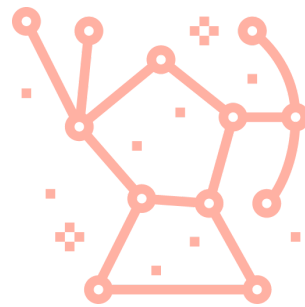


Projet Symfony de LE4



Symforion

Groupe composé de : Duriez Matthias, Ramzi Othmane, Capioux Nathan et
Baude Loïck

<https://github.com/LoickBde/Symforion>

Sommaire

| | |
|---|----|
| Introduction | 3 |
| Installation et lancement | 4 |
| Storyboard de l'application | 5 |
| Écran de connexion | 5 |
| Écran d'accueil élève | 5 |
| Écran d'accueil Professeur | 6 |
| Écran d'accueil admin | 6 |
| Écran de gestion des promos (Admin) | 7 |
| Écran de gestion des notes (Professeurs) | 7 |
| Écran de visualisation des notes | 8 |
| Modèle objet | 9 |
| Implémentation de l'héritage User (Nathan) | 10 |
| Authentification et firewall (Nathan) | 11 |
| Ajout de notes (Loïck) | 14 |
| Côté backend | 14 |
| Côté frontend | 15 |
| Visualisation des notes (Matthias) | 16 |
| Côté frontend | 16 |
| Côté backend | 17 |
| Génération automatique du bulletin en PDF (Othmane) | 18 |

Introduction

L'objectif initial de notre projet était de faire un site de visualisation de leurs notes pour des étudiants et de saisie de notes pour des professeurs.

L'inspiration étant notre site de visualisation de note WebAurion.

Nous avons ainsi développé les fonctionnalités suivantes :

- Modélisation de la bdd
- Implémentation de l'héritage User
- Développement de fixtures
- Authentification et firewall
- Visualisation des notes par les élèves
- Génération de bulletin PDF pour les élèves
- Saisi des notes par les professeurs
- Gestion des promotions, matières et utilisateurs par les administrateurs

Elles ne seront pas toutes expliquées dans ce bulletin compte-rendu même si elles font parties intégrantes de notre site web.

Installation et lancement

Installation :

Créer un `.env.local` à partir du `.env` existant et remplacer la ligne ci-dessous par vos informations de BDD :

```
DATABASE_URL="postgresql://<user>:<password>@127.0.0.1:5432/<db_name>?serverVersion=12.6&charset=utf8"
```

Ensuite exécuter la commande :

```
composer deploy  
php bin/console doctrine:fixtures:load;
```

Ou

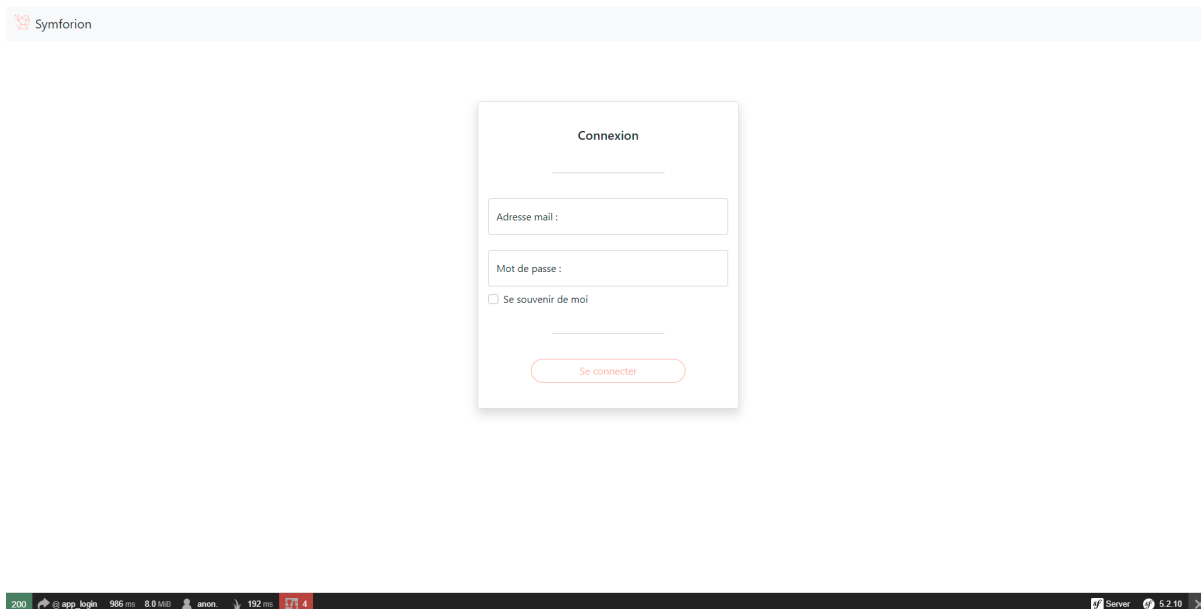
```
composer install;  
composer update;  
php bin/console doctrine:database:create;  
php bin/console doctrine:migration:diff;  
php bin/console doctrine:migration:migrate;  
php bin/console doctrine:fixtures:load;
```

Le projet contient plusieurs Fixtures afin d'avoir un site fonctionnel sans ajout. Pour se connecter, vous pouvez utiliser un de ces 3 comptes :

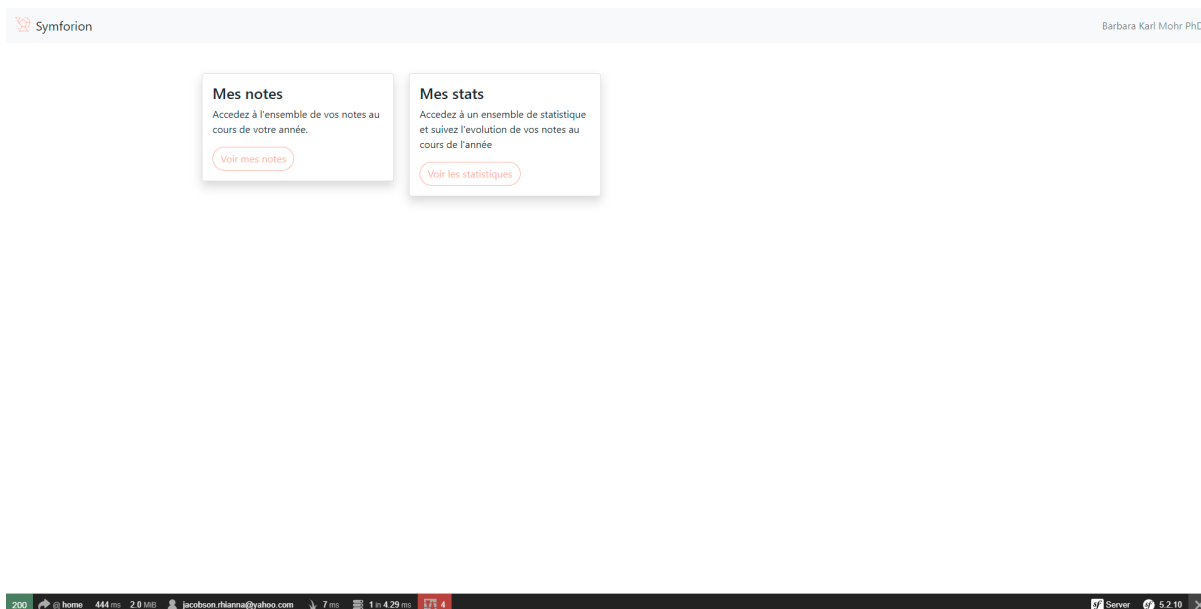
- Compte élève : `eleve@gmail.com` - Eleve123
- Compte professeur : `prof@gmail.com` - Prof123
- Compte administrateur : `admin0@gmail.com` - Admin123

Storyboard de l'application

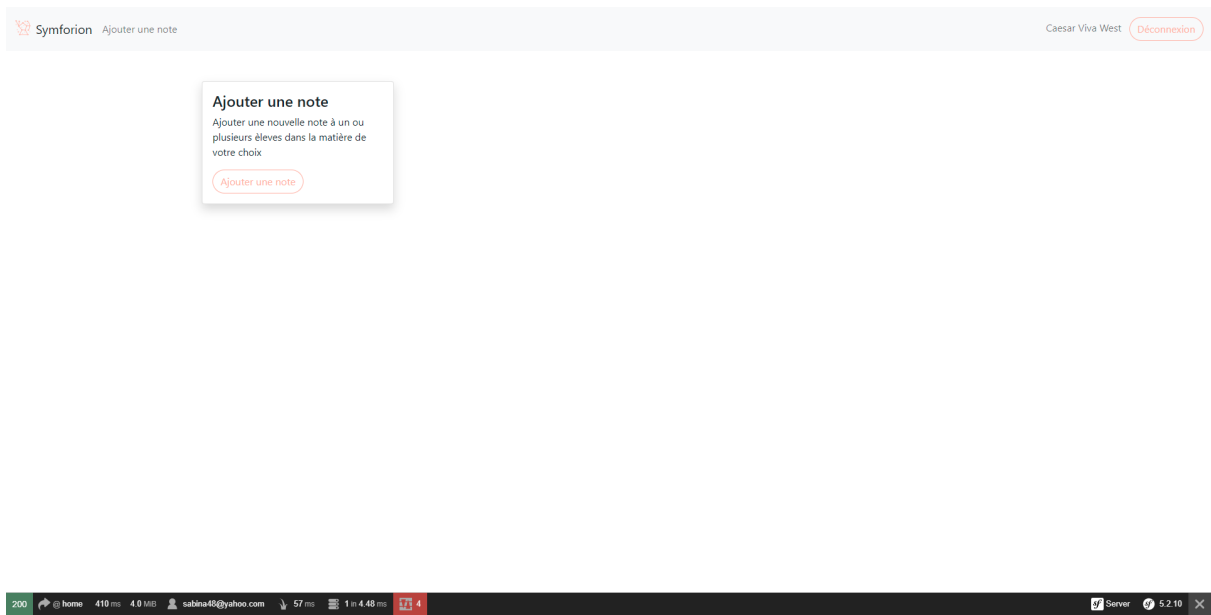
Écran de connexion



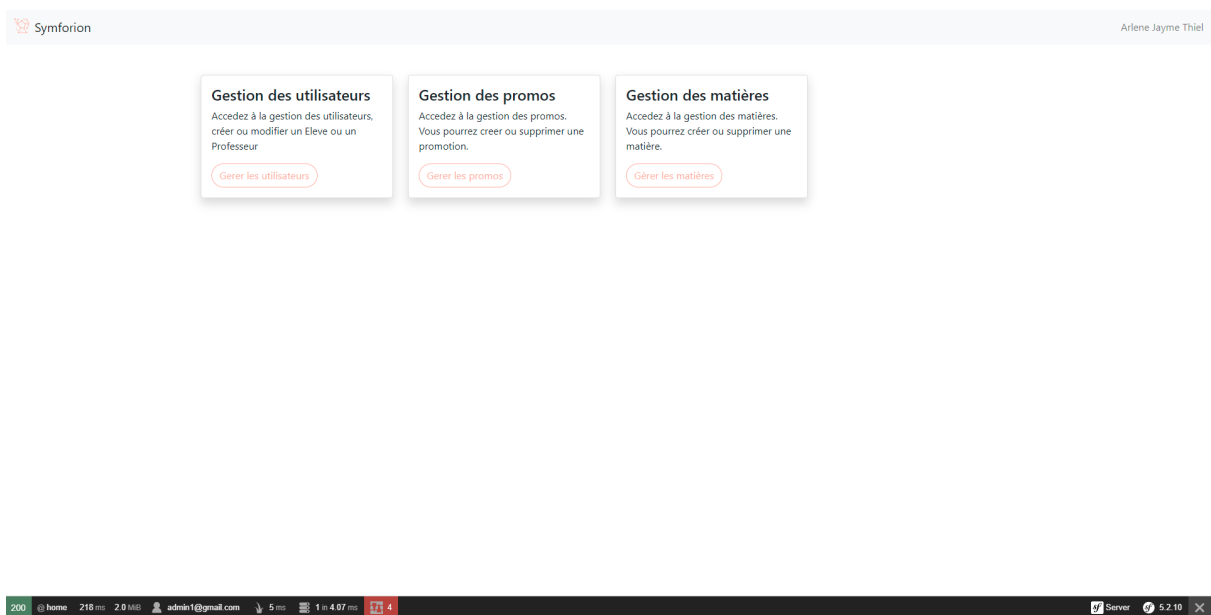
Écran d'accueil élève



Écran d'accueil Professeur



Écran d'accueil admin



Écran de gestion des promos (Admin)

Symforion

Regan Brenden GoodwinDéconnexion

Gestion des Promos

Ajouter une promo

Nom de la promo

Ajouter la promo

Liste des Promos

Show 10 entries

Search:

| Id | Name | Action |
|----|------|--------|
| 15 | LE1 | |
| 16 | LE2 | |
| 44 | LE3 | |
| 45 | LE4 | |
| 46 | LE5 | |
| 47 | LA1 | |
| 48 | LA2 | |
| 49 | LA3 | |

Showing 1 to 8 of 8 entries

Previous1Next

Ecran de gestion des utilisateurs (Admin)

Symforion

Arlene Jayme Thiel

Gestion des Utilisateurs

Ajouter un utilisateurs

Ajouter un eleve

Ajouter un professeur

Ajouter un administrateur

Liste des Utilisateurs

Show 10 entries

Search:

| Id | Email | Prenom | Nom | Role | Action |
|-----|------------------|-----------|------------------------------|----------------|--------|
| 408 | admin0@gmail.com | Royal | Mr. German Runolfsdottir PhD | Administrateur | |
| 409 | admin1@gmail.com | Arlene | Jayme Thiel | Administrateur | |
| 410 | admin2@gmail.com | Houston | Rodger Kihn | Administrateur | |
| 411 | admin3@gmail.com | Jayda | Ezequiel Breitenberg | Administrateur | |
| 412 | admin4@gmail.com | Margarete | Jacky Johnston | Administrateur | |
| 413 | admin5@gmail.com | Ethan | Judah Gaylord | Administrateur | |
| 414 | admin6@gmail.com | Veronica | Dr. Nikolas Wiegand II | Administrateur | |
| 415 | admin7@gmail.com | Sydney | Jade Hirthe | Administrateur | |
| 416 | admin8@gmail.com | Maeve | Madison Jerde | Administrateur | |
| 417 | admin9@gmail.com | Carmine | Kameron Cormier | Administrateur | |

Ecran de gestion des matières (Admin)

Symforion

Arlene Jayme Thiel

Gestion des Matières

Ajouter une metière

Ajouter la matière

Liste des Matières

Show 10 entries

Search:

| id | Name | Teacher | Action |
|----|---------------|------------------------------|-------------|
| 54 | Mathématiques | Burnice Dr. Cooper Mertz Jr. | <div></div> |

| 55 | Mécanique | Aliza Humberto Friesen | |

| 56 | Algorithmie | Rolando Justus Skiles | |

| 57 | SDA | Jamel Casimir Howell | |

Showing 1 to 6 of 6 entries

Previous1Next

Écran de gestion des notes (Professeurs)

Symforion

Ajouter une note

Caesar Viva West

Déconnexion

Bonjour Viva West Caesar.

Ajouter une note

Matières: Mathématiques

Promo: LE1

Type: CB

Élève: Annalise Littell - Iliana

Note (/20):

Coef.

Description

Effacer les champs

Ajouter la note

200

@ manage_marks

299 ms

4.0 MB

1

sabina48@yahoo.com

26 ms

3 in 8.33 ms

4

Server

6.2.10

Écran de visualisation des notes (élèves)

SymforionAfficher mes notes

Barbara Karl Mohr PhDDéconnexion

Mes dernières notes

Show10entries

Search:

| note | coef | type | description | matière |
|------|------|------|--|--------------------|
| 0 | 3 | CC | Libero quas est nihil fugiat eos eos sed. | Micro-informatique |
| 1 | 0 | CC | Non tempore ratione nostrum hic eaque autem excepturi. | Web |
| 4 | 4 | CC | Qui vitae nulla consequuntur ducimus eos possimus. | Micro-informatique |
| 8 | 2 | CC | Qui illum fugit sapiente optio autem esse. | Micro-informatique |
| 16 | 3 | CC | Deleniti nisi ratione voluptatum repellat. | Mathématiques |
| 19 | 3 | CT | Necessitatibus hic dignissimos ea molestiae eos vero ut. | Web |
| 19 | 3 | CC | Accusamus et molestiae temporibus amet. | Web |

Showing 1 to 7 of 7 entries

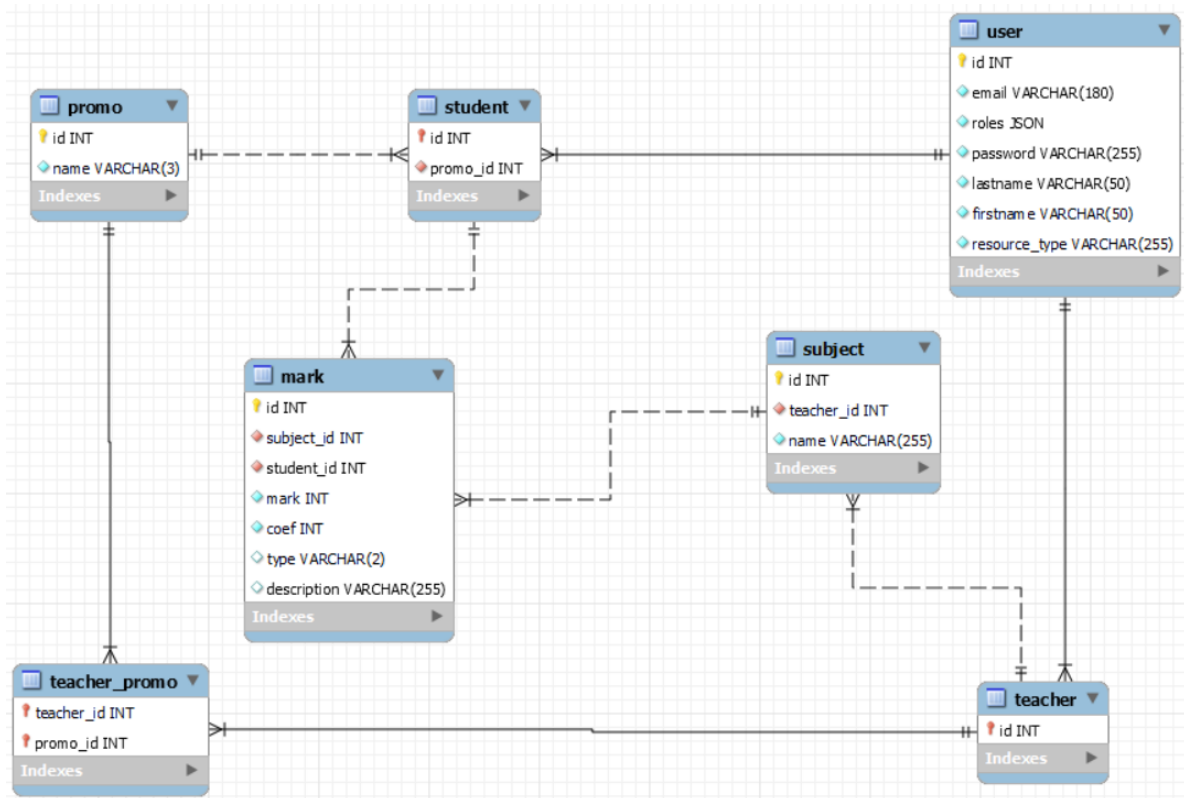
Previous1Next

200display_marks335ms2.0msjacobson.rhianaa@yahoo.com23ms5in7.54ms4

Server6.2.10

Modèle objet

Pour notre application, nous avons modélisé les relations suivantes. Les utilisateurs (administrateur, professeur et étudiant) héritent tous de la table user. Nous avons ensuite les relations logiques entre les utilisateurs et les différentes entités (un étudiant à une promotion, peut avoir plusieurs notes, ...).



Implémentation de l'héritage User (Nathan)

Dans le cadre de ce projet nous avons besoin de différencier trois types d'utilisateurs au sein de l'application :

- Les Administrateurs : Gèrent la création/modification d'utilisateurs (encore à faire) et de promos
- Les Professeurs : Ajoutent les des notes aux élève dans les matières où ils sont assignés
- Les Élèves : Visualisent leurs notes et leurs statistiques (encore à faire). Ils peuvent aussi avoir un bulletin sous format PDF (encore à faire ?)

Ces trois classes héritent de la classe User de symfony comportant des informations tels que :

- Email
- Password
- Nom
- Prenom
- Rôles
- ...

D'après le modèle objet, un élève est relié à une promo et des notes tandis qu'un prof est relié à **plusieurs** promo ainsi qu'à des matières.

Doctrine (ORM utilisé par symfony) propose 2 implémentations des héritages (au niveau de la BDD) :

- **Single-Table Inheritance** : Les attributs des classes filles sont tous remontés dans la classe mère (dans notre cas, la classe User). Certains attributs seront donc NULL et Doctrine ajoutera une colonne Discriminator pour faire la différence entre chaque classe.
- **Class-Table Inheritance** : Chaque classe (mère et filles) sera transformée en une table au niveau de la BDD. Chaque table fille sera liée à son la table mère via une contrainte de clé étrangère.

Nous avons, au départ, décidé d'utiliser la première méthode (Single-Table Inheritance) mais celle-ci n'était pas adaptée à notre modèle. En effet, les profs et les élèves sont liés à des promos mais avec des relations différentes (ManyToMany pour les professeurs et OneToMany pour les élèves).

Au niveau de la bdd, cela créait une colonne promo_id pour qualifier la relation Promo/élève or cette colonne était NOT_NULL. Nous étions donc donc dans l'impossibilité de créer un professeur car l'attribut promo_id n'était pas spécifié.

Nous avons donc opté pour la seconde méthode qui résout ce problème.

Authentication et firewall (Nathan)

Afin de mettre en place un système de connexion sur l'application, nous avons dû mettre en place plusieurs éléments.

Le premier d'entre eux est la classe User. Symfony nous permet de créer une classe User toute faite grâce à la commande :

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

La classe User créée comporte alors des informations de base :

- email
- password
- username
- firstName
- lastName
- rôles : Ces rôles permettent de restreindre l'accès à certaines pages selon le types d'utilisateurs

La seconde chose à faire est de mettre en place un firewall et un provider dans le fichier security.yaml (dans le fichier config/packages/)

Le provider permet de spécifier quelle classe représentera les utilisateurs dans les sessions, il faut alors spécifier la classe User.

```
providers:
    # used to reload user from session & other features (e.g.
    # switch_user)
    app_user_provider:
        entity:
            class: App\Entity\Users
            property: email
```

Le firewall sert quant à lui à spécifier la route servant à identifier un User (login_path), la route permettant de se déconnecter, les paramètres du remember_me (temps d'expiration par exemple) ainsi que l'authenticator.

Voici à quoi ressemble notre firewall main : (tu peux mettre en yaml stp matiacé)

```
firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false
  main:
    anonymous: true
    lazy: true
    provider: app_user_provider
    remember_me:
      secret: '%kernel.secret%'
      lifetime: 604800 # 1 week in seconds
      path: /
      secure: true

    form_login:
      login_path: app_login
      check_path: app_login
    guard:
      authenticators:
        - App\Security\LoginFormAuthenticator
    logout:
      path: app_logout
```

Enfin, la dernière chose à faire est d'implémenter l'authenticator. Cette dernière contient une partie de la logique de connexion. Elle permet entre autre de :

- Récupérer les credentials issu du formulaire (fonction getCredentials)
- Récupérer le user lors de la connexion en se basant sur son email (fonction getUser)
- Checker si le mot de passe correspond (fonction checkCredentials)
- Rediriger l'utilisateur en cas de succès de l'authentification (fonction onAuthenticationSuccess)

A noter que nous avons utilisé des mots de passe cryptés grâce à l'algorithme bcrypt (spécifier dans le security.yaml). L'authenticator va alors automatiquement utilisé cet algorithme pour checker si les mot de passes correspondent

Le fichiers security.yaml nous permet aussi de restreindre l'accès à certaines routes selon l'utilisateur. (à mettre en yaml stp matiasse)

```
access_control:
  - { path: ^/admin, roles: ROLE_ADMIN }
  - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY}
  - { path: ^/manage/promo,roles: ROLE_ADMIN }
  - { path: ^/manage/user, roles: ROLE_ADMIN }
  - { path: ^/manage, roles: ROLE_TEACHER }
  - { path: ^/display, roles: ROLE_STUDENT }
  - { path: ^/, roles: ROLE_USER }
```

Cela nous permet alors de :

- Rendre le site accessible aux personnes connectées uniquement
- Rendre le portail de connexion accessible aux personnes non-connectées uniquement (Cela permet de rediriger automatiquement vers la page de connexion si on essaye d'accéder à quelque route de l'application)
- Rendre les pages des professeurs uniquement accessibles aux professeurs (et pareil pour les élèves et administrateurs)

Ajout de notes (Loïck)

Cette fonctionnalité est disponible pour un professeur. Après connexion, il aura la possibilité d'ajouter des notes à ses élèves. La page se présente comme suit.

The screenshot shows a web browser window with the Symfonion application. The header bar includes the Symfonion logo, the text 'Ajouter une note', and a user profile section for 'Caesar Viva West' with a 'Déconnexion' button. Below the header, a message reads 'Bonjour Viva West Caesar.' The main content area features a form titled 'Ajouter une note'. The form contains several fields: 'Matières' (set to 'Mathématiques'), 'Promo' (set to 'LE1'), 'Élève' (set to 'Annalise Littet - Iliana'), 'Type' (set to 'CB'), 'Note (/20):' (empty), 'Coef.' (empty), and a 'Description' text area. At the bottom of the form are two buttons: 'Effacer les champs' and 'Ajouter la note'. The browser's address bar shows the URL 'manage_marks' and various status icons.

Une fois arrivé sur cette page, le professeur se voit proposer un formulaire. Il peut y faire plusieurs chose:

- D'abord, il peut choisir la matière dans laquelle ajouter la note.
- Ensuite, il choisit la promotion et l'élève à qui attribuer la note.
- Puis il choisit le type de contrôle, la note, le coefficient et la description.
- Enfin, il peut valider pour enregistrer et attribuer la note à l'élève.
- Après validation, seul le champ 'note' est remis à 0. De plus, la sélection de l'élève passe au suivant. Ainsi il suffit juste de (re)renseigner la note et de (re) valider.

Côté backend

Cette page est gérée par le contrôleur nommé 'ManageMarksController'. Au sein de ce contrôleur, plusieurs routes sont définies pour :

- Afficher la page et charger les données du professeur (promotion et matières)
- Récupérer les étudiants en fonction de la promotion choisie. L'appel de cette route se fait en ajax.
- Enregistrer une note (type, valeur, coefficient, description) à un élève donné, pour une matière donnée). L'appel de cette route se fait en ajax.

Les données professeur sont chargées à partir de l'utilisateur qui est connecté. Symfony permet de récupérer l'utilisateur courant grâce à la

fonction ' \$this->getUser();' directement dans le contrôleur. Cette fonction m'a été très utile.

Les données sont récupérées à partir des différents repository de l'application (student, teacher, subject, ...). Les données enregistrées, en l'occurrence ici la note de l'élève, est sauvegardée dans la base de données en traitant un JSON reçu via ajax et en utilisant Doctrine pour la persistance des données.

Côté frontend

Côté client, au niveau du design, j'ai utilisé bootstrap 5.0 pour le rendu de la page. J'ai utilisé différents composants :

- Un élément de type card pour le conteneur
- Un formulaire
- Des inputs pour les champs
- Des boutons (stylisés pour notre correspondre avec notre thème défini)

J'ai créé une interface simple, épurée et avec des formes et des couleurs du thème de l'application que l'on a défini ensemble.

Le nom de l'utilisateur courant (connecté) est récupéré dans le template grâce l'objet 'app.user' disponible grâce à twig. Cela m'a également été très utile et cela ressemble beaucoup à la fonction pour récupérer l'utilisateur dans le contrôleur.

Au niveau du script, j'ai utilisé JavaScript et le framework jQuery pour simplifier son utilisation.

J'ai créé un évènement sur le 'select-input' des promotions pour effectuer un appel ajax afin de récupérer les élèves d'une promotion donnée. Un second évènement sur le bouton 'ajouter une note' (ou l'appui sur 'entrer') permet de récupérer les champs des inputs afin de créer un objet JSON à envoyer au contrôleur en ajax afin d'enregistrer une note.

Lors de l'ajout d'une note, une vérification est réalisée. Si il manque le coefficient et la valeur de la note, un message d'erreur apparaît. De plus, il est impossible pour l'utilisateur d'entrer une note ou un coefficient incohérent (note < 0, note > 20 ou coefficient <= 0).

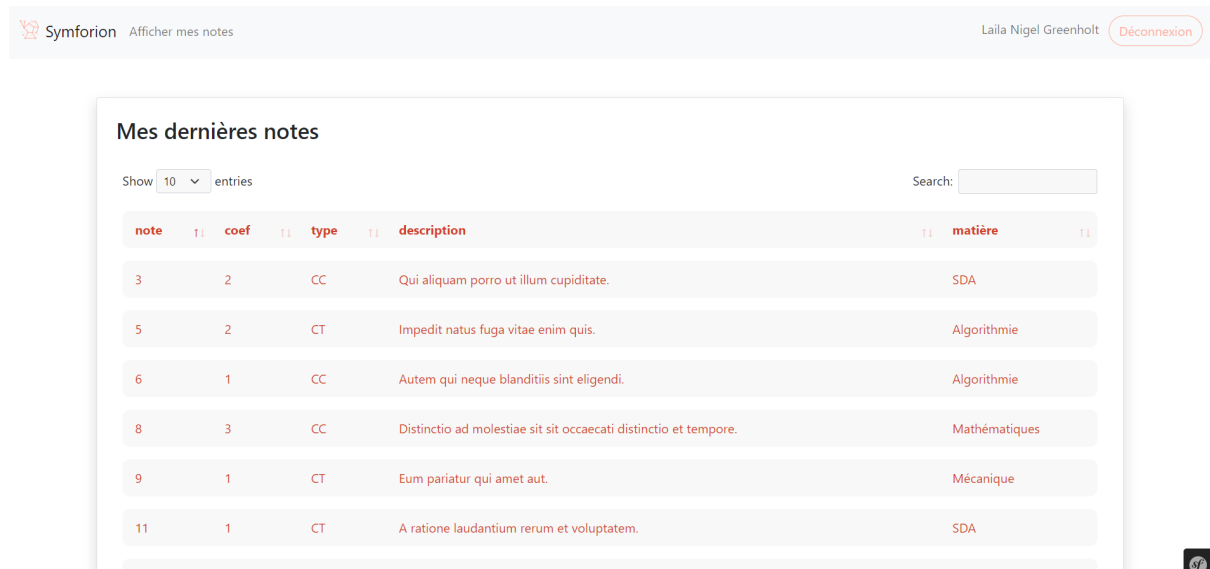
Si l'ajout de la note s'est correctement réalisée, seul l'input note est réinitialisé et l'élève suivant de la liste est sélectionné. Ainsi il est plus facile et plus rapide pour le professeur d'ajouter des notes. De plus, il peut utiliser le bouton 'entrer' pour aller encore plus vite.

Enfin, il a la possibilité de nettoyer tous les champs du formulaire grâce au bouton prévu à cet effet.

Visualisation des notes (Matthias)

Cette fonctionnalité est disponible aux usagés de type "élève". Une fois qu'il est connecté, l'élève peut visualiser ses notes sur cette page.

La page se présente de cette manière :



The screenshot shows a web interface for a student named Laila Nigel Greenholt. The page title is "Mes dernières notes". It features a table with the following data:

| note | coef | type | description | matière |
|------|------|------|--|---------------|
| 3 | 2 | CC | Qui aliquam porro ut illum cupiditate. | SDA |
| 5 | 2 | CT | Impedit natus fuga vitae enim quis. | Algorithmie |
| 6 | 1 | CC | Autem qui neque blanditiis sint eligendi. | Algorithmie |
| 8 | 3 | CC | Distinctio ad molestiae sit sit occaecati distinctio et tempore. | Mathématiques |
| 9 | 1 | CT | Eum pariatur qui amet aut. | Mécanique |
| 11 | 1 | CT | A ratione laudantium rerum et voluptatem. | SDA |

Côté frontend

Comme nous pouvons le voir, la page se présente comme un simple tableau avec pour chaque ligne, une note. Cette note comporte plusieurs attributs et est directement lié aux notes ajoutées dans la page **Ajout de notes**

La note comporte ainsi son élément principal, la note, mais aussi, son coefficient, son type, sa description, et la matière associée.

Pour afficher les notes, on reçoit un tableau avec toutes les notes, et on peuple le tableau d'affichage de ces dernières.

Le tableau est stylisé afin d'avoir un aspect avec des lignes rondes et des formes arrondies. Les couleurs sont en accord avec le reste du site. Le style Bootstrap 5 est utilisé aussi pour que le tableau soit plus harmonieux et encore une fois en accord avec le reste du site.

Ce tableau est en réalité un dataTable de JQuery, ces tableaux permettent d'intégrer directement divers fonctionnalités comme par exemple le trie de toutes les colonnes par l'ordre croissant ou décroissant, mais aussi la recherche d'une note par n'importe lequel de ses attributs, par exemple le nom de la matière, ou encore le type de note.

Enfin ce tableau formate automatiquement le nombre de notes affichées, il permet de choisir d'afficher 10 ou plus de notes, et formate ainsi le nombre de pages à construire et à afficher.

Côté backend

Lorsqu'un utilisateur effectue une requête sur la page */display/marks*, le controller **DisplayMarks** la récupère. A ce moment-là, il récupère l'utilisateur courant et l'assigne à l'attribut "Student".

Une fois cela fait, le controller envoie une requête la base de données, plus particulièrement la table de note afin de récupérer un tableau contenant toutes les notes de l'élève actuel.

Il envoie ensuite cette information à la page Twig qui se chargera de l'affichage pour l'utilisateur.

Génération automatique du bulletin en PDF (Othmane)

Cette fonctionnalité est disponible chez l'élève. Cette partie a un rapport avec la partie précédente qui est: ajouter à la page précédente est un lien pour que le bulletin soit généré et téléchargé en PDF.

La page se présente maintenant de cette manière :

Mes dernières notes

Show 10 entries

Search:

| note | coef | type | description | matière |
|------|------|------|---|--------------------|
| 0 | 2 | CC | Consequatur velit sapiente qui itaque dolor. | Mécanique |
| 2 | 4 | CT | Nemo totam sit repellat beatae vel repellat veniam explicabo. | SDA |
| 4 | 4 | CT | Vel architecto similique aliquam et est libero. | Algorithmie |
| 5 | 1 | CC | Est similique eveniet et. | Mécanique |
| 6 | 1 | CT | Repellat velit iste officiis natus ducimus. | Web |
| 9 | 1 | CT | Laudantium in et voluptas quo omnis. | Mécanique |
| 10 | 2 | CT | In cupiditate est quae eius. | Mathématiques |
| 13 | 2 | CT | Et corrupti debitis nihil architecto quibusdam. | SDA |
| 14 | 2 | CT | Ab id rerum sequi adipisci totam aliquam mollitia mollitia. | Algorithmie |
| 18 | 3 | CC | Repudiandae dolore facere eum totam. | Micro-informatique |

Showing 1 to 10 of 10 entries

Previous 1 Next

Voir Bulletin

Après avoir appuyé sur le lien, notre bulletin se génère et on le voit de cette manière :



Le fonctionnement pour afficher le bulletin est pratiquement le même que celui de l'affichage des notes. C'est-à-dire que lorsque l'étudiant visualise son bulletin, il fait une requête vers `/display/bulletin`, le contrôleur `DisplayMarks` la récupère. Après on récupère le contenu du template, dans notre cas le tableau de notes, pour qu'ensuite on utilise ce contenu pour générer un objet Response qui est dans notre cas un PDF. Pour cela, il faut installer le paquet `KnpSnappy` à l'aide de la commande et:

```
composer require knplabs/knp-snappy
```

et aussi `wkhtmltopdf`.

Pour le calcul de la moyenne générale de l'élève, je l'ai fait directement sur la page twig du bulletin; on a créé deux variables:

- La première pour calculer la note multipliée par le coefficient de cette note
- La deuxième pour calculer la somme de tous les coefficients.

Pour qu'ensuite l'afficher dans un autre tableau.