

1. Présentation du projet

EduTest est une plateforme web de gestion et d'évaluation éducative développée en **Symfony 6**. Elle permet à trois profils d'utilisateurs d'interagir :

-  **Administrateur** → Gère les enseignants, étudiants et examens
-  **Enseignant** → Crée, affecte et corrige des examens
-  **Étudiant** → Passe les examens et visualise ses résultats

2. Technologies principales

| Outil | Usage |
|---------------------|---------------------------------|
| Symfony 6 | Framework backend PHP |
| Doctrine ORM | Gestion de la base de données |
| Twig | Moteur de templates |
| Bootstrap 5 | Mise en page et design |
| Chart.js | Graphiques du tableau de bord |
| mPDF | Exportation PDF |
| League CSV | Exportation CSV |
| MySQL | Base de données |
| Composer | Gestionnaire de dépendances PHP |

⌚ 3. Structure du projet

```
└── src/
    ├── Controller/
    ├── Entity/
    ├── Repository/
    ├── Security/
    └── Form/
    ├── templates/
    │   ├── admin/
    │   ├── teacher/
    │   ├── student/
    │   └── base.html.twig
    └── public/
        └── config/
            └── security.yaml
```

✿ 4. Entités principales

◆ User

Représente les enseignants, étudiants et administrateurs.

Champs clés :

```
php

#[ORM\Column(length: 180, unique: true)]
private ?string $email = null;

#[ORM\Column(type: 'json')]
private array $roles = [];

#[ORM\Column(length: 255)]
private ?string $fullName = null;

#[ORM\Column(options: ['default' => false])]
private bool $isApproved = false;
```

Relations :

```
php
```

```
#\[ORM\OneToMany(targetEntity: Assignment::class, mappedBy: 'student')]  
private Collection $assignments;
```

Rôle :

- Authentification via email et mot de passe
- Détermination du rôle (ROLE_ADMIN, ROLE_TEACHER, ROLE_STUDENT)
- Gestion de l'approbation par l'admin

◆ **Exam**

Représente un examen créé par un enseignant.

```
php
```

```
#\[ORM\Column(Length: 255)]  
private ?string $title = null;  
  
#\[ORM\Column(type: 'text', nullable: true)]  
private ?string $description = null;  
  
#\[ORM\Column]  
private ?int $durationMinutes = null;  
  
#\[ORM\Column(type: 'datetime')]  
private ?\DateTimeInterface $startAt = null;
```

Relations :

```
php
```

```
#\[ORM\OneToMany(targetEntity: Assignment::class, mappedBy: 'exam')]  
private Collection $assignments;
```

◆ Assignment

Relie un étudiant à un examen.

```
php
```

```
#[ORM\ManyToOne(targetEntity: User::class, inversedBy: 'assignments')]  
private ?User $student = null;  
  
#[ORM\ManyToOne(targetEntity: Exam::class, inversedBy: 'assignments')]  
private ?Exam $exam = null;  
  
#[ORM\Column(nullable: true)]  
private ?float $finalGrade = null;
```

Rôle :

- Stocke la note finale de l'étudiant
- Permet le suivi des soumissions

🧠 5. Repositories

UserRepository

Méthode personnalisée pour filtrer les rôles :

```
php
```

```
public function findByRole(string $role): array  
{  
    return $this->createQueryBuilder('u')  
        ->where('u.roles LIKE :role')  
        ->setParameter('role', '%' . $role . '%')  
        ->orderBy('u.fullName', 'ASC')  
        ->getQuery()  
        ->getResult();  
}
```

👉 Utilité :

- Récupérer tous les enseignants (ROLE_TEACHER)
- Récupérer tous les étudiants (ROLE_STUDENT)

6. Contrôleurs principaux

- ◆ AdminController
- Route de base : /admin

Rôles :

- Gère les utilisateurs
- Approuve, désactive ou supprime un utilisateur
- Affiche les statistiques globales
- Gère la création d'utilisateurs

Exemple de méthode :

```
php

#[Route('/dashboard', name: 'admin_dashboard')]
public function dashboard(EntityManagerInterface $em): Response
{
    $teachers = $userRepository->findByRole('ROLE_TEACHER');
    $students = $userRepository->findByRole('ROLE_STUDENT');
    $exams = $examRepository->findAll();

    return $this->render('admin/dashboard.html.twig', [
        'teachers' => $teachers,
        'students' => $students,
        'exams' => $exams
    ]);
}
```

-
- ◆ TeacherDashboardController

- Route : /teacher/dashboard

Fonctions :

- Affiche la liste des examens créés par l'enseignant
- Calcule les statistiques : nombre d'examens, taux de soumission, moyenne des notes

```
php
```

```
$query = $examRepository->createQueryBuilder('e')->orderBy('e.id', 'DESC')->getQuery();
$exams = $paginator->paginate($query, $request->query->getInt('page', 1), 10);
```

-
- ◆ AdminExportController

- Route : /admin/exam/{id}/export/pdf ou /csv

Objectif :

Exporter les résultats d'un examen au format **PDF** ou **CSV**.

```
php
```

```
$mpdf = new Mpdf(['default_font' => 'dejavusans']);  
$mpdf->WriteHTML($html);  
return new Response($mpdf->Output("exam_{$exam->getId()}.pdf", 'I'));
```

◆ AssignmentController

● /teacher/assignments

But :

- Lier un examen à un étudiant
- Modifier ou supprimer une affectation

⌚ 7. Routes principales

| Route | Rôle | Accès |
|-----------------------------|-------------------------------|--------------|
| /admin/dashboard | Vue globale admin | ROLE_ADMIN |
| /teacher/dashboard | Tableau enseignant | ROLE_TEACHER |
| /student/exams | Liste des examens disponibles | ROLE_STUDENT |
| /exam | Liste des examens (CRUD) | ROLE_TEACHER |
| /admin/exam/{id}/export/pdf | Export PDF | ROLE_ADMIN |
| /login / register | Authentification | Public |

💻 8. Templates Twig

base.html.twig

Contient :

- Navbar globale
- Footer
- Bloc body pour les pages enfants

admin/dashboard.html.twig

- Affiche les statistiques
- Gère les enseignants et étudiants
- Liste les examens
- Sidebar de navigation (Tableau de bord / Utilisateurs / Examens / Déconnexion)

teacher/dashboard.html.twig

- Liste des examens de l'enseignant
- Indique les statistiques de soumission
- Intègre des graphiques avec **Chart.js**

exam/index.html.twig

- Affiche tous les examens disponibles
 - Bouton “Retour” ajouté pour revenir en arrière
-

9. Sécurité (security.yaml)

yaml

```
access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }
    - { path: ^/teacher, roles: ROLE_TEACHER }
    - { path: ^/student, roles: ROLE_STUDENT }
    - { path: ^/exam, roles: IS_AUTHENTICATED_FULLY }
```

LoginSuccessHandler :

Redirection automatique après connexion :

```
if (in_array('ROLE_ADMIN', $roles)) return new RedirectResponse($this->urlGenerator->generate('admin_dashboard'));

if (in_array('ROLE_TEACHER', $roles)) return new RedirectResponse($this->urlGenerator->generate('teacher_dashboard'));

if (in_array('ROLE_STUDENT', $roles)) return new RedirectResponse($this->urlGenerator->generate('student_exams'));
```

10. Fonctionnalités principales (avec code)

Connexion / Inscription

twig

```
<form method="post">
    <input type="email" name="_username" required>
    <input type="password" name="_password" required>
</form>
```

→ Authentifie selon le rôle et redirige automatiquement.

Création d'un examen

php

```
$form = $this->createForm(ExamType::class, $exam);
$form->handleRequest($request);
```

Affectation d'un examen

php

```
$assignment = new Assignment();
$assignment->setExam($exam);
$assignment->setStudent($student);
$em->persist($assignment);
$em->flush();
```

Statistiques dynamiques

js

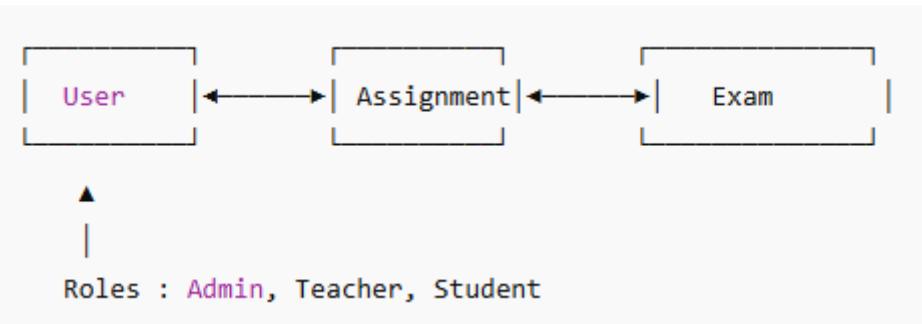
```
new Chart(ctx, {
    type: 'bar',
    data: { labels: ['Enseignants', 'Étudiants', 'Examens'],
            datasets: [{ data: [10, 50, 5] }] }
});
```

Export PDF

```
php
```

```
$html = $this->renderView('admin/export_pdf.html.twig', ['exam' => $exam]);  
$mpdf->WriteHTML($html);
```

11. Schéma simplifié de la base de données



12. Commandes utiles

```
bash
```

```
composer install  
php bin/console doctrine:database:create  
php bin/console doctrine:migrations:migrate  
symfony serve
```

13. Résumé des rôles

| Rôle | Accès |
|------------|---|
| Admin | Gère utilisateurs et examens |
| Enseignant | Crée, affecte, exporte examens |
| Étudiant | Passe les examens et consulte résultats |

Commandes Symfony utilisées pour le projet EduTest

1 — Initialisation du projet

Création du projet Symfony

```
bash
```

```
symfony new examapp --webapp
```

Crée un nouveau projet Symfony avec les composants essentiels : HTTP, Twig, Doctrine, Security, etc.

Démarrer le serveur local Symfony

```
bash
```

```
symfony serve
```

Lance le serveur local pour tester l'application sur <http://127.0.0.1:8000>.

2 — Configuration de la base de données

Vérifier la configuration .env

```
bash
```

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/examapp?serverVersion=8.0"
```

Définit la connexion à MySQL (nom de ta base : **examapp**).

Créer la base de données

```
bash
```

```
php bin/console doctrine:database:create
```

Génère une base de données vide à partir de la configuration .env

Générer les tables à partir des entités

```
bash
```

```
php bin/console make:migration  
php bin/console doctrine:migrations:migrate
```

- `make:migration` → crée un fichier SQL décrivant les changements
 - `migrate` → applique les changements à la base réelle
-

3 — Gestion des utilisateurs

Génération de l'entité User

```
bash
```

```
php bin/console make:user
```

Crée l'entité User avec le système d'authentification intégré à Symfony.
Elle implémente `UserInterface` et `PasswordAuthenticatedUserInterface`.

Génération du système de login

```
bash
```

```
php bin/console make:auth
```

Crée un contrôleur de sécurité (`SecurityController`) et les routes `/login` & `/logout`.

Hachage du mot de passe admin (optionnel)

```
bash
```

```
php bin/console security:hash-password
```

Permet de générer un mot de passe chiffré à insérer dans la BDD manuellement.

4 — Crédation des entités principales

Générer Exam

```
bash
```

```
php bin/console make:entity Exam
```

Entité représentant les examens (titre, durée, dates...).

Générer Assignment

```
bash
```

```
php bin/console make:entity Assignment
```

Entité qui relie un étudiant (User) à un Exam donné (et stocke la note).

Ajouter des relations entre entités

```
bash
```

```
php bin/console make:entity User  
# puis ajouter OneToMany et ManyToOne
```

Ajoute les relations Doctrine (User ↔ Assignment ↔ Exam).

5 — Génération automatique des formulaires

Formulaire pour Exam

```
bash
```

```
php bin/console make:form ExamType
```

Permet aux enseignants d'ajouter ou modifier un examen.

Formulaire pour Assignment

```
bash
```

```
php bin/console make:form AssignmentType
```

6 — Crédation des contrôleurs

Pour les enseignants

```
bash
```

```
php bin/console make:controller TeacherDashboardController
```

Affiche la liste des examens créés et leurs statistiques.

Pour les étudiants

```
bash
```

```
php bin/console make:controller StudentController
```

Permet aux étudiants de voir et passer leurs examens.

Pour l'administrateur

```
bash
```

```
php bin/console make:controller AdminController
```

Tableau de bord global : gestion des utilisateurs, approbations et examens.

Pour l'exportation des données

```
bash
```

```
php bin/console make:controller AdminExportController
```

Gère l'export **PDF** et **CSV** des résultats d'examens.

7 — Configuration de la sécurité

Vérification des rôles et redirection après connexion

Ajout du handler :

```
bash
```

```
php bin/console make:service LoginSuccessHandler
```

Service qui redirige l'utilisateur après login selon son rôle :

- Admin → /admin/dashboard
- Teacher → /teacher/dashboard
- Student → /student/exams

Nettoyer le cache après modification de la sécurité

```
bash
```

```
php bin/console cache:clear
```

À exécuter après chaque changement dans security.yaml ou services.yaml.

— Migrations et mise à jour

Ajouter un nouveau champ à une entité

```
bash
```

```
php bin/console make:entity User
# Ajouter un champ comme "isApproved"
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

— Outils de maintenance

Vérifier le schéma Doctrine

```
bash
```

```
php bin/console doctrine:schema:validate
```

Vérifie la cohérence entre les entités et la base.

Voir les routes disponibles

```
bash
```

```
php bin/console debug:router
```

Affiche toutes les routes générées dans le projet.

Tester le serveur

bash

```
php -S 127.0.0.1:8000 -t public
```

Méthode alternative pour lancer le serveur sans Symfony CLI.

10 — Installation des dépendances externes

PDF (mPDF)

bash

```
composer require mpdf/mpdf
```

Export CSV

bash

```
composer require league/csv
```

Graphiques (Chart.js)

Utilisé via CDN :

html

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

11 — Tests et déploiement

Charger les fixtures (optionnel)

bash

```
php bin/console doctrine:fixtures:load
```

Si tu veux initialiser des données fictives (enseignants, étudiants, examens).

1 2 — Résumé des commandes essentielles

| Étape | Commande | Description |
|----------------------|--|------------------------------|
| Créer le projet | <code>symfony new examapp --webapp</code> | Initialise le projet |
| Démarrer le serveur | <code>symfony serve</code> | Lancer l'app localement |
| Créer la BDD | <code>php bin/console doctrine:database:create</code> | Crée la base |
| Migration | <code>php bin/console doctrine:migrations:migrate</code> | Applique les schémas |
| Créer une entité | <code>php bin/console make:entity</code> | Crée une table |
| Créer un contrôleur | <code>php bin/console make:controller</code> | Crée un contrôleur |
| Créer un formulaire | <code>php bin/console make:form</code> | Crée un formulaire |
| Créer un utilisateur | <code>php bin/console make:user</code> | Système d'authentification |
| Cache | <code>php bin/console cache:clear</code> | Nettoie le cache |
| Routes | <code>php bin/console debug:router</code> | Liste des routes disponibles |

1 3 — Étapes finales pour relancer le projet

```
bash

git clone https://github.com/stshauke/edutest.git
cd edutest
composer install
php bin/console doctrine:database:create
php bin/console doctrine:migrations:migrate
symfony serve
```