité :

- Formulaire Twig avec champs :
 - email du tuteur, et password
- À la soumission :
 - Recherche du tuteur en base via Doctrine
 - Si trouvé → stocker son id en session :

```
$session->set('tuteur_id', $tuteur->getId());
```

- Redirection vers /dashboard
- Sinon → message d'erreur

3.2. Déconnexion /logout (obligatoire)

Créer une route /logout qui :

- supprime l'id du tuteur en session : `$session->remove('tuteur_id');`
- redirige vers /login
- affiche un message de confirmation

Dans la barre de navigation, prévoir un bouton “**Déconnexion**” visible uniquement pour un utilisateur connecté.

4. Tableau de bord du tuteur (/dashboard)

Affiche :

- Les informations du tuteur connecté
- La liste des étudiants qu'il suit

- Les prochaines visites planifiées
- Un lien pour ajouter un étudiant

5. Gestion des étudiants (obligatoire)

5.1. Liste des étudiants

Route : /etudiants

Afficher :

- Nom – Prénom – Formation
- Boutons :
 - Voir visites
 - Ajouter une visite
 - Modifier
 - Supprimer (optionnel)

5.2. Ajouter un étudiant

Route : /etudiants/new

- Form Type : EtudiantType
 - Lier automatiquement l'étudiant au tuteur connecté :
- ```
$etudiant->setTuteur($tuteurConnecte);
```

### 5.3.Modifier un étudiant

Vérifier qu'il appartient au tuteur connecté.

## 6. Gestion des visites (obligatoire)

### 6.1. Liste des visites d'un étudiant

Route : /etudiants/{id}/visites

### 6.2. Ajouter une visite

Route : /etudiants/{id}/visites/new

Préremplir automatiquement :

```
$visite->setEtudiant($etudiant);
$visite->setTuteur($tuteurConnecte);
$visite->setStatut('prévue');
```

## 6.3. Modifier une visite

Route : /visites/{id}/edit

## 7. Compte-rendu de visite (obligatoire)

Route : /visites/{id}/compte-rendu

Formulaire contenant :

- champ compteRendu

Afficher aussi :

- l'étudiant concerné
- la date
- le commentaire initial
- le statut actuel

On doit pouvoir l'exporter en pdf.

## 8. Partie 5 — Fonctionnalités obligatoires

### 8.1. Champ statut dans Visite

Valeurs possibles : “prévue”, “réalisée”, “annulée”.

### 8.2. Filtre par statut dans la liste des visites

Formulaire GET avec un select :

- Toutes
- Prévue
- Réalisée
- Annulée

### 8.3. Tri des visites par date

Lien “Trier ↑” et “Trier ↓”

Utiliser un Repository personnalisé.

### 8.4. Mise en forme Bootstrap (obligatoire)

Ajoutez au minimum :

- Navbar contenant le bouton “Déconnexion”
- Mise en page propre (tableaux, formulaires, boutons)

## 9. Livrables

- Projet Symfony complet fonctionnel dans Docker
- Fonctionnalités backend & frontend
- Connexion/déconnexion
- CRUD Étudiants
- CRUD Visites
- Filtrage et tri
- Front minimal avec Bootstrap
- README contenant :
  - instructions d'installation
  - instructions de connexion
  - captures d'écran facultatives

## 10. Évaluation au Contrôle Terminal (CT)

Ce mini-projet sert de support aux notions qui seront évaluées au CT, notamment :

- Relations entre entités
- Migrations Doctrine
- API Platform (création de tuteurs obligatoire)
- Sessions Symfony (login/déconnexion)
- Twig + Form Types
- Bonne structuration du projet
- Filtre, tri, affichage des données
- Validation des entités
- Sécurité (cela sous entend que vous devez avoir implémenter un certain nombre de mécanismes vu en cours...)