

Rapport final : Projet programmation Web II

Baldewyns Tanguy

2020-2021

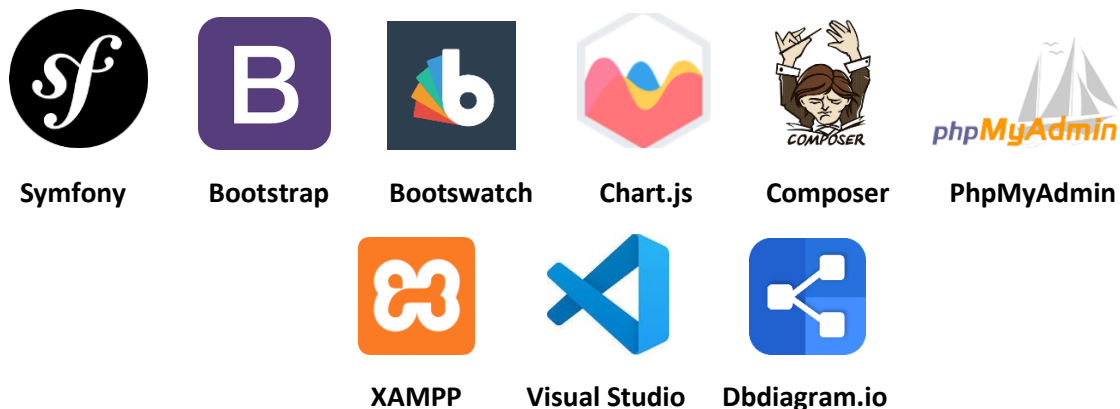
1.0 Introduction.....	4
2.0 Analyse et choix des outils utilisé lors du développement	4
2.1 Symfony.....	4
2.2 Bootstrap.....	4
2.3 Bootswatch.....	4
2.4 Chart.js.....	4
2.5 Composer	4
2.6 PhpMyAdmin.....	5
2.7 XAMPP	5
2.8 Visual Studio Code.....	5
2.9 Dbdiagram.io	5
3.0 Méthodologie :	5
3.1 Introduction à Symfony :	5
3.2 Imagination des concepts :	5
3.3 Imagination de la base de données :	5
3.4 Mise en place des fonctionnalités requise :	5
3.5 Imagination et mise en place des fonctionnalités supplémentaires :	5
3.6 Réalisation du rapport :	5
4.0 Identification des concepts	6
4.1 Concept global :	6
4.2 Base de données :	6
4.3 Table « user » :	7
4.4 Table « notes » :	7
4.5 Table « registration_to_course » :	7
4.6 Table « assoc_teacher_courses » :	7
4.7 Table « message » :	8
4.8 Table « work »:	8
4.9 Table « ue » :	8
4.10 Table « courses » :	8
4.11 Unité d'enseignement (UE) et cours:	9
4.12 Rôles :	9
4.13 Relation entre cours et professeurs :	10
5.0 Identification des fonctionnalités.....	11
5.1 Un système d'inscription\connexion sur la plateforme :	11
5.2 Deux interfaces dédiées « administrateur » et « étudiant » :	12

5.3 Les administrateurs doivent pouvoir encoder des données sur la plateforme concernant les parcours étudiants :	13
5.4 Les données encodées doivent être visualisées graphiquement (à l'aide de diagramme) : ..	14
5.5 Un étudiant ne peut visualiser que les données le concernant :	14
5.6 Un administrateur doit pouvoir visualiser les données concernant un groupe spécifiques d'étudiants qu'il a sélectionné :	15
6.0 Fonctionnalités supplémentaires implémentées	16
6.1 Edition du profil :	16
6.2 Recherche intelligente d'un utilisateur :	16
6.3 Profil utilisateur :	18
7.0 Fonctionnalités supplémentaires partiellement implémentées	20
7.1 Messagerie :	20
7.2 système d'évènement :	21
8.0 Conclusion	22
9.0 Sources :	23
10.0 Annexes	24
10.1 Création d'une base de données :	24
10.2 Remplissage automatiques des champs de la table « user » :	24
10.3 Garantir l'unicité d'un champs d'une table de données :	25
10.4 Interface visible par un visiteur non-authentifié :	25
10.5 Définition de la « class » en fonction du « type » :	25
10.6 Entièrement de la page d'accueil par statuts :	26
10.7 Ajouter des restrictions à des champs d'un formulaire :	27
10.8 Déploiement des graphiques au sein de Symfony :	28
10.9 Filtrer les éléments à afficher dans le diagramme des notes :	29
10.10 Filtrer les notes pour n'avoir accès qu'à ses notes en tant qu'étudiant :	30
10.11 Vérification de la clé d'inscription à un cours	30

1.0 Introduction

Dans le cadre du cours de programmation Web II, il nous a été demandé de réaliser une application permettant d'encoder et de suivre les parcours académiques des étudiants. Au niveau des technologies imposées, il nous a été demandé d'utiliser un Framework, celui-ci ne nous a pas été imposé. Après avoir comparé différents Framework présent sur le marché, j'ai décidé d'utiliser « Symfony ». Son utilisation sera détaillée plus tard dans le rapport. Dans un premier temps, je vous citerai les technologies utilisées ainsi que leur intérêt au sein du projet, ensuite, je développerai les concepts, suite à cela, il vous sera expliqué les fonctionnalités requises et supplémentaires. Pour finir, il vous sera expliqué les problèmes rencontrés ainsi qu'une conclusion globale du projet.

2.0 Analyse et choix des outils utilisé lors du développement



2.1 Symfony: Symfony est un Framework écrit en PHP qui fournit des fonctionnalités intéressantes qui visent à accélérer et à simplifier le développement d'un site web. De plus Symfony possède une documentation très complète et intuitive. De plus, il est encore régulièrement mis à jour. Pour exemple, la dernière version à ce jour est la version 5.2.0 et est disponible depuis le 30 novembre 2020

2.2 Bootstrap: Bootstrap est un ensemble de composant permettant la personnalisation de la partie design de sites et d'applications web.

2.3 Bootswatch: Bootswatch fait partie de Bootstrap et permet de personnaliser un site web en fonction d'un thème. De plus, il s'agit d'un open source, ce qui signifie que tout le monde peut créer un nouveau thème et le mettre à disposition des autres utilisateurs. Bootswatch sera utilisé dans ce projet pour avoir un rendu final propre et harmonieux.

2.4 Chart.js: Chart.js est une bibliothèque JavaScript gratuite permettant la visualisation des données en utilisant 8 types de graphiques (barre, ligne, zone, tarte, bulle, radar, polaire et diffusion). Chart.js sera utilisée dans ce projet pour afficher graphiquement les données des parcours des étudiants.

2.5 Composer: Composer est un logiciel gestionnaire de dépendance, cela signifie concrètement qu'il va nous permettre d'installer toutes les bibliothèques dont nous avons besoin.

2.6 PhpMyAdmin: PhpMyAdmin est une application web de gestion de bases de données SQL. Cette application nous permet de visualiser et administrer les bases de données. Dans ce projet nous allons seulement visualiser les bases de données car la partie administration se fera directement depuis Symfony.

2.7 XAMPP: XAMPP est un ensemble de logiciel permettant la mise en place d'un serveur Web local. Nous l'utiliserons ici pour faire la relation entre la base de données et l'application web

2.8 Visual Studio Code: Visual Studio Code est l'éditeur de code que nous allons utiliser pour le développement de ce projet

2.9 Dbdiagram.io: dbdiagram.io est une application web permettant de designer des bases de données

3.0 Méthodologie :

La réalisation d'un projet aussi important se fait suivant une méthodologie. Cette méthodologie est différente car elle se base sur les habitudes de toutes à chacun, de plus, d'autres facteurs peuvent la modifier.

3.1 Introduction à Symfony :

En effet, il s'agit de mon premier projet réaliser avec Symfony, cela signifie qu'il fallait en apprendre le fonctionnement, ce qui n'était pas simple dû au fait que c'est la première fois que je réalise un projet en passant par un Framework.

3.2 Imagination des concepts :

Après avoir compris comment fonctionnait Symfony, il fallait imaginer une application reprenant toutes les fonctionnalités requises.

3.3 Imagination de la base de données :

Après avoir imaginé une application valable, il fallait imaginer une base de données cohérente.

3.4 Mise en place des fonctionnalités requise :

La partie prenant le plus de temps est la mise en place des fonctionnalités requise. Car, en se basant sur un énoncé, il faut arriver à reproduire ce qui est demandé le plus fidèlement possible.

3.5 Imagination et mise en place des fonctionnalités supplémentaires :

Comme il nous a été demandé d'inclure des fonctionnalités supplémentaires au projet, il a d'abord fallu les imaginer pour en trouver qui en soit utiles et cohérente, ensuite, il a fallu les mettre en place.

3.6 Réalisation du rapport :

Après avoir fini l'application, il faut réussir à retranscrire et expliqué ce qui a été codé sous forme de rapport de la manière la plus complète possible.

4.0 Identification des concepts

4.1 Concept global :

Le but premier de l'application est de créer une application permettant de suivre le parcours d'étudiants. Ceci est le but qui nous est demandé dans l'énoncé du projet. Mais mon objectif, en plus de ça, a été de créer une application complète permettant l'administration d'un établissement scolaire. Mon modèle d'administration se base sur le fonctionnement en place dans la haute école. Cet objectif permettrait un affichage dynamique des pages.

4.2 Base de données :

Dans le cadre de ce projet, la base de données « projet2 » a été créée directement depuis Symfony. (Voir annexes) Cette base de données comporte de multiples tables :

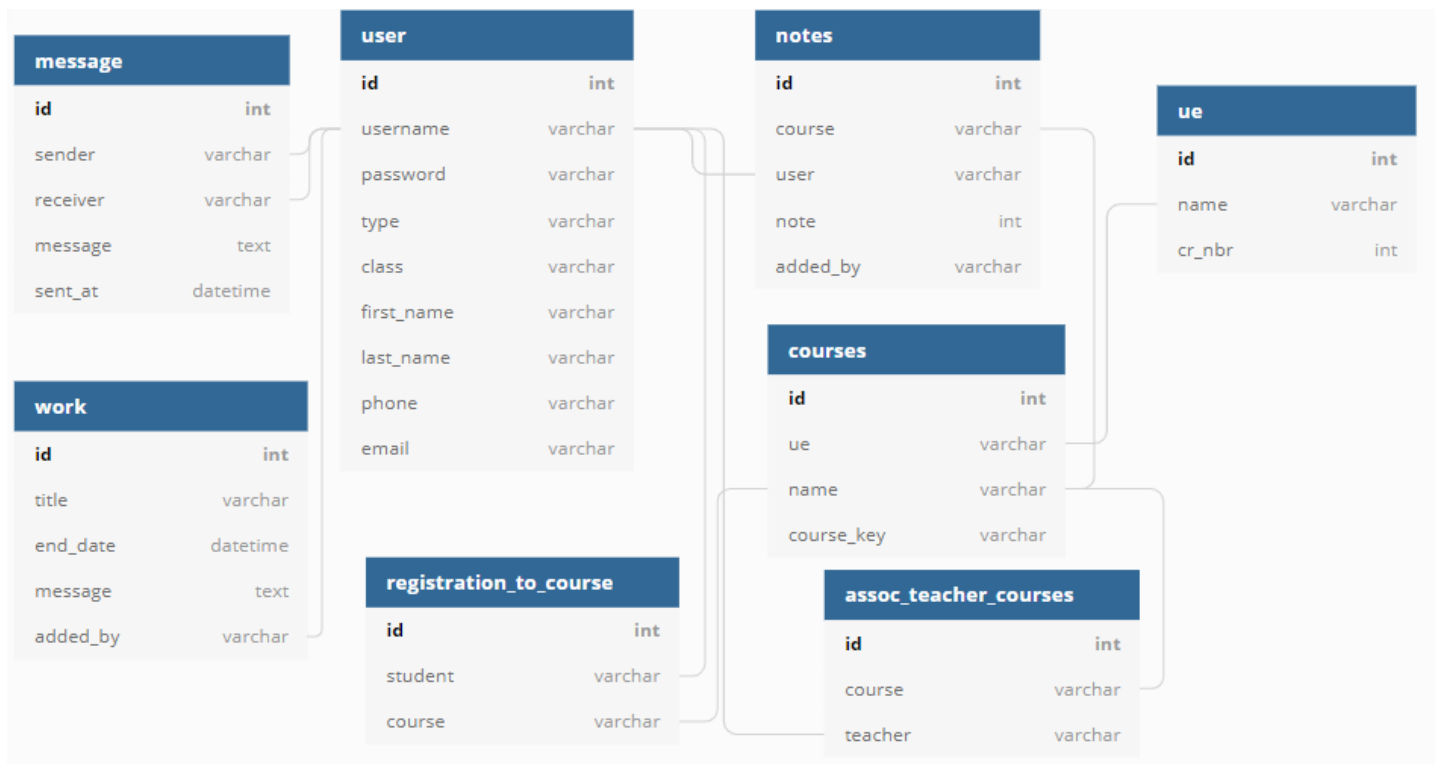


Schéma de la base de données utilisé par le projet

4.3 Table « user » :

user	
id	int
username	varchar
password	varchar
type	varchar
class	varchar
first_name	varchar
last_name	varchar
phone	varchar
email	varchar

La table « user » est la table reprenant toutes les informations personnelles des utilisateurs. Chaque utilisateur compte un « username » unique qui sera formé via la concaténation de la 1^{ère} lettre du « first_name » et de l'entièreté du « last_name ». De plus le paramètre « email » sera lui aussi formé automatiquement. Le reste des champs seront remplis à l'aide d'un formulaire

4.4 Table « notes » :

notes	
id	int
course	varchar
user	varchar
note	int
added_by	varchar

La table « notes » est la table listant toutes les notes de tous les élèves ainsi que le nom d'utilisateur du professeur ayant ajouté chacune des notes.

4.5 Table « registration_to_course » :

registration_to_course	
id	int
student	varchar
course	varchar

La table « registration_to_course » est utilisée pour inscrire un élève à un cours. Elle contient donc deux champs « student » pour l'élève et « course » pour le cours.

4.6 Table « assoc_teacher_courses » :

assoc_teacher_courses	
id	int
course	varchar
teacher	varchar

Tout comme la table « registration_to_courses », cette table est une table associative, elle sert à faire la relation entre deux champs, chacun issu de deux tables différentes. Ici la relation est faite entre un cours (« course ») et un utilisateur (« teacher ») de la table « user »

4.7 Table « message » :

message	
id	int
sender	varchar
receiver	varchar
message	text
sent_at	datetime

Cette table est utile lors de l'envoi et la réception de messages. Elle contient un paramètre « sender » contenant l'expéditeur du message, « receiver », le destinataire, « message » contenant le message en question et « sent_at » la date à laquelle le message a été envoyé à la seconde près

4.8 Table « work » :

work	
id	int
title	varchar
end_date	datetime
message	text
added_by	varchar

Cette table permet de lister les événements. Le nom « event » aurait été plus adéquat mais il s'agit d'un mot réservé qui pourrait ne plus garantir le bon fonctionnement de la base de données dans son entièreté. Les champs présents dans cette table sont : un id, unique et auto-incrémenté, un titre, un message (optionnel, si le titre est assez clair), une date de fin (sous le format « datetime ») et added_by qui sera lié à un nom d'utilisateur de la table « user ».

4.9 Table « ue » :

ue	
id	int
name	varchar
cr_nbr	int

La table « ue » représente les unités d'enseignements présente dans la haute école. La table comporte trois champs. Premièrement « id » qui est un numéro unique auto-incrémenté. Ensuite, « name » qui est le nom de l'unité. Enfin, « cr_nbr » qui représente le nombre de crédits que vaut l'unité.

4.10 Table « courses » :

courses	
id	int
ue	varchar
name	varchar
course_key	varchar

La table « courses » représente chaque cours qui sont donnés au sein de la haute école. Comme chaque cours (AA) fait partie d'une UE, le champs « ue » sera remplie avec une valeur présente dans le champs « name » de la table « ue ». De plus, chaque cours comporte un nom et une clé d'inscription.

4.11 Unité d'enseignement (UE) et cours:

Vu que le projet vise à reproduire le modèle en place dans la haute école. Il a été décidé d'ajouter des unités d'enseignement reproduire au mieux le modèle. Une unité peut contenir un ou plusieurs cours. Les unités et les cours ne peuvent être ajoutés que par les administrateurs.


Chaque **unité** comporte :

- Un id
- Un nom
- Un nombre de crédit.

ID	Name	Nbr of credits	
22	ARCHITECTURE ET TECHNOLOGIE DES ORDINATEURS	8	

Chaque **cours** possède :

- Un id
- Un nom
- Une UE
- Une clé d'inscription

ID	Name	UE	Key	
14	Technologie de l'informatique	ARCHITECTURE ET TECHNOLOGIE DE L'ORDINATEUR	keykey	

4.12 Rôles :

L'un des concepts clés est la gestion des rôles, en effet, un utilisateur peut avoir 3 statuts.

Administrateur, professeur ou utilisateur :

- Un utilisateur (étudiant) peut :

- S'inscrire à des cours
- Consulter ses points
- Éditer son profile
- Envoyer des messages
- (Consulter la liste des travaux à remettre) (pas complètement implémenté)

- Un professeur peut :

- Ajouter des notes aux étudiants
- Éditer son profile
- Envoyer des messages
- (Créer un évènement) (pas complètement implémenté)
- Envoyer des messages

- En revanche, un administrateur a tous les droits, il peut :

- Inscrire un nouveau compte
- Faire la relation entre les professeurs et les course
- Ajouter des cours
- Ajouter des Unités d'enseignement (UE)
- Consulter la liste des utilisateurs
- Consulter les profils des utilisateurs
- (Créer un évènement) (pas complètement implémenté)
- Ajouter des notes aux étudiants

Consulter le profil de chaque utilisateur Toutes ces fonctionnalités ne sont disponibles uniquement lorsque le visiteur est authentifié

4.13 Relation entre cours et professeurs :

Pour une raison de cohérence et de sécurité, j'ai décidé de mettre en place un système de relation entre les cours et les utilisateurs (professeur). Cela permet de ne proposer que les cours reliés au professeur en question lors de l'ajout de notes

Course	Teacher
Programmation web II	jriggio
Make relation	

5.0 Identification des fonctionnalités

Comme expliqué dans l'énoncé qui nous a été donné il y a plusieurs semaines, il est demandé d'inclure des fonctionnalités requises au sein du projet :

5.1 Un système d'inscription\connexion sur la plateforme :

- Inscription :

(Seulement disponible lorsqu'un utilisateur est authentifié en tant qu'administrateur)

Registration

First name	Last name
<input type="text" value="First Name"/>	<input type="text" value="Last Name"/>
Type	Class
<input type="text" value="Student"/>	<input type="text" value="BAC1"/>
Password	Confirm password
<input type="text" value="Password"/>	<input type="text" value="Confirm Password"/>

Register

(La « class » sera automatique définie comme « NONE » si le « type » n'est pas « Student »)

- Connexion :

(Seulement disponible lorsqu'un utilisateur n'est pas encore authentifié)

Connection

<input type="text" value="Username"/>	<input type="text" value="Password"/>
---------------------------------------	---------------------------------------

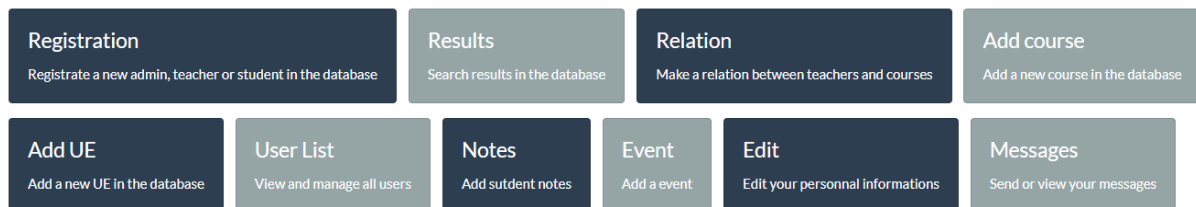
Connection

5.2 Deux interfaces dédiées « administrateur » et « étudiant » :

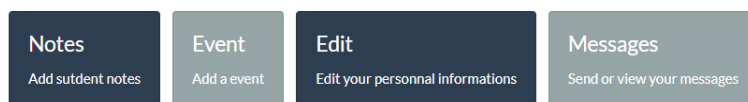
Pour des raisons de sécurité et comme cela a été expliqué lors de 4.0 Identification des concepts, j'ai décidé de rajouter un rôle (professeurs) aux deux rôles déjà existants (administrateur et étudiant).

Voici un aperçu de la page d'accueil dédiées à chacun des rôles (+ voir annexes) :

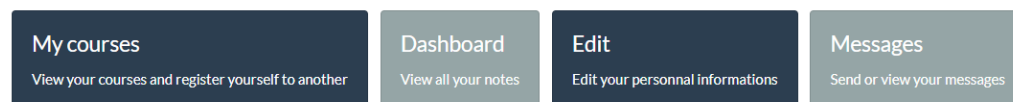
- Administrateur



- Professeur :



- Etudiant :



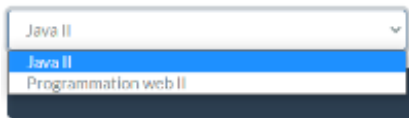
5.3 Les administrateurs doivent pouvoir encoder des données sur la plateforme concernant les parcours étudiants :

Ce sont principalement les professeurs qui encodent les points, mais cela peut aussi se faire depuis l'interface dédiée à l'administrateur. Les professeurs et les administrateurs sont donc les deux seuls rôles ayant le droit d'ajouter des notes aux étudiants.

Course	User	Note
Java II	sstudent	Note
Add		

Nous pouvons apercevoir 4 éléments dans ce formulaire :

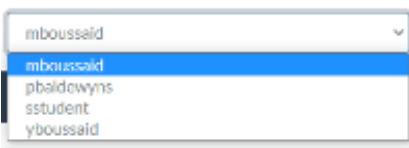
Course



Un champ proposant les cours attribués au professeur.

(Interface professeur)

User



Un champ affichant la liste des étudiants

Note

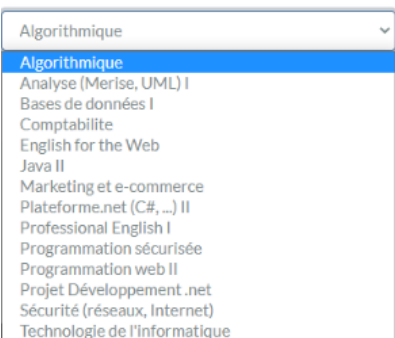


Un champ permettant d'entrer une note entre 0 et 20 (+ détails en annexes)

Add

Un bouton permettant de valider le formulaire

Course



Remarque : si l'ajout de notes se fait depuis un profil administrateur, il n'y a pas de restriction concernant les cours disponibles, car aucun administrateur n'a de cours qui lui sont attribués. C'est pourquoi tous les cours lui seront proposés.

5.4 Les données encodées doivent être visualisées graphiquement (à l'aide de diagramme) :

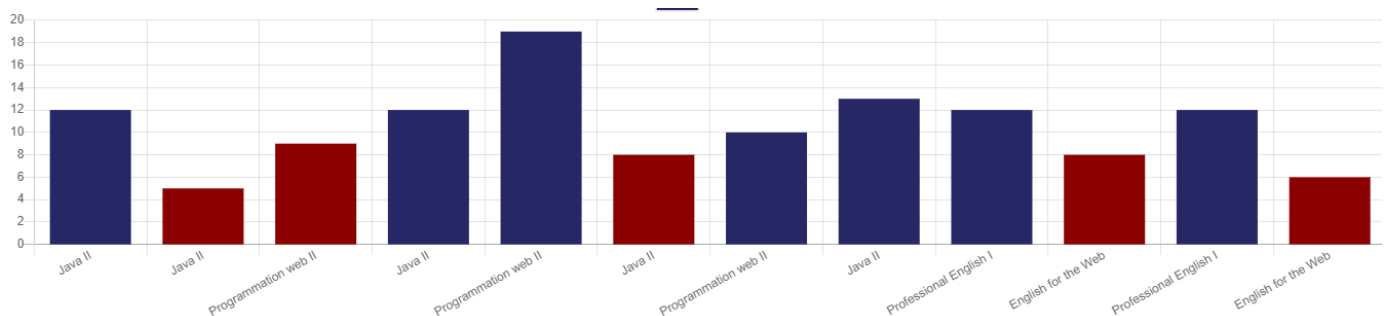
Après s'être authentifié, un étudiant peut consulter son « Dashboard ». Le « Dashboard » est l'endroit où l'étudiant peut visualiser ses points sous forme de liste et de graphique.

- Liste

List of Notes

Course	Note / 20
Java II	12
Java II	5
Programmation web II	9

- Graphique



L'affichage, peut se faire d'une manière globale, comme affiché ci-dessus, mais cela peut aussi se faire de manière plus précise. Pour cela, un système de filtre a été mis en place pour permettre un affichage par cours, par note ou les deux ensembles (voir annexes). Le filtre s'applique au graphique et à la liste.

5.5 Un étudiant ne peut visualiser que les données le concernant :

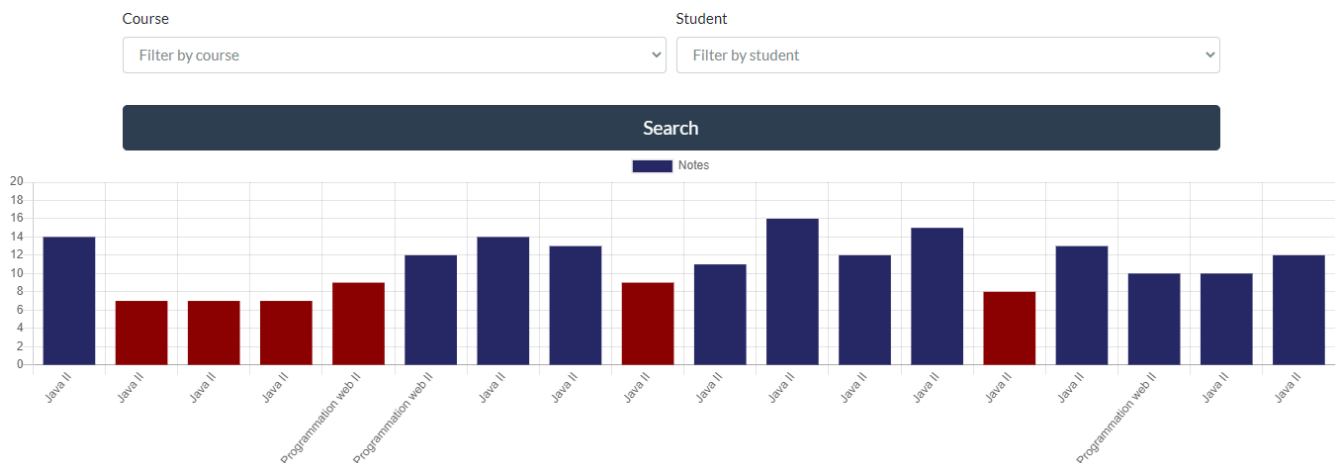
La liste et le graphique des notes sont filtrés en fonction de l'utilisateur connecté pour ne donner l'accès qu'aux personnes en ayant l'autorisation. (Voir annexes)

5.6 Un administrateur doit pouvoir visualiser les données concernant un groupe spécifiques d'étudiants qu'il a sélectionné :

Un administrateur est capable de consulter les données selon, plusieurs critères. Cela peut se faire en fonction de chaque élève, mais aussi de manière plus globale en filtrant les données par cours. L'affichage des données se fait par graphique et par liste. Pour finir, un calcul de la moyenne se fait sur la sélection.

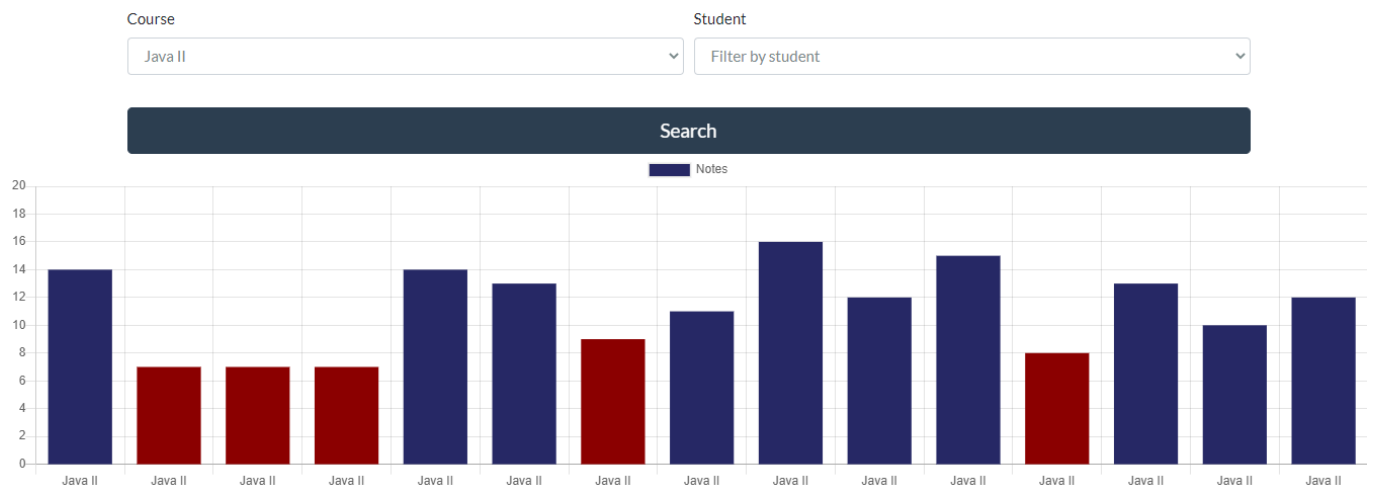
- Affichage graphique sans appliquer les filtres

Medium note of your selection : 11.05555555556



- Affichage graphique en appliquant les filtres

Medium note of your selection : 11.2



6.0 Fonctionnalités supplémentaires implémentées

En plus de l'ajout des concepts expliqués au chapitre du même nom. Il a été décidé d'ajouter d'autres fonctionnalités

6.1 Edition du profil :

Pour des raisons de sécurité, les champs « Username », « Type », « Class » et « Email » ne peuvent être modifiés.

Edit Profile

First name

Admin

Last name

Admin

Username

admin

Type

ROLE_ADMIN

Class

NONE

Phone

Ex: 0412345678

Email

Admin.dmin@projetWeb.com

Password

Password

Confirm password

Password

Update

6.2 Recherche intelligente d'un utilisateur :

Lorsqu'un utilisateur est authentifié comme étant un administrateur, cela lui donne accès à la liste de tous les utilisateurs inscrits. Il peut décider d'afficher tous les utilisateurs à la suite, ou alors, en afficher un certain groupe. La sélection se fait au moyen de 3 filtres bien distincts.

Username

Username

Class

NONE

Type

NONE

Search

ID	Username	Role	Class	Delete
No result				

Par défaut, la liste est vide car aucune sélection n'a été faite

Username: Class: Type:

ID	Username	Role	Class	Delete
1	tbaldewyns	ROLE_ADMIN	NONE	

Premièrement, la recherche peut s'effectuer en fonction du nom d'utilisateur.

Class

NONE
BAC1
BAC2
BAC3

Ensuite, la recherche peut se faire en fonction de l'année d'apprentissage de l'élève.

Type

NONE
Student
Teacher
Admin

Enfin, la sélection des utilisateurs à afficher peut aussi se faire via le rôle de l'utilisateur, qu'il soit un étudiant, un professeur ou un administrateur.

Les 3 filtres peuvent s'additionner pour permettre une recherche plus précise.

6.3 Profil utilisateur :

De nouveau pour la partie administrateur, chaque utilisateur possède un profile utilisateur permettant d'afficher toutes les informations relatives à l'utilisateur en question.

L'accès est possible en directement sur le nom d'utilisateur, le rôle ou la classe du profil recherché

Username


Class

NONE

Type

NONE

Search

ID	Username	Role	Class	Delete
1	tbal dewyns	ROLE_ADMIN	NONE	

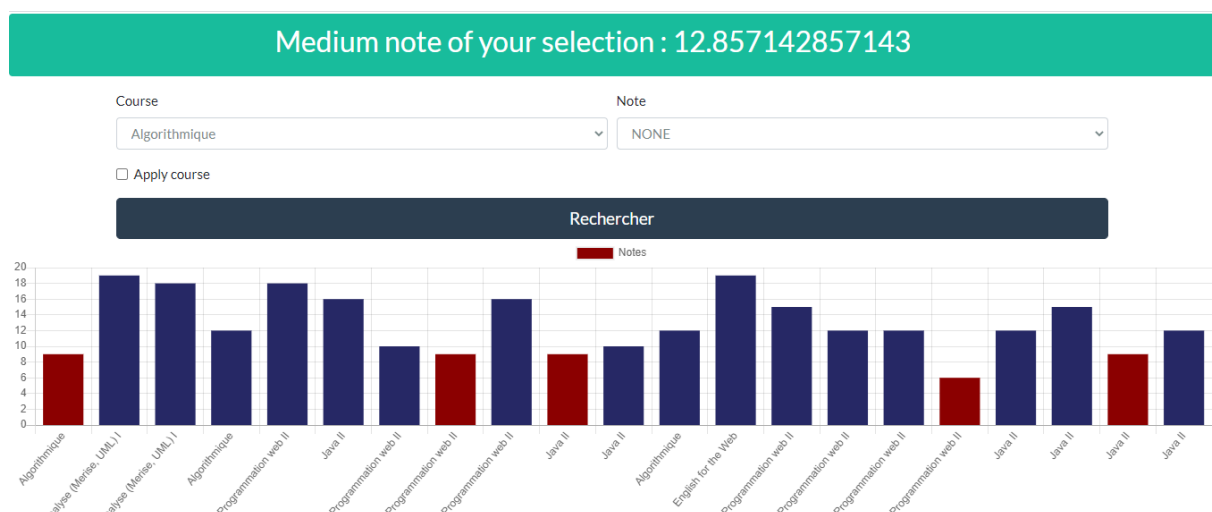
Le profil se divise en plusieurs parties :

- Données personnelles

Profile of Tanguy Baldewyns

First name		Last name	
Tanguy		Baldewyns	
Username	Type	Class	
tbaldewyns	ROLE_ADMIN	NONE	
Phone		Email	
Ex : 0412345678		Tanguy.Baldewyns@projetWeb.com	

- Un Dashboard personnel affichant les points reçus si l'utilisateur est un étudiant et affichant les points émis si l'utilisateur est un professeur ou un administrateur. Deux filtres peuvent être appliqué permettant une recherche plus ciblée.



- La liste reprenant les points du graphique mais sous forme de liste (les filtres s'appliquent aussi sur la liste).

Course	Note / 20
Algorithmique	9
Analyse (Merise, UML) I	19
Analyse (Merise, UML) I	18
Algorithmique	12
Programmation web II	18
Java II	16
Programmation web II	10

- La liste des cours auxquels l'utilisateur est inscrit (seulement disponible pour les étudiants).

Courses

ID	Course
3	Algorithmique
5	Java II

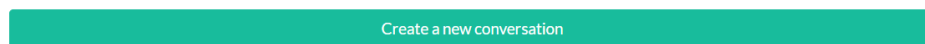
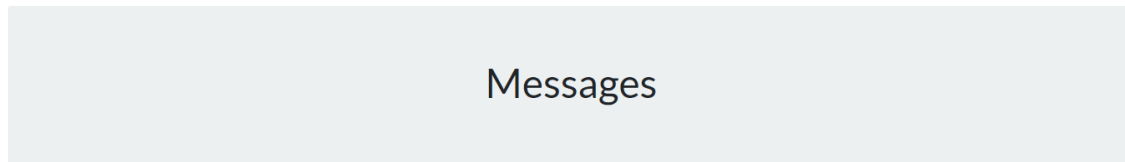
7.0 Fonctionnalités supplémentaires partiellement implémentées

Dû à certains malheureux facteurs, certaines fonctionnalités n'ont pu être complètement implémentées

7.1 Messagerie :

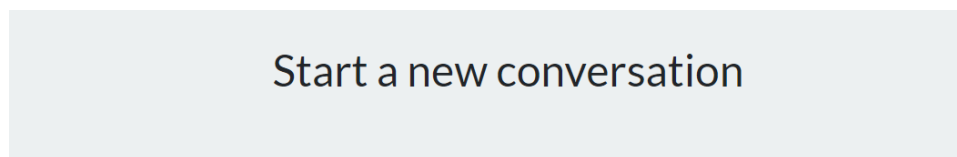
L'idée était de mettre en place un système de messagerie entre tous les utilisateurs.

- La première page du système de messagerie reprend un bouton permettant de créer une nouvelle conversation ainsi que la liste des conversations récents



Sender	Message	Date
yboussaid	Salut	December 14th at 2:35am

- Page permettant de créer une nouvelle conversation

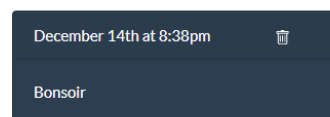
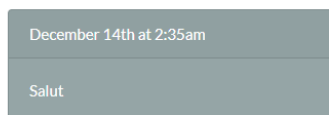


Send to
tbaldewyns

Message
Message

Send

- Conversation avec un autre utilisateur. Chaque message est affiché avec la date à laquelle il a été envoyé ainsi que le message en lui-même. Les messages envoyés se mettent à droite et les messages reçus à gauche.



Message

Send

Le problème qui fait que cette fonctionnalité n'est pas complètement implémentée est que la liste des conversations affiche les messages reçus et non les conversations. La fonctionnalité est donc fonctionnelle mais pas entièrement finie. On peut dire que le système de messagerie est fini à 90%

7.2 système d'évènement :

En plus du système de messagerie, il a été imaginé de mettre en place un système d'évènement qui serait semblable à celui de la haute école. Concrètement, un évènement est un travail à remettre. Un évènement peut être mis en place par un administrateur ou par un professeur. La différence entre les deux se fait au niveau du choix des cours disponibles. Un administrateur peut créer un évènement dans tous les cours, alors qu'un professeur ne peut créer un évènement que dans les cours qui lui ont été attribué.

Event

Course

Java II

End date

Jan

1

2021

00

:

00

Title

Titre de l'évènement

Message

Message optionnel

Add Event

(Exemple d'évènement créé pour le cours de Java II finissant le 1^{er} janvier 2021 à minuit)

List of added events

ID	Course	Title	Message	End date
2	Java II	Titre de l'évènement	Message optionnel	2021 January 1st at 12:00am

Dans l'idée, cet évènement devrait être visible seulement par les étudiants inscrit dans le cours en question. Le système d'inscription à un cours est mis en place.

- Liste des cours auxquels l'étudiant est déjà inscrit

My courses

ID	Course
3	Algorithmique
5	Java II

- Nouvelle inscription

New Registration

Course	Course key
<input type="text" value="Algorithmique"/>	<input type="text" value="Key of the course"/>
<input type="button" value="Register"/>	

Une inscription à un cours se fait en sélectionnant le cours en question et sa clé d'inscription. Une fois le bouton « register » pressé, comparaison est faite avec le champs « course_key » de la classe « courses ». Si les deux champs sont égaux, l'inscription est accordée (voir annexes).

Le problème ici est que, dû au manque de temps, une page affichant les évènements en questions n'a pas pu être créée. On peut dire que le système d'évènement est fini à 80%.

8.0 Conclusion

En conclusion, ce projet m'a honnêtement apporté beaucoup de nouvelles connaissances et le fait de devoir passer par un Framework est un challenge supplémentaire, cela m'a permis d'avoir une approche très différente dans le cadre du développement d'un projet web. Par ailleurs, je pense réellement utiliser Symfony pour mes projets futurs. Car même s'il est assez compliqué à prendre en main, il offre beaucoup de fonctionnalités qui permettent d'écrire une application. Si je devais commencer un nouveau je veillerais à mettre en place des conventions de nommage plus stricts surtout au niveau des bases de données, cela me permettrait de ne pas perdre de temps à chercher si le nom d'une table est au singulier ou au pluriel, avec une première lettre en majuscule ou en minuscule. De plus, j'utiliserais les clés étrangères car je me suis rendu compte à la fin du projet que cela aidait dans certains cas. Mais je ne regrette pas d'avoir fait ces erreurs car elles me permettent d'apprendre.

9.0 Sources¹ :

Introduction à Symfony :

<https://symfony.com/what-is-symfony>

Introduction en vidéo à Symfony :

https://www.youtube.com/watch?v=UTusmVpwJXo&list=PLpUhHhXoxrjdQLodxIHfY09_9XzqdPBW8

Installation de Symfony :

<https://symfony.com/doc/current/setup.html>

Création d'un projet avec Symfony :

https://symfony.com/doc/4.2/best_practices/creating-the-project.html

Base de données avec Symfony :

<https://symfony.com/doc/current/doctrine.html>

L'authentification avec Symfony :

<https://symfony.com/doc/current/components/security/authentication.html>

Formulaires avec Symfony :

<https://symfony.com/doc/current/forms.html>

Contraintes de validation des champs d'un formulaire :

<https://symfony.com/doc/current/reference/constraints.html>

Type de champs dans un formulaire Symfony :

<https://symfony.com/doc/current/reference/forms/types.html>

Sécurité dans Symfony :

<https://symfony.com/doc/current/security.html>

Comprendre les routes avec Symfony :

<https://symfony.com/doc/current/routing.html>

Intégration d'un graphique dynamique Avec Chart.js :

<https://www.youtube.com/watch?v=kf-eY3iZ7Fg>

Conversion d'un objet en chaîne de caractère avec Symfony :

<https://stackoverflow.com/questions/47479268/symfony-3-object-of-class-could-not-be-converted-to-string>

¹ Liste non exhaustive des sources utilisées lors du développement du projet.

10.0 Annexes

10.1 Création d'une base de données :

Installation de Doctrine :

Installing Doctrine ¶

First, install Doctrine support via the `orm` [Symfony pack](#), as well as the MakerBundle, which will help generate some code:

```
> composer require symfony/orm-pack
> composer require --dev symfony/maker-bundle
```

Configuration du `.env` :

Configuring the Database ¶

The database connection information is stored as an environment variable called `DATABASE_URL`. For development, you can find and customize this inside `.env`:

```
###> doctrine/doctrine-bundle ###
# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html
# For an SQLite database, use: "sqlite:///kernel.project_dir%/var/data.db"
# For a PostgreSQL database, use: "postgresql://db_user:db_password@127.0.0.1:5432/db_name?serverVersion=12"
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
DATABASE_URL=mysql://root:@127.0.0.1:3306/Projet1
###< doctrine/doctrine-bundle ###
```

Création de la base de donnée :

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

10.2 Remplissage automatique des champs de la table « user » :

```
use Symfony\Component\String\UnicodeString;
```

Username :

```
$username = (new UnicodeString())
    ->append(substr($user->getFirstName(),0,1))
    ->append($user->getLastName());
$user->setUsername(strtolower($username));
```


Email :

```
$email = (new UnicodeString())
    ->append($user->getFirstName())
    ->append('.')
    ->append($user->getLastName())
    ->append('@projetWeb.com');

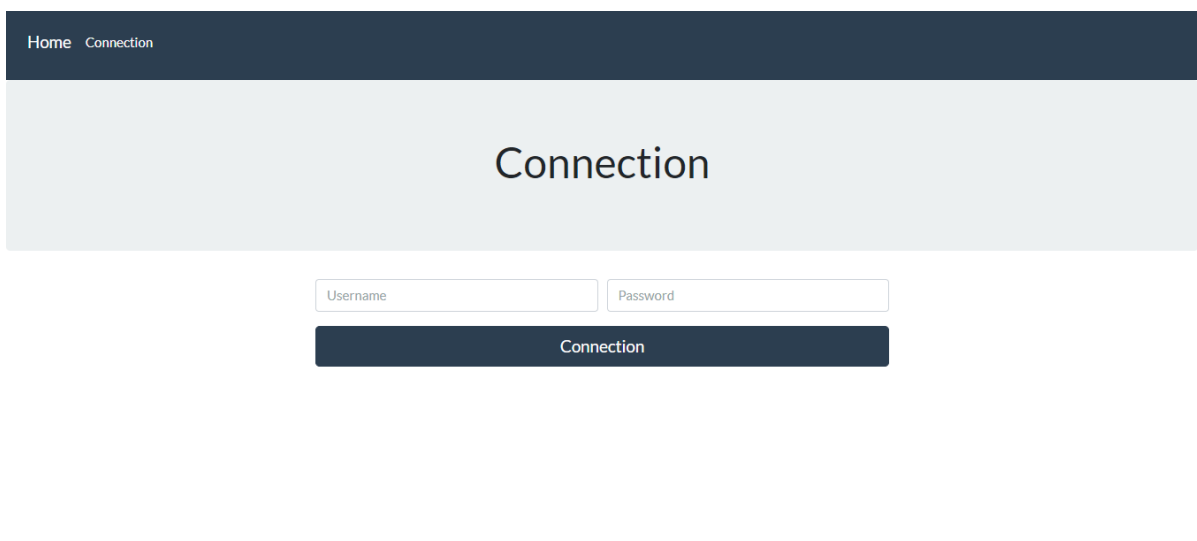
$user->setEmail($email);
```

10.3 Garantir l'unicité d'un champs d'une table de données :

```
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;

/**
 * @ORM\Entity(repositoryClass=UserRepository::class)
 * @UniqueEntity(fields={"username"}, message="Le nom d'utilisateur est déjà utilisé")
 */
class User implements UserInterface
{
```

10.4 Interface visible par un visiteur non-authentifié :

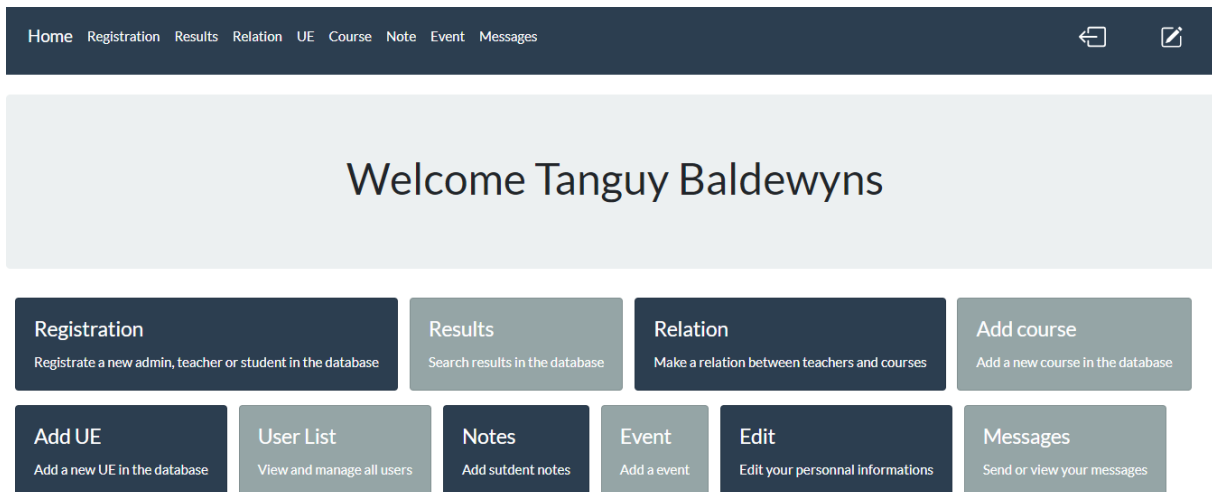


10.5 Définition de la « class » en fonction du « type » :

```
//Teste si il s'agit d'un étudiant
//Sinon la class est automatiquement 'NONE'
if($user->getType() != 'ROLE_USER'){
    $user->setClass('NONE');
}
```

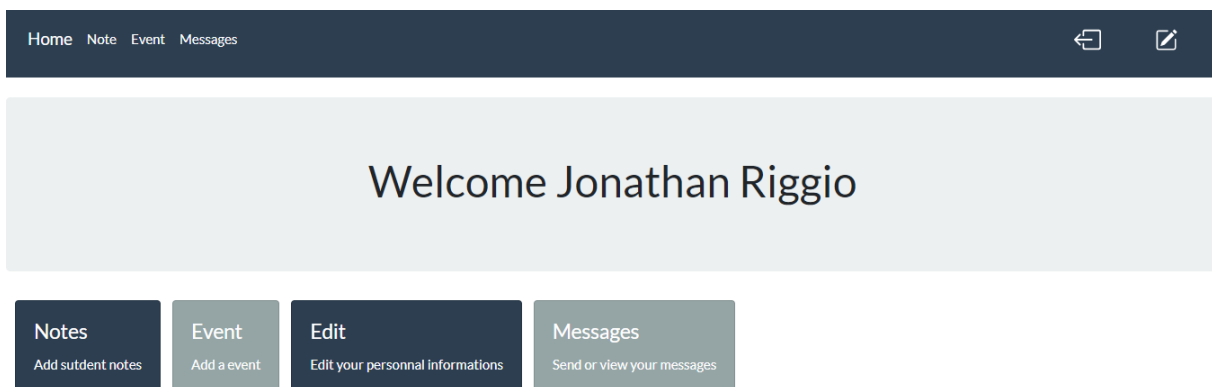
10.6 Entièreté de la page d'accueil par statuts :

Administrateur :



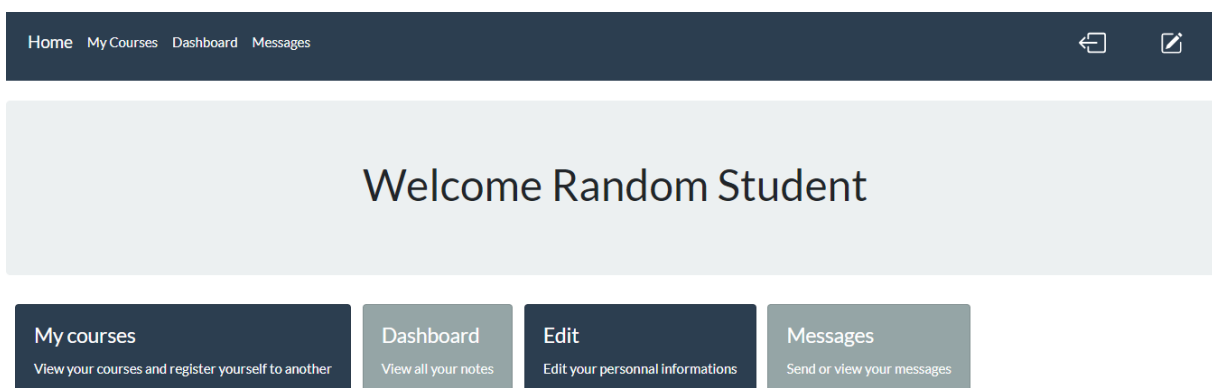
The Administrator dashboard features a dark blue header with navigation links: Home, Registration, Results, Relation, UE, Course, Note, Event, and Messages. On the right of the header are icons for a left arrow and a pencil. The main content area has a light gray background with the text "Welcome Tanguy Baldewyns". Below this, there are eight buttons arranged in two rows. The first row contains: "Registration" (dark blue, "Register a new admin, teacher or student in the database"), "Results" (light gray, "Search results in the database"), "Relation" (dark blue, "Make a relation between teachers and courses"), and "Add course" (light gray, "Add a new course in the database"). The second row contains: "Add UE" (dark blue, "Add a new UE in the database"), "User List" (light gray, "View and manage all users"), "Notes" (dark blue, "Add student notes"), "Event" (light gray, "Add a event"), "Edit" (dark blue, "Edit your personal informations"), and "Messages" (light gray, "Send or view your messages").

Professeur :



The Professor dashboard features a dark blue header with navigation links: Home, Note, Event, and Messages. On the right of the header are icons for a left arrow and a pencil. The main content area has a light gray background with the text "Welcome Jonathan Riggio". Below this, there are four buttons arranged in a single row: "Notes" (dark blue, "Add student notes"), "Event" (light gray, "Add a event"), "Edit" (dark blue, "Edit your personal informations"), and "Messages" (light gray, "Send or view your messages").

Étudiant :



The Student dashboard features a dark blue header with navigation links: Home, My Courses, Dashboard, and Messages. On the right of the header are icons for a left arrow and a pencil. The main content area has a light gray background with the text "Welcome Random Student". Below this, there are four buttons arranged in a single row: "My courses" (dark blue, "View your courses and register yourself to another"), "Dashboard" (light gray, "View all your notes"), "Edit" (dark blue, "Edit your personal informations"), and "Messages" (light gray, "Send or view your messages").

10.7 Ajouter des restrictions à des champs d'un formulaire :

```
use Symfony\Component\Validator\Constraints as Assert;
```

Précise que la valeur minimum est 0 et que la valeur maximum est 20

```
/**
 * @ORM\Column(type="integer")
 * @Assert\Range(
 *     min = 0,
 *     max = 20,
 *     notInRangeMessage = "You must be between {{ min }} and {{ max }} to enter",
 * )
 */
private $Note;
```

(La ligne @ORM\Column(type=integer) signifie que la variable « \$Note » est un champ de la base de données).

Test en rentrant une valeur non permise

Note

ERROR You must be between 0 and 20 to enter

10.8 Déploiement des graphiques au sein de Symfony :

```
//Récupère les notes de l'étudiant dans la base de données en tenant de l'application ou non du filtre
$notes = $noteRepo->findNotesBySearch($user, $search);
//Tableau contenant les cours
$noteCourse = [];
//Tableau contenant les note
$noteNote = [];
//Tableau contenant les couleurs des colonnes
$noteColor = [];
//Variables permettant de calculer la moyenne des notes sélectionnées
$totalNote = 0;
$nbrNotes = 0;
$mediumNote = 0;

//Répartition des valeurs dans les bons tableaux
foreach ($notes as $note) {
    //Cours
    $noteCourse[] = $note->getCourse();
    //Note
    $noteNote[] = $note->getNote();
    //Couleurs
    if ($note->getNote() < 10){
        //Note en dessous de la moyenne
        $noteColor[] = "darkred";
    }else{
        //Note au dessus de la moyenne
        $noteColor[] = "#132d46";
    }
    //Addition des valeurs des notes
    $totalNote += $note->getNote();
    //Calcul du nombre de note
    $nbrNotes ++;
}
//S'il y'a au moins une note
if ($nbrNotes > 0){
    //Calcul de la moyenne
    $mediumNote = $totalNote / $nbrNotes;
}

//Revoie des différent tableau pour qu'ils soient utilisable dans le fichier twig
'noteCourse' => json_encode($noteCourse),
'noteNote' => json_encode($noteNote),
'noteColor' => json_encode($noteColor),
'notes' => $notes,
```

Comme les graphiques sont utilisés plusieurs fois de la même manière (Dashboard et userprofile). Il a été décidé de créer un Template qui va être appelé plusieurs fois pour éviter la duplication de code.

```
{% block javascripts %}
<script>
    let categories = document.querySelector("#chart")
    let categGraph = new Chart(categories, {
        type: "bar",
        data: {
            labels: {{ noteCourse|raw }},
            datasets: [{
                label: "Notes",
                data: {{ noteNote|raw }},
                backgroundColor: {{ noteColor|raw }}
            }]
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true,
                        min: 0,
                        max: 20
                    }
                }]
            }
        }
    })
</script>
{% endblock %}
```

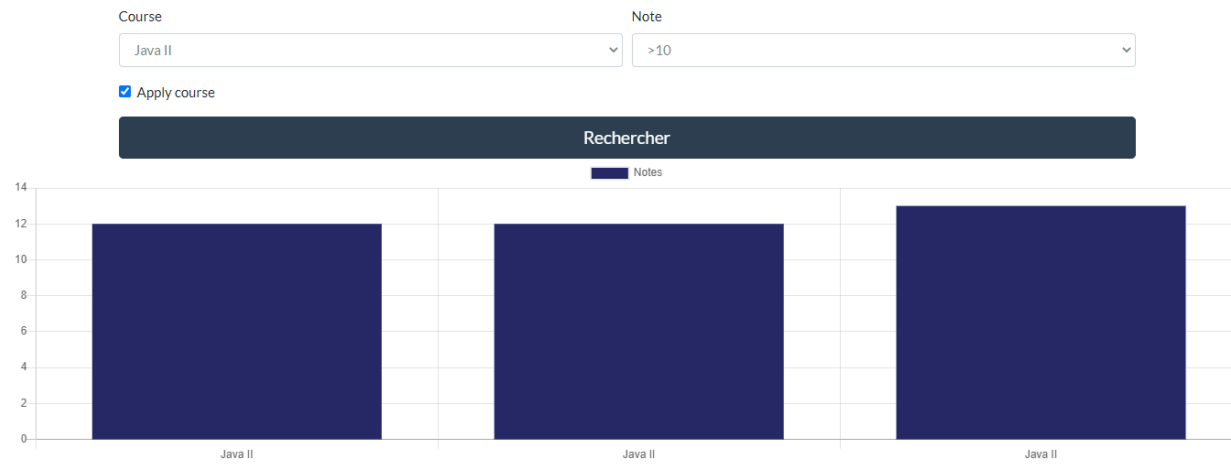
```
{% block javascripts %}
[[ include "student/javascriptDashboard.html.twig" %]]
{% endblock %}
```

Script javascript permettant le dessin du graphique en bar.

Appel du Template permettant le dessin du graphique en bar.

10.9 Filtrer les éléments à afficher dans le diagramme des notes :

Exemple : afficher les notes au-dessus de la moyenne pour le cours de Java



Fonction permettant de mettre en place le système de filtre

```
public function findNotesBySearch($user, NotesPropertySearch $search)
{
    //Création de la requête
    $query = $this->createQueryBuilder('n')
    //Filtre selon l'utilisateur authentifié
    ->andWhere('n.User = :username')
    ->setParameter('username', $user);
    //Applique le filtre concernant le cours seulement si la case "apply" est cochée
    if($search->getCourse() && $search->getApply()){
        $query = $query
        ->andWhere('n.Course = :course')
        ->setParameter('course', $search->getCourse());
    }
    //Applique le filtre des notes si la valeur de "notes" est égale à "<10" ou ">10"
    //Le filtre ne s'applique pas si la valeur de "notes" est égale à "NONE"
    if($search->getNote() == '<10'){
        $query = $query
        ->andWhere('n.Note < 10');
    }else if($search->getNote() == '>10'){
        $query = $query
        ->andWhere('n.Note >= 10');
    }
    $query = $query
    ->orderBy('n.User', 'ASC')
    ->getQuery()
    ->getResult();
    ;
    return $query;        ;
}
```

10.10 Filtrer les notes pour n'avoir accès qu'à ses notes en tant qu'étudiant :

```
public function findByCurrentUser($currentUser)
{
    //Création de la requête
    return $this->createQueryBuilder('n')
        //Chercher toutes les lignes ou le "User" est egal a l'étudiant authentifié dans la base de données
        ->andWhere('n.User = :username')
        ->setParameter('username', $currentUser)

        ->orderBy('n.id', 'DESC')
        ->getQuery()
        ->getResult()
    ;
}
```

10.11 Vérification de la clé d'inscription à un cours

```
//Récupération du cours dans la table "courses"
$course = $courseRepo->findOneByName($regist->getCourse());

//Vérifie si le formulaire est soumis (Si le bouton "register" est pressé)
if($form->isSubmitted() && $form->isValid()){

    $regist->setStudent($this->getUser()->getUsername());

    $manager->getManager()->persist($regist);

    //Comparasion de la clé du cours et de la clés entrée par l'étudiant
    if ($course->getCourseKey() != $regist->getCourseKey()){
        //Afficher ce message si les deux champs ne sont pas égaux
        $this->addFlash("danger", "Wrong key" );
        return $this->redirectToRoute('registrationToCourse');
    }
    //Ligne permettant l'integration de la nouvelle inscription dans la base de données
    $manager->getManager()->flush();
    //Affiche ce message si les deux champs sont égaux
    $this->addFlash("success", "You are registered" );
    return $this->redirectToRoute('registrationToCourse');
```