

# Rapport d'activité

## PPE2 - Quiz-Dubois

### Courréjou Matthieu

#### Table des matières

Introduction.....	2
1. Résumé du contexte .....	2
2. Mission globale à effectuer .....	2
Etapes de développement.....	2
1. Première étape : Préparer l'environnement de travail.....	2
1. Mise en place de l'environnement logiciel Back-end .....	3
2. Mise en place de l'environnement logiciel Front-end .....	5
3. Installation de logiciels de test .....	6
2. Etape 2 : Mise en place de l'application Symfony et des dépôts distants .....	6
1. Mise en place application Symfony .....	6
2. Mise en place dépôts distant.....	7
Chaque commit est accompagné d'un message avec 4 points : .....	8
3. Etape 3 : Création du modèle de l'application et génération de la documentation technique ..	8
a. Modèle de l'application .....	8
b. Documentation technique.....	12
4. Etape 4 : Codes les fonctionnalités de l'application à partir des cas d'utilisation .....	13
a. Résumé des cas d'utilisation.....	13
b. Cas d'utilisation 1 (Se connecter à l'application) : .....	17
c. Cas d'utilisation 2 : Créer un quiz .....	20
d. Cas d'utilisation 3 : Créer une évaluation à partir d'un quiz.....	22
e. Cas d'utilisation 5 : Participer à une évaluation .....	23
f. Cas d'utilisation 6 : Corriger une évaluation.....	26
g. Cas d'utilisation 7 : Consulter sa copie .....	29
h. Bilan de l'étape .....	31
5. Quatrième étape : Documentation, Déploiement et mise en production .....	31
a. Documentation .....	31
b. Déploiement et mise en production .....	32
i. Déploiement sur heroku (back) : .....	32

ii. Déploiement sur Netlify (front) : .....	34
c. Sauvegardes.....	36
6. Conclusion du projet .....	37

## Introduction

### 1. [Résumé du contexte](#)

Le centre de formation « Formations Dubois », proposant des formations en ligne ainsi qu'en présentiel à ses apprenants. La pandémie du COVID-19, et les confinements qui en découlent ont pris de cours le centre, qui a tant bien que mal tenté d'assurer ses sessions de formation. Une des plus grandes difficultés fut l'évaluation des élèves, la plupart des plateformes étant coûteuses, ou ne correspondant pas aux besoins de l'entreprise.

### 2. [Mission globale à effectuer](#)

Actuellement en poste de développeur au service recherche et développement du centre de formation Dubois, J'ai pour mission le développement d'une application web, permettant aux professeurs de créer et administrer des évaluations, auxquelles les élèves peuvent participer pendant une période donnée. Le centre de formation souhaite à terme pouvoir créer une application mobile, voir de bureau sur le même principe, c'est pourquoi le back-end sera séparé du front-end pour permettre à d'autres clients d'être greffés dans le futur.

Afin d'accomplir cette tâche, il faut déjà choisir la pile applicative qui sera utilisée au cours du développement. J'ai personnellement fait le choix de coder l'application en Typescript, avec le framework React, assurant ainsi une solidité de l'application, ainsi qu'un grand gain de temps de développement. Concernant la partie back-end, je me suis orienté vers une application PHP, sur le framework Symfony, réputé pour sa très grande stabilité, et permettant encore une fois d'accélérer grandement le temps de développement.

## Etapes de développement

### 1. [Première étape : Préparer l'environnement de travail](#)

Ici il n'y a pas d'IDE obligatoire, comme cela pourrait l'être avec un projet WinForms et VisualStudio, c'est pourquoi j'ai choisi d'utiliser ceux avec lesquels je suis le plus à l'aise : WebStorm et PhpStorm.

Les deux projets seront soumis à des standards de code rigoureux, forcés par l'utilisation d'ESLint, et de PHPCodeSniffer.

## 1. Mise en place de l'environnement logiciel Back-end

Pour cette première étape, il va tout d'abord falloir mettre en place l'environnement de travail. Voici les différentes étapes pour ce faire :

Nous allons d'abord nous occuper de la partie back-end, et pour ce faire nous allons installer la dernière version de docker, ici la version windows en suivant la documentation sur le site de l'éditeur.

## Install Docker Desktop on Windows

Welcome to Docker Desktop for Windows. This page contains information about Docker Desktop for Windows system requirements, download URL, instructions to install and update Docker Desktop for Windows.

Docker Desktop for Windows

For checksums, see [Release notes](#)

### Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than \$10 million USD in annual revenue) requires a paid subscription.

## System requirements

Your Windows machine must meet the following requirements to successfully install Docker Desktop.

WSL 2 backend

[Hyper-V backend and Windows containers](#)

### WSL 2 backend

- Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit: Home or Pro 21H1 (build 19043) or higher, or Enterprise or Education 20H2 (build 19042) or higher.
- Enable the WSL 2 feature on Windows. For detailed instructions, refer to the [Microsoft documentation](#).
- The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:
  - 64-bit processor with [Second Level Address Translation \(SLAT\)](#)
  - 4GB system RAM

*Remarque : Il vous faudra probablement installer le noyau linux intégré à windows nommé WSL2*

Une fois chose faite, nous allons créer un fichier docker compose, nous permettant d'avoir un container avec Mysql et Phpmyadmin :

```

docker-compose.yml: The compose specification establishes a
version: '3.1'
services:
  db:
    image: mysql:latest
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: ticket_io
    ports:
      - "3306:3306"
  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    restart: always
    environment:
      PMA_HOST: db
      PMA_USER: root
      PMA_PASSWORD: root
    ports:
      - "8100:80"


```

Il nous suffira alors de se rendre dans le dossier contenant le fichier docker-compose.yml, et d'exécuter la commande « docker compose up -d ». Nous avons maintenant accès à une base de données MySQL ainsi qu'à phpMyAdmin.

Nous allons ensuite installer tout le nécessaire pour Symfony, en suivant la documentation. (voir : <https://symfony.com/doc/current/setup.html>).







## Installing & Setting up the Symfony Framework

[Edit this page](#)

 Do you prefer video tutorials? Check out the [Harmonious Development with Symfony](#) screencast series.

### Technical Requirements

Before creating your first Symfony application you must:

- Install PHP 8.1 or higher and these PHP extensions (which are installed and enabled by default in most PHP 8 installations):  ctype,  iconv,  PCRE,  session,  SimpleXML and  tokenizer;
- [Install Composer](#), which is used to install PHP packages.

Optionally, you can also [install Symfony CLI](#). This creates a binary called 'symfony' that provides all the tools you need to develop and run your Symfony application locally.

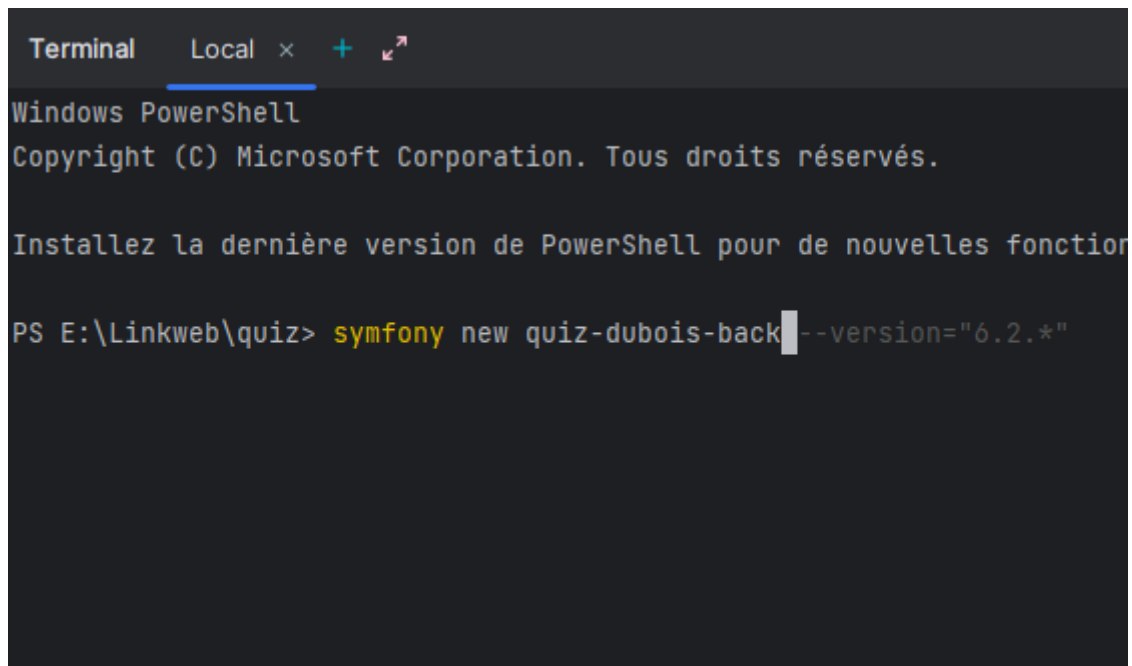
The 'symfony' binary also provides a tool to check if your computer meets all requirements. Open your console terminal and run this command:

```

$ symfony check:requirements

```

Nous avons alors tout le nécessaire pour créer l'application Symfony, ce que nous allons faire de la manière suivante :



```
Terminal Local × + ↵
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités

PS E:\Linkweb\quiz> symfony new quiz-dubois-back --version="6.2.*"
```

Nous installons ensuite tous les paquets composer qui seront nécessaires au développement du projet (voir composer.json dans le projet dubois-quiz-back).

## 2. Mise en place de l'environnement logiciel Front-end

Cette étape va être très simple, il nous suffira d'installer la dernière version de nodeJs :

Node.js® is an open-source, cross-platform JavaScript runtime environment.

Download for Windows (x64)

18.16.0 LTS

Recommended For Most Users

20.2.0 Current

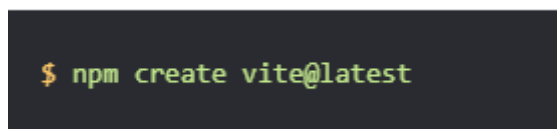
Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

For information about supported releases, see the [release schedule](#).

Et d'exécuter la commande suivante dans le répertoire où l'on souhaite créer l'application :



```
$ npm create vite@latest
```

Nous installons ensuite tous les paquets npm nécessaires au développement du projet (voir package.json sur l'application client)

### 3. Installation de logiciels de test

La partie Back-end sera développée au préalable de la partie front-end, il faudra donc un environnement de test http, afin d'effectuer des test fonctionnels et de non-régression sur l'API.

Je décide ici d'utiliser le logiciel Postman.

**Bilan : Nous avons à présent tous les outils nécessaires pour commencer le développement du projet, ainsi que tester ce dernier. Ici la seule difficulté potentielle est l'installation de PHP sur windows, où certaines étapes ne sont pas forcements spécifiées dans la documentation pouvant aboutir à des messages d'erreur.**

## 2. Etape 2 : Mise en place de l'application Symfony et des dépôts distants

### 1. Mise en place application Symfony

Il est primordial qu'une application côté serveur journalise les évènements qu'elle produit. C'est pourquoi nous allons installer et paramétrer Monolog sur l'application Symfony, en définissant les niveaux de log et où ces derniers vont être stockés.

```
monolog:
  channels:
    - deprecation # Deprecations are logged in the dedicated "deprecation" channel when it exists

when@dev:
  monolog:
    handlers:
      main:
        type: stream
        path: "%kernel.logs_dir%/%kernel.environment%.log"
        level: info
        channels: ["!event"]
        # uncomment to get logging in your browser
        # you may have to allow bigger header sizes in your Web server configuration
        #firephp:
        #   type: firephp
        #   level: info
        #chrome:
        #   type: chrome
        #   level: info
      console:
        type: console
        process_psr_3_messages: false
        channels: ["!event", "!doctrine", "!console"]
```

*Remarque : En environnement de développement, les logs seront stockés dans le dossier /var/log. Cependant en production, ils seront envoyé à la console php://stderr, il faudra alors paramétrer votre environnement de production.*

Concernant la sécurité et l'authentification, l'application back-end utilisera des JWT (JSON web tokens), qui seront stockés par les user-agent en tant que cookies. Il faut aussi paramétrer les CORS, afin d'autoriser les requêtes provenant du domaine de notre application cliente. Nous installons alors les bundle Symfony nous permettant de faire cela, et voici leurs configurations :

*Voir /config/packages/nelmio\_cors.yaml*

```

nelmio_cors:
  defaults:
    allow_origin: ['*']
    allow_methods: ['GET']
    allow_headers: ['*']
    expose_headers: ['Link']
    max_age: 3600
  paths:
    '^/api/':
      origin_regex: true
      allow_origin: ['%env(CORS_ALLOW_ORIGIN)%']
      allow_methods: ['GET', 'OPTIONS', 'POST', 'PUT', 'PATCH', 'DELETE']
      allow_headers: ['Content-Type', 'X-API-Key', 'X-Window-Id']
      allow_credentials: true

```

Voir `/config/packages/lexik_jwt_authentication.yaml`

```

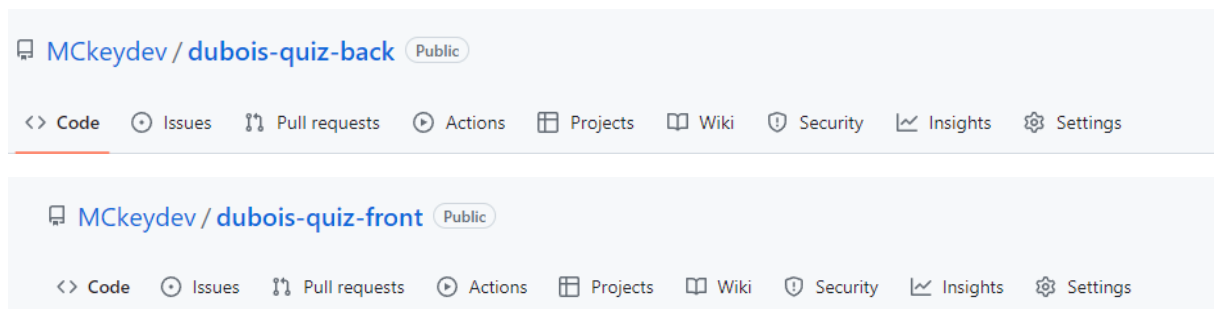
lexik_jwt_authentication:
  secret_key: '%env(resolve:JWT_SECRET_KEY)%'
  public_key: '%env(resolve:JWT_PUBLIC_KEY)%'
  pass_phrase: '%env(JWT_PASSPHRASE)%'
  token_ttl: 604800 # in seconds, default is 3600
  token_extractors:
    # check token in a cookie
    split_cookie:
      enabled: true
      cookies:
        - jwt_hp
        - jwt_s
  set_cookies:
    jwt_hp:
      lifetime: null
      samesite: none
      secure: true
      path: /
      domain: null
      httpOnly: true
      split:
        - header
        - payload

```

## 2. Mise en place dépôts distant

Ces deux dépôts seront tous deux hébergés sur Github. Afin de gérer de manière plus aisée les différentes branches et versions du projet, je décide d'utiliser le logiciel GitKraken, car c'est celui que j'ai l'occasion d'utiliser tous les jours au cours de mon alternance.

Voici les deux dépôts :



Chaque modification sera poussée selon une nomenclature précise :

```
[++][UPD][OutputGroups] Fixed a few output groups
[+][UPD][Evaluation] Changed output groups
[+][UPD][Evaluation] Removed unused code, and modified normalization groups
[++][UPD][Evaluation] WIP2
[+][UPD][EvaluationRepository] WIP
[++][UPD][CopyPreview] Added missing data in copy previews
[+++][ADD][API] API is almost done
[++][UPD][SplitCookies] Changed security to JWT split cookies
[++][UPD][Login] Added user info to login success response
[++][UPD][CRUD] Added a few routes.
[+++][UPD][Relations] Modified relations
```

Chaque commit est accompagné d'un message avec 4 points :

- L'importance du commit
- Le type d'opération
- La partie de l'application modifiée
- La description du commit

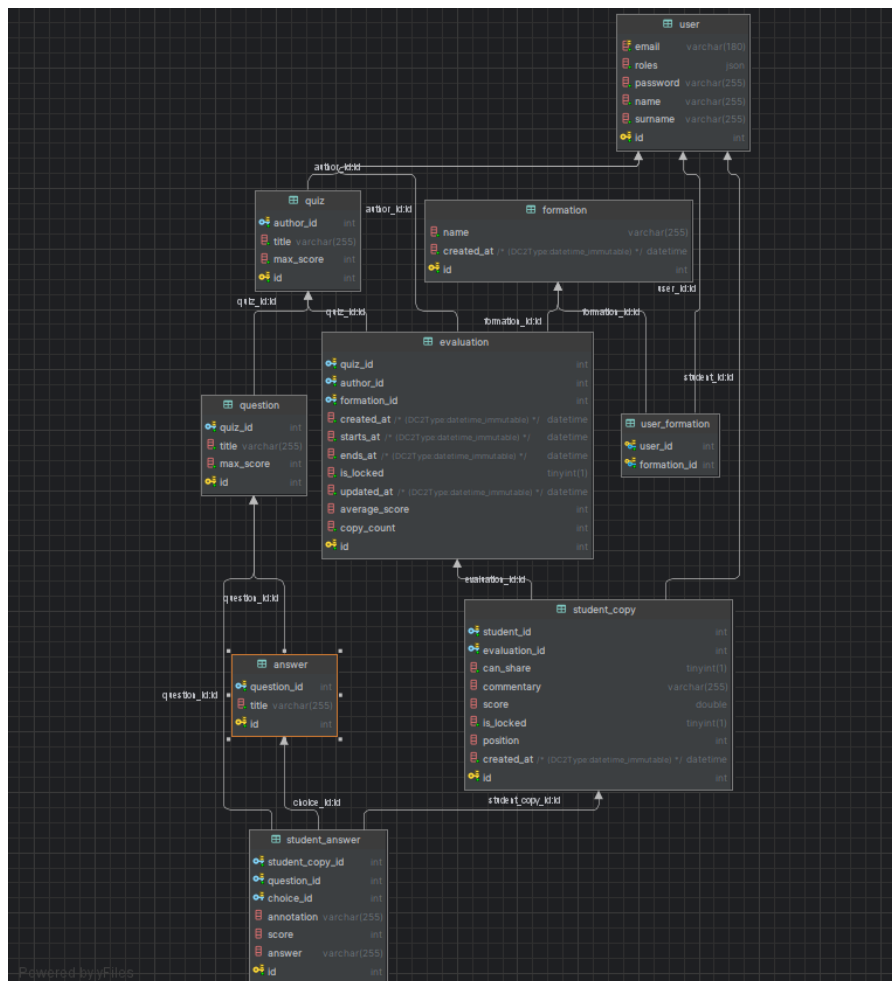
**Bilan : Les dépôts distants sont dorénavant prêts, et le projet symfony est sécurisé et journalisé.**

### 3. [Etape 3 : Création du modèle de l'application et génération de la documentation technique](#)

#### a. Modèle de l'application

Symfony utilise un ORM appelé Doctrine, cela signifie que toutes mes tables seront représentées sous forme d'objet, et qu'il devrait en logique pas être nécessaire de manipuler directement les tables de la base. Voici le modèle de données souhaité pour l'application :





Nous allons commencer par créer un utilisateur avec seulement les droits nécessaires, sur la base, afin de sécuriser au maximum l'application :

```
CREATE USER 'app_dubois_quiz'@'localhost' IDENTIFIED BY 'azdCVBxc23$*qQ';
```

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP ON dubois_quiz.* TO
'app_dubois_quiz'@'localhost';
```

Maintenant que nous avons un accès sécurisé à la base de données, nous créons un fichier `.env.local` ou nous spécifions toutes nos variables d'environnement, et notamment la chaîne de connexion à la base de données :

```

###> symfony/framework-bundle ###
APP_ENV=dev
APP_SECRET=38e355e1bcdefb87cfc2692f01df5afa
###< symfony/framework-bundle ###

###> doctrine/doctrine-bundle ###
# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#con
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
#
# DATABASE_URL="sqlite:///kernel.project_dir%/var/data.db"
# DATABASE_URL="mysql://app_dubois_quiz:azdCVBxc23$*qQ@127.0.0.1:3306/dubois-quiz?serverVersion=8&charset=utf8mb4"
# DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8"
###< doctrine/doctrine-bundle ###

###> lexik/jwt-authentication-bundle ###
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
JWT_PASSPHRASE=dc197904a344dbba4855ec89fb2ba557
###< lexik/jwt-authentication-bundle ###

###> nelmio/cors-bundle ###
CORS_ALLOW_ORIGIN='^https?://(localhost|127\.0\.0\.1)(:[0-9]+)?$'
###< nelmio/cors-bundle ###

```

Maintenant, il va falloir créer les tables. Or pour ce faire nous allons passer directement par le code, en créant la partie Model de l'application, sous la forme d'objets nommés « Entités ». Symfony dispose de commandes permettant la création de ces objets, nous allons alors créer tous les objets correspondants à toutes les tables du modèle souhaité.

```

PS E:\Linkweb\quiz> symfony console make:entity

Class name of the entity to create or update (e.g. OrangeKangaroo):
> Table

You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> propriété1

Field type (enter ? to see all types) [string]:
>

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Table.php

Add another property? Enter the property name (or press <return> to stop adding fields):
>

Success!

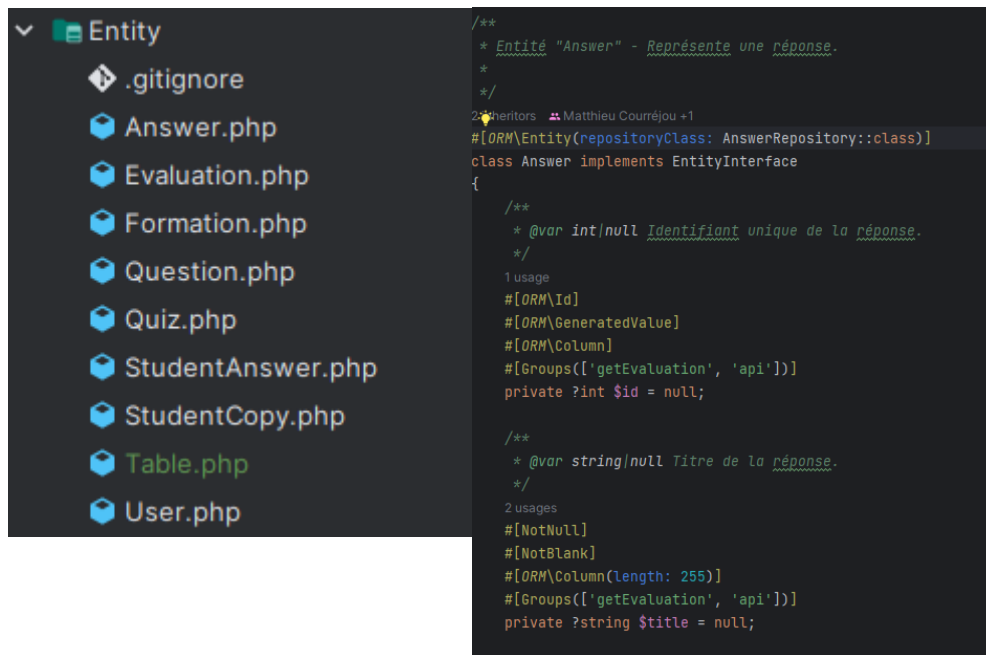
```

*Exemple de création d'une entité.  
Pour que la base de données  
corresponde à cette entité, il faudra  
effectuer des migrations avec les  
commandes suivantes :*

*Symfony console make :migration*

*Symfony console d :m :m*

Une fois toutes les entités créées nous nous retrouvons avec ce modèle :



Et voici à quoi ressemble la base de données (visualisée à l'aide de PhpMyAdmin) :

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> answer	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> evaluation	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	64,0 kio	-
<input type="checkbox"/> formation	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> question	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> quiz	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> student_answer	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	64,0 kio	-
<input type="checkbox"/> student_copy	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> user	Parcourir Structure Rechercher Insérer Vider Supprimer	6	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> user_formation	Parcourir Structure Rechercher Insérer Vider Supprimer	6	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<b>9 tables</b>	<b>Somme</b>	<b>14</b>	<b>InnoDB</b>	<b>utf8mb4_0900_ai_ci</b>	<b>368,0 kio</b>	<b>0 0</b>

Les classes d'accès à la base de données, se nomment des « Repositories ». Il y en a autant qu'il y a d'entités, et ces dans ces derniers que vont être stockées toutes les fonctions d'action en base de données, voici un exemple :

```
/**
 * @extends ServiceEntityRepository<StudentCopy>
 *
 * @method StudentCopy|null find($id, $lockMode = null, $lockVersion = null)
 * @method StudentCopy|null findOneBy(array $criteria, array $orderBy = null)
 * @method StudentCopy[] findAll()
 * @method StudentCopy[] findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
// Matthieu Courréjou +1
class StudentCopyRepository extends ServiceEntityRepository
{
    // Matthieu Courréjou
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, StudentCopy::class);
    }

    // Matthieu Courréjou
    public function save(StudentCopy $entity, bool $flush = false): void
    {
        $this->getEntityManager()->persist($entity);

        if ($flush) {
            $this->getEntityManager()->flush();
        }
    }

    // Matthieu Courréjou
    public function remove(StudentCopy $entity, bool $flush = false): void
    {
        $this->getEntityManager()->remove($entity);
    }
}
```

## b. Documentation technique

Afin que d'autres équipes de développeurs puissent travailler avec aise sur le projet, il est nécessaire de créer une documentation technique. Pour ce faire j'ai décidé d'en générer une automatiquement à l'aide du logiciel PhpDocumentor, et des commentaires normalisés PHPDoc présents dans tout le projet.

Il suffit de télécharger Phpdocumentor au format Phar, et d'exécuter la commande suivante :

```
PS E:\Linkweb\quiz> php phpDocumentor.phar run -d E:\Linkweb\quiz\src -t E:\Linkweb\quiz\doc
```

Remarque : « -d » est le répertoire du projet à scanner, « -t » est le répertoire de destination de la doc

Voici ce que nous obtenons :

The screenshot shows a phpDoc documentation page for a project named 'App'. The page has a green header with the title 'App' and a search bar. The left sidebar contains a navigation menu with sections: 'Namespaces' (listing App, Command, Controller, DataFixtures, DTO, Entity, EventListener, Factory, Interfaces, Repository, Service), 'Packages' (listing Application), 'Reports' (listing Deprecated, Errors, Markers), and 'Indices' (listing Files). The main content area is titled 'App' and 'Namespaces', listing various classes and interfaces with green circular icons: Command, Controller, DataFixtures, DTO, Entity, EventListener, Factory, Interfaces, Repository, and Service. Below this, there is a section titled 'Interfaces, Classes, Traits and Enums' with a green circular icon and the text 'Kernel'.

Cette documentation sera mise à jour tout au long du projet, à chaque ajout de fonctionnalité ou de classe, il ne faudra alors pas oublier de documenter le code à l'aide de commentaires phpDoc.

**Bilan : La base de données est créée et prête au développement, aussi la documentation technique est générée, nous sommes alors prêt à commencer le développement des fonctionnalités de l'application.**

#### 4. Etape 4 : Codes les fonctionnalités de l'application à partir des cas d'utilisation

##### a. Résumé des cas d'utilisation

Voici tous les cas d'utilisations spécifiés par le cahier des charges. Chaque fonctionnalité sera développée de la sorte :

- Développement de la fonctionnalité sur l'application Symfony
- Test de la fonctionnalité sur Postman
- Test de non-régression de toutes les fonctionnalités précédentes (avec une batterie de tests postman)
- Développement de la fonctionnalité sur l'application cliente
- Test de non-régression des autres fonctionnalités côté client

Afin de ne pas alourdir le dossier plus que nécessaire, je ne vais pas indiquer tous les tests postman de chaque fonctionnalité, vous pouvez tout de même aller voir la collection de test du projet, sur le lien suivant : <https://ticket-io.postman.co/workspace/Team-Workspace~b2a59d48-203a-4aa0-80eb-bc691672742a/collection/20766209-24cb22fa-17fa-4955-8bef-98c2ffcf547b?action=share&creator=20766209>

#### Cas d'utilisation n°1 : Se connecter

Acteur	Utilisateur (élève et formateur)
Événement déclencheur	Aucun
Intérêt	Accéder aux fonctionnalités de création/participation au quiz
Précondition	Aucune
Scénario nominal	1. Le système affiche les zones de saisie du login et du mot de passe. 2. Le responsable saisit son login, son mot de passe et valide. 3. Le système authentifie les informations du responsable et affiche la liste des personnels et les boutons de commande.
Scénario alternatif	3a. Les champs ne sont pas tous remplis : retour au point 2. 3b. Le login et/ou le mot de passe ne correspondent pas à ceux enregistrés dans la base de données : retour au point 2.

#### Cas d'utilisation n°2 : Créer un quiz

Acteur	Formateur
Événement déclencheur	Formateur souhaite créer un nouveau quiz
Intérêt	Permettre au formateur de créer des quiz avec des questions (QCM, choix libre, etc.)
Précondition	Le formateur est connecté au système
Scénario nominal	1. Le formateur accède à l'interface de création de quiz 2. Le système affiche les options de paramétrage du quiz, telles que le titre, la durée, les types de questions, etc.

<b>Acteur</b>	<b>Formateur</b>
Scénario alternatif	2a. Le formateur ne remplit pas tous les champs requis : affichage d'une alerte et retour au point 2.

### Cas d'utilisation n°3 : Créer une évaluation à partir d'un quiz

<b>Acteur</b>	<b>Formateur</b>
Événement déclencheur	Le formateur souhaite créer une évaluation basée sur un quiz existant
Intérêt	Permettre au formateur de créer des évaluations en associant un quiz à une formation spécifique
Précondition	Le formateur est connecté au système et le quiz à utiliser existe
Scénario nominal	1. Le formateur accède à l'interface de création d'évaluation
	2. Le système affiche la liste des quiz existants disponibles pour la création d'une évaluation
Scénario alternatif	-

### Cas d'utilisation n°4 : Modifier les dates d'une évaluation

<b>Acteur</b>	<b>Formateur</b>
Événement déclencheur	Le formateur souhaite modifier les dates de début et de fin d'une évaluation
Intérêt	Permettre au formateur d'ajuster les dates d'une évaluation pour répondre aux besoins de la formation
Précondition	Aucun élève n'a encore participé à l'évaluation
Scénario nominal	1. Le formateur accède à l'interface de modification des dates d'évaluation
	2. Le système affiche les dates de début et de fin actuelles de l'évaluation
	3. Le formateur modifie les dates de début et de fin selon les besoins de la formation
	4. Le formateur enregistre les nouvelles dates d'évaluation
Scénario alternatif	-

#### Cas d'utilisation n°5 : Participer à une évaluation

Acteur	Apprenant
Événement déclencheur	L'apprenant souhaite participer à une évaluation disponible dans le cadre d'une formation spécifique
Intérêt	Permettre à l'apprenant de répondre aux questions d'une évaluation dans le cadre de sa formation
Précondition	L'évaluation est disponible pour l'apprenant selon la date et l'heure définies
Scénario nominal	1. L'apprenant accède à l'interface de l'évaluation
	2. Le système affiche les questions de l'évaluation
	3. L'apprenant répond aux questions en sélectionnant les options ou en saisissant les réponses
	4. L'apprenant soumet ses réponses
Scénario alternatif	3a. L'apprenant ne répond pas à toutes les questions obligatoires : affichage d'une alerte et retour au point 3

#### Cas d'utilisation n°6 : Corriger une évaluation

Acteur	Formateur
Événement déclencheur	L'évaluation est terminée et les apprenants ont soumis leurs réponses
Intérêt	Permettre au formateur de corriger les copies des apprenants et d'attribuer les notes correspondantes
Précondition	Le quiz a été terminé par les apprenants
Scénario nominal	1. Le formateur accède à l'interface de correction de l'évaluation
	2. Le système affiche les copies des apprenants avec les questions et les réponses fournies
	3. Le formateur ajoute des annotations et attribue des points à chaque question
	4. Le formateur saisit une annotation globale pour la copie et calcule la note globale
	5. Le formateur valide la correction de l'évaluation
Scénario alternatif	-



### Cas d'utilisation n°7 : Voir sa copie

Acteur	Apprenant
Événement déclencheur	L'évaluation du quiz est corrigée et les notes sont disponibles
Intérêt	Permettre à l'apprenant de consulter sa copie corrigée, y compris les réponses fournies, les annotations du formateur et la note attribuée
Précondition	L'évaluation du quiz a été corrigée et les notes sont disponibles
Scénario nominal	1. L'apprenant accède à l'interface de consultation de sa copie 2. Le système affiche la copie de l'apprenant avec les questions, les réponses fournies, les annotations du formateur et la note attribuée
Scénario alternatif	-

#### b. Cas d'utilisation 1 (Se connecter à l'application) :

Toute utilisations de l'application nécessite une authentification. C'est pourquoi c'est la première fonctionnalité que nous allons développer. Par chance, Symfony permet de gérer l'authentification de manière simple et sécurisée. Pour ce faire il faut déjà installer le bundle correspondant à cette fonctionnalité :

```
$ composer require symfony/security-bundle
```

Une fois chose faite, il faut créer une entité « User » qui implémente plusieurs interfaces, comme précisé dans la documentation :

```
/**
 * Entité "User" - Représente les utilisateurs de l'application.
 */
#[ORM\Entity(repositoryClass: UserRepository::class)]
class User implements UserInterface, PasswordAuthenticatedUserInterface, EntityInterface
{
```

Il faut ensuite indiquer à Symfony que c'est par cette classe que nos utilisateurs seront authentifiés :

```

security:
    enable_authenticator_manager: true
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
    providers:
        app_user_provider:
            entity:|
                class: App\Entity\User
                property: email
            # used to reload user from session & other features (e.g. switch_user)

```

Le framework va s'occuper de réceptionner les requêtes sur la route « /login », vérifier les identifiants et apporter la réponse appropriée. Ayant besoin côté client de multiples informations utilisateur, j'ai ajouté un Listener qui vient compléter la réponse de connexion réussie avec des informations :

```

/**
 * Méthode appelée lorsqu'une réponse d'authentification réussit.
 *
 * @param AuthenticationSuccessEvent $event L'événement de réussite d'authentification
 */
Matthieu Courréjou
public function onAuthenticationSuccessResponse(AuthenticationSuccessEvent $event)
{
    // Récupérer les données de l'événement
    $data = $event->getData();
    $user = $event->getUser();

    // Vérifier si l'utilisateur est une instance de UserInterface
    if (!$user instanceof UserInterface) {
        return;
    }

    // Ajouter les données supplémentaires à la réponse
    $data['data'] = [
        'id' => $user->getId(),
        'email' => $user->getUserIdentifier(),
        'roles' => $user->getRoles(),
        'name' => $user->getName(),
        'surname' => $user->getSurname(),
    ];

    // Mettre à jour les données de l'événement avec les données modifiées
    $event->setData($data);
}

```

Nous allons maintenant tester la fonctionnalité avec Postman :

The screenshot shows a REST client interface with a sidebar on the left containing a tree view with folders like 'Quiz', 'Evaluation', 'StudentCopy', 'Security', and 'User'. The main area displays a POST request to '[[BaseURL]]/login'. The request body is a JSON object with 'username' and 'password' fields. Below the request, the 'Body' tab is selected, showing the response in JSON format. The response is a JSON object with a 'data' field containing user information.

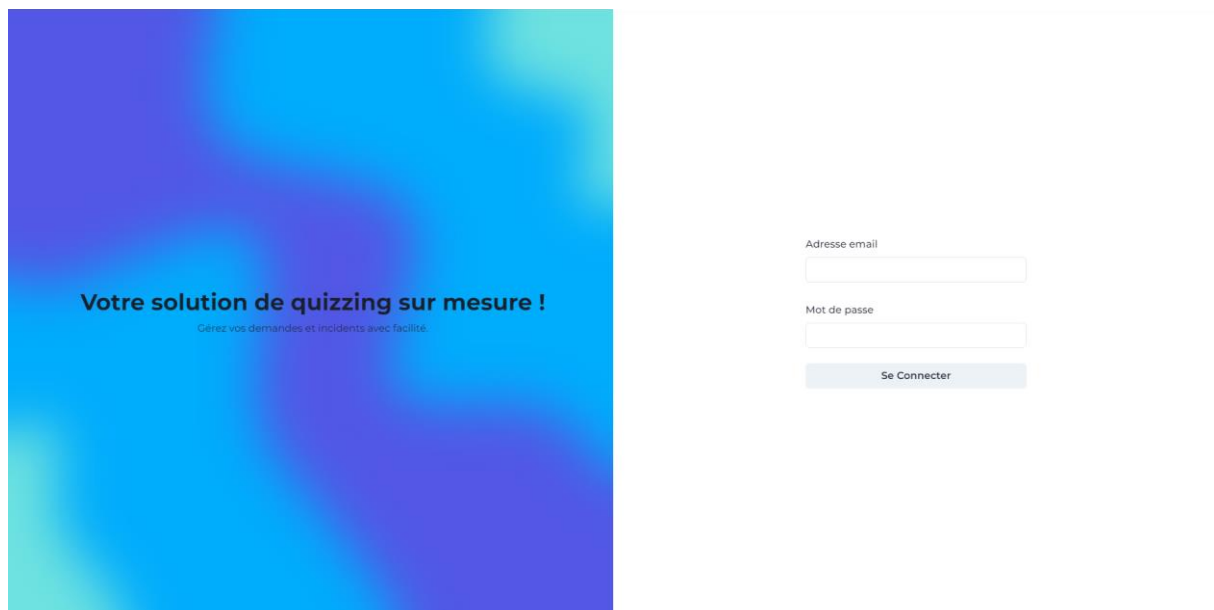
```
1 POST [[BaseURL]]/login
2
3 {
4   "username": "formateur1@quiz.com",
5   "password": "password"
6 }
7
8
9
10
11
12
```

Body Cookies (2) Headers (10) Test Results

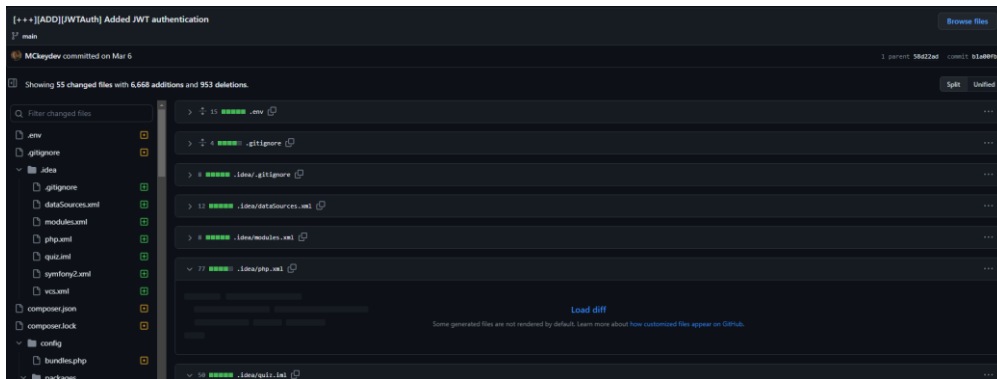
Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "id": 150,
4     "email": "formateur1@quiz.com",
5     "roles": [
6       "ROLE_FORMATEUR",
7       "ROLE_USER"
8     ],
9     "name": "Kunze",
10    "surname": "Jose"
11  }
12 }
```

Tout fonctionne parfaitement, nous allons alors pouvoir développer le formulaire côté client qui permet de se connecter depuis l'application :



*Voir composant Login.tsx*



**Bilan :** Comme à chaque fin de fonctionnalité, nous effectuons les commit sur chacun des repos, et nous mettons à jour la documentation technique.

### c. Cas d'utilisation 2 : Créer un quiz

Voici la route correspondant à cette fonctionnalité :

```
/**
 * Crée un nouveau quiz.
 *
 * @param User $user L'utilisateur formateur connecté.
 * @param Request $request L'objet Request contenant les données de la requête.
 * @param ApiRequestValidator $apiRequestValidator Le service de validation des requêtes API.
 * @param EntityManagerInterface $entityManager L'instance de l'EntityManager.
 * @return JsonResponse La réponse JSON indiquant la création réussie du quiz.
 */
#[Route('/api/quiz', name: 'app_quiz', methods: ['POST'])]
public function createQuiz(
    #[CurrentUser] User $user,
    Request $request,
    ApiRequestValidator $apiRequestValidator,
    EntityManagerInterface $entityManager,
): JsonResponse {
    $this->denyAccessUnlessGranted('ROLE_FORMATEUR');

    /**
     * @var Quiz $dto
     */
    $dto = $apiRequestValidator->checkRequestValidity($request, type: Quiz::class, isArray: false);

    // Calcule la note maximale du quiz en faisant la somme des barèmes de chaque question
    $maxScore = $dto->getQuestions()->reduce(function (int $accumulator, Question $question): int {
        return $accumulator + $question->getMaxScore();
    }, initial: 0);

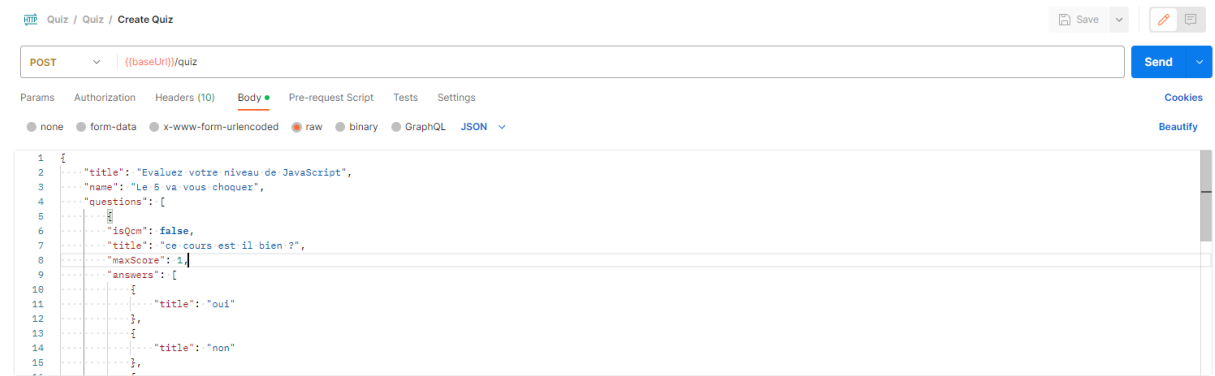
    $dto->setMaxScore($maxScore);

    $dto->setAuthor($user);
    $entityManager->persist($dto);
    $entityManager->flush();

    return new JsonResponse(['message' => 'Quiz was successfully created.', 'id' => $dto->getId()], status: Response::HTTP_CREATED);
}
```

*Le contrôleur s'assure que l'utilisateur est bien un formateur, et va valider que le requête correspond bien au model attendu. Une fois chose faite il va pouvoir créer l'entité quiz et la persister en base.*

Voici le test Postman :



Et voici l'interface côté client :

The client interface for creating a quiz consists of the following elements:

- Titre du quiz**: A text input field.
- Intitulé de la question**: A text input field.
- Points de la question**: A numeric input field with a value of 0 and a dropdown arrow.
- Ajouter un choix**: A button to add a choice.
- Intitulé de la question**: A text input field for a sub-question.
- Points de la question**: A numeric input field with a value of 0 and a dropdown arrow.
- Choix 1**: A text input field with a delete icon (X).
- Choix 2**: A text input field with a delete icon (X).
- Ajouter un choix**: A button to add a choice.
- + Ajouter une question**: A button to add a new question.

At the bottom of the interface is a button labeled **Créer le quiz**.

#### d. Cas d'utilisation 3 : Créer une évaluation à partir d'un quiz

Voici la route prenant en charge cette fonctionnalité :

```
/**
 * Crée une nouvelle évaluation.
 *
 * @param User $user L'utilisateur actuel
 * @param Request $request La requête HTTP
 * @param ApiRequestValidator $apiRequestValidator Le validateur de requête
 * @param EntityManagerInterface $entityManager Le gestionnaire d'entités
 * @param SerializerInterface $serializer Le sérialiseur
 *
 * @return JsonResponse La réponse JSON contenant l'évaluation créée
 */
#[Route('/api/evaluation/create', name: 'app_evaluation_create', methods: ['POST'])]
public function createEvaluation(
    #[CurrentUser] User $user,
    Request $request,
    ApiRequestValidator $apiRequestValidator,
    EntityManagerInterface $entityManager,
    SerializerInterface $serializer,
): JsonResponse {
    // TODO: Make this in access control
    $this->denyAccessUnlessGranted('ROLE_FORMATEUR');

    // Validates the request format
    /**
     * @var CreateEvaluationDTO $dto
     */
    $dto = $apiRequestValidator->checkRequestValidity($request, type: CreateEvaluationDTO::class, isArray: false);

    // Récupères le quiz, et vérifie s'il existe et si l'utilisateur est autorisé dessus
    $quiz = $entityManager->getRepository(Quiz::class)->find($dto->quiz);
    if (!$quiz || !$quiz->isOwner($user)) {
        return new JsonResponse(data: "Le quiz renseigné n'est pas valide", status: Response::HTTP_NOT_FOUND);
    }

    $formation = $entityManager->getRepository(Formation::class)->find($dto->formation);
    $isAllowedOnFormation = $user->getFormations()->contains($formation);
```

```
if (!$isAllowedOnFormation) {
    return new JsonResponse(data: "La formation renseignée n'est pas valide", status: Response::HTTP_NOT_FOUND);
}

$evaluation = new Evaluation();
$evaluation
    ->setStartsAt($dto->startsAt)
    ->setEndsAt($dto->endsAt)
    ->setAuthor($user)
    ->setQuiz($quiz)
    ->setFormation($formation);

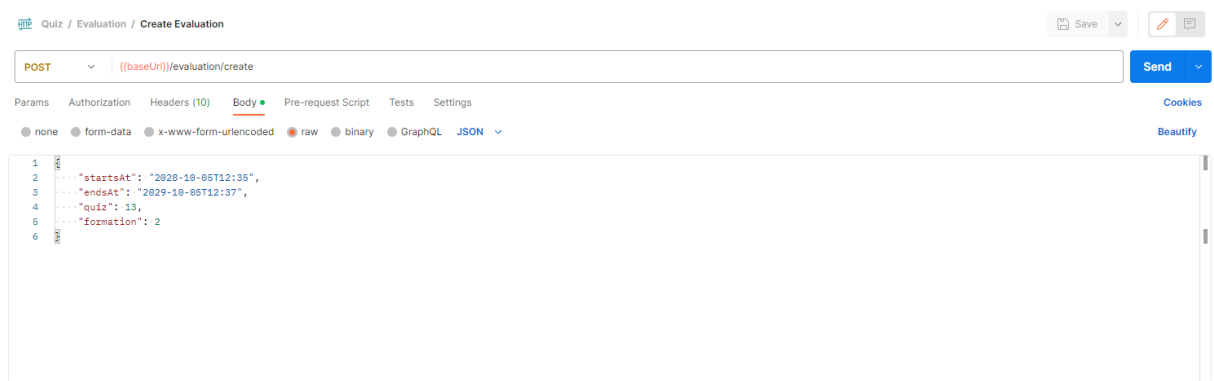
$entityManager->persist($evaluation);
$entityManager->flush();

return new JsonResponse($serializer->serialize($evaluation, format: 'json', ['groups' => 'getEvaluation']), status: Response::HTTP_CREATED);
```

Ce contrôleur n'est autorisé qu'aux formateurs. Il va d'abord vérifier que la requête correspond bien au DTO « CreateEvaluationDTO », et si c'est le cas l'instancier. Ensuite, il va vérifier si le quiz spécifié dans le DTO existe, et si l'utilisateur est bien l'auteur du quiz. Il faudra ensuite récupérer les formations de

*l'enseignant, et vérifier si la formation spécifiée dans la requête est l'une d'entre elles. Une fois toutes ces vérifications terminées, nous pouvons créer l'évaluation.*

### Test postman correspondant à la fonctionnalité :



### Interface permettant la création de l'évaluation :

A web form titled 'Créer une évaluation'. It contains four input fields: 'Quiz' with a dropdown menu showing 'Quiz test numéro 1'; 'Formation' with a dropdown menu showing 'BTS SIO SLAM'; 'Date de début' with a date picker showing 'jj/mm/aaaa --:--'; and 'Date de fin' with a date picker showing 'jj/mm/aaaa --:--'. At the bottom left is a button labeled 'Créer l'évaluation'.

### e. Cas d'utilisation 5 : Participer à une évaluation

Ce cas implique plusieurs choses :

- Que l'élève voie toutes les évaluations auquel il peut participer
- Et qu'il puisse participer et envoyer sa copie

Voici les deux routes s'occupant de ces deux cas :

```

/**
 * Affiche les informations nécessaires pour la page d'accueil d'un utilisateur élève.
 *
 * @param User $user L'utilisateur élève connecté.
 * @param EntityManagerInterface $entityManager L'instance de l'EntityManager.
 * @return JsonResponse La réponse JSON contenant les informations pour la page d'accueil de l'élève.
 */
// Matthieu Courréjou +1
public function studentHome(
    User $user,
    EntityManagerInterface $entityManager,
): JsonResponse {
    $responseData = [];

    // Formations de l'utilisateur
    $responseData['formations'] = $user->getFormations();

    $evaluationRepository = $entityManager->getRepository(Evaluation::class);

    // Evaluations en cours et à venir de l'utilisateur
    $responseData['onGoing'] = $evaluationRepository->findOngoingEvaluations($user);
    $responseData['incoming'] = $evaluationRepository->findIncomingEvaluations($user);

    return $this->json($responseData, context: ['groups' => 'api']);
}

```

Ici c'est « *findOnGoingEvaluations* » qui s'occupe de retourner à l'élève toutes les évaluations auquel il peut participer.

```

/**
 * Soumet une copie d'élève pour une évaluation.
 *
 * @param user $user L'utilisateur élève connecté
 * @param request $request L'objet Request contenant les données de la requête
 * @param evaluation $evaluation L'évaluation
 * @param entityManagerInterface $entityManager L'instance de l'EntityManager
 * @param apiRequestValidator $apiRequestValidator Le service de validation des requêtes API
 * @param validatorInterface $validator L'instance du Validator
 *
 * @return JsonResponse|Response la réponse JSON indiquant la soumission réussie de la copie
 *
 * @throws \JsonException
 */
// Matthieu Courréjou +1
#[Route('/api/evaluation/{id}/studentCopy/submit', name: 'app_student_copy_submit', methods: ['POST'])]
public function submitStudentCopy(
    #[CurrentUser] User $user,
    Request $request,
    Evaluation $evaluation,
    EntityManagerInterface $entityManager,
    ApiRequestValidator $apiRequestValidator,
    ValidatorInterface $validator
): JsonResponse|Response {
    // Récupère toutes les formations autorisées pour l'évaluation
    $formation = $evaluation->getFormation();
}

```



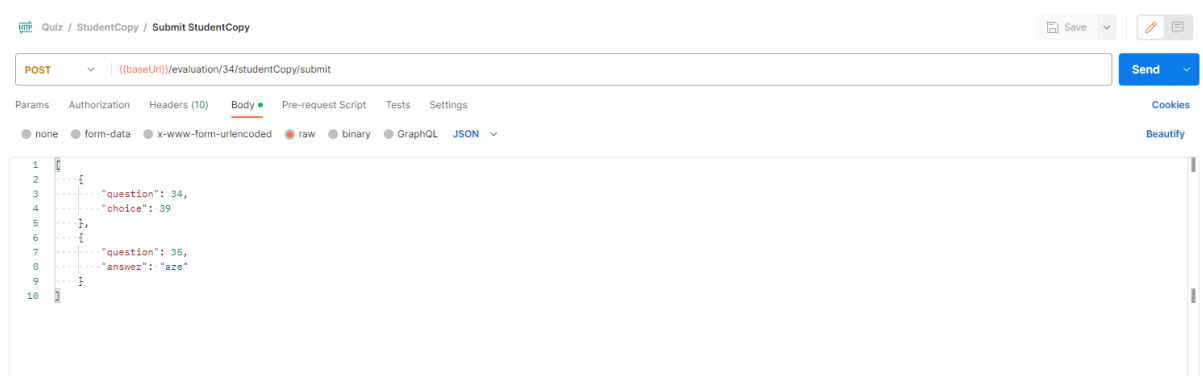
Tout le code n'est pas montré car trop long, mais ce contrôleur fait plusieurs vérifications avant de traiter le contenu de la requête :

- Que l'utilisateur est bien dans une formation visée par l'évaluation
- Que l'utilisateur n'a pas déjà fait l'évaluation
- Et que l'évaluation est en cours

Il s'assure aussi que le contenu de la requête correspond au DTO attendu.

Il va ensuite créer la copie, ainsi que chacun des réponses envoyées par l'élève.

**Voici le test postman correspondant :**



**Voici l'interface côté client :**

Vos évaluations en cours

EN COURS

Du 29 mai 2023  
au 29 mai 2023

Quiz test numéro 1

Formation

BTS SIO SLAM

>

**Quiz test numéro 1**

Quel est la bonne réponse ?

☐ a

☐ b

Question libre

Saisissez votre réponse

Terminer l'évaluation

Quiz test numéro 1

Quel est la bonne réponse ?

☒ a
☐ b

Question libre

reponse

Terminer l'évaluation

Soumettre votre copie

Étes vous sûr de vouloir envoyer votre copie ?

Cette action est irréversible, et vous ne pourrez plus la modifier.

Je me rellis

Envoyer

## f. Cas d'utilisation 6 : Corriger une évaluation

De la même manière que le cas précédent, cette fonctionnalité implique plusieurs choses :

- Que le formateur voie toutes les copies qu'il doit corriger
- Qu'il puisse cliquer sur la copie et arriver sur la page de correction
- Qu'il valide la correction et l'envoie au serveur

Voici la fonction qui permet de récupérer en base les évaluations correspondantes :

```

/**
 * Récupère les évaluations du formateur dont la date de fin est passée et pour lesquelles au moins une copie non corrigée existe.
 *
 * @param User $user L'utilisateur formateur pour lequel rechercher les évaluations à corriger.
 * @return array|null Un tableau d'objets Evaluation ou null.
 */
1 usage  🧑  Matthieu Courréjou
public function findEvaluationsToGrade(User $user): array|null
{
    $now = new \DateTimeImmutable();

    return $this->createQueryBuilder( alias: 'e' )
        ->innerJoin( join: 'e.studentCopies', alias: 'sc' )
        ->andWhere('e.author = :user')
        ->setParameter( key: 'user', $user )
        ->andWhere('sc.commentary IS NULL')
        ->andWhere('e.endsAt <= :now')
        ->setParameter( key: 'now', $now )
        ->getQuery()
        ->getResult();
}

```

Le formateur voit alors toutes les évaluations à corriger, il faut maintenant lui afficher les copies à corriger correspondant à l'évaluation :

```
/**
 * Fonction qui cherche en base toutes les copies que le professeur doit corriger.
 *
 * @return float|int|mixed|string
 */
1 usage  🧑 Matthieu Courréjou
public function findStudentCopiesToGrade(Evaluation $evaluation)
{
    return $this->createQueryBuilder( alias: 'student_copy')
        ->andWhere('student_copy.evaluation = :eval')
        ->setParameter( key: 'eval', $evaluation)
        ->andWhere('student_copy.commentary IS NULL')
        ->andWhere('student_copy.score IS NULL')
        ->getQuery()
        ->getResult();
}
```

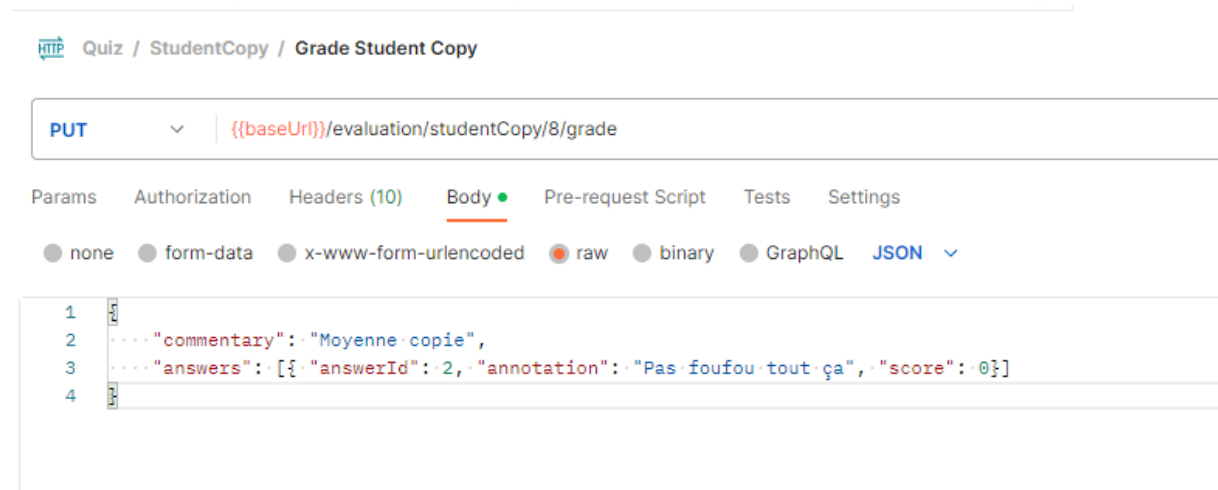
Et maintenant voici la route du contrôleur qui permet au formateur de soumettre sa correction :

```
/**
 * Corrige une copie d'élève.
 *
 * @param request $request L'objet Request contenant les données de la requête
 * @param user $user L'utilisateur connecté
 * @param StudentCopy $studentCopy la copie de l'élève à corriger
 * @param validatorInterface $validator L'instance du Validator
 * @param entityManagerInterface $entityManager L'instance de l'EntityManager
 * @param ApiRequestValidator $apiRequestValidator le service de validation des requêtes API
 *
 * @return Response la réponse indiquant que la copie a été corrigée avec succès
 */
🧑 Matthieu Courréjou +1
#[Route('/api/evaluation/studentCopy/{id}/grade', name: 'app_student_copy_grade', methods: ['PUT'])]
public function gradeStudentCopy(
    Request $request,
    #[CurrentUser] User $user,
    StudentCopy $studentCopy,
    ValidatorInterface $validator,
    EntityManagerInterface $entityManager,
    ApiRequestValidator $apiRequestValidator
): Response {
    // Assure que l'utilisateur est bien l'auteur de l'évaluation
    $this->denyAccessUnlessGranted( attribute: 'ROLE_FORMATEUR');
    $this->isAllowedOnResource($studentCopy->getEvaluation(), $user);

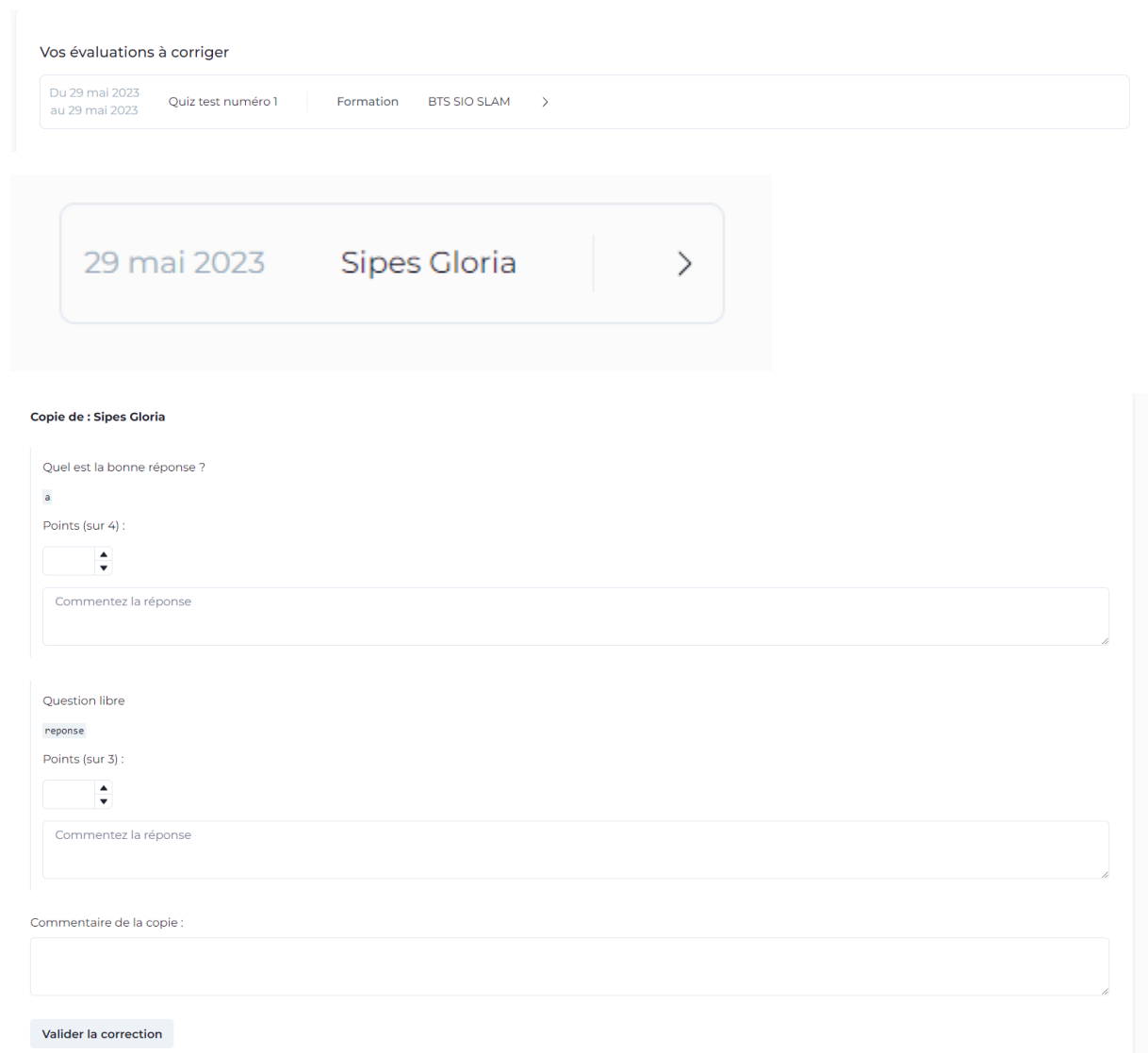
    // Validation du contenu de la requête
    $dto = $apiRequestValidator->checkRequestValidity(
        $request->getContent(),
        type: GradeCopyDTO::class
    );

    /**
     * Nous savons que le DTO possède bien un commentaire et un tableau,
     * maintenant il faut vérifier le contenu du tableau
     */
    /**
     * @var GradeCopyDTO $dto
     */
}
```

Voici le test Postman permettant de tester cette fonctionnalité :



Voici les interfaces de cette fonctionnalité :



g. Cas d'utilisation 7 : Consulter sa copie

Il faut ici afficher à l'élève toutes ses copies corrigées :

```
/**
 * Fonction qui récupères en base toutes les copies notées et corrigées de l'élève.
 *
 * @return float|int|mixed|string
 */
1 usage  🧑 Matthieu Courréjou
public function findGradedStudentCopies(User $user)
{
    return $this->createQueryBuilder( alias: 'student_copy')
        ->andWhere('student_copy.student = :user')
        ->setParameter( key: 'user', $user)
        ->andWhere('student_copy.commentary IS NOT NULL')
        ->andWhere('student_copy.score IS NOT NULL')
        ->getQuery()
        ->getResult();
}
```

Il faudra ensuite afficher la copie sélectionnée par l'élève :

```
/**
 * Fonction qui récupères en base une copie notée et corrigée de l'élève.
 *
 * @return float|int|mixed|string
 *
 * @throws NonUniqueResultException
 */
1 usage  🧑 Matthieu Courréjou *
public function findSingleGradedCopy(int $id)
{
    return $this->createQueryBuilder( alias: 'student_copy')
        ->andWhere('student_copy.id = :id')
        ->setParameter( key: 'id', $id)
        ->andWhere('student_copy.commentary IS NOT NULL')
        ->andWhere('student_copy.score IS NOT NULL')
        ->getQuery()
        ->getOneOrNullResult();
}
```


Voici à la route du contrôleur qui permet cette fonctionnalité :

```
/**
 * Récupère une copie notée et corrigée détaillée d'un élève.
 *
 * @param user $user L'utilisateur connecté
 * @param int $copyID L'ID de la copie
 * @param entityManagerInterface $entityManager L'instance de l'EntityManager
 *
 * @return JsonResponse la réponse JSON contenant la copie notée et corrigée détaillée de l'élève
 */
#[Route('/api/studentCopy/{copyID}/graded', name: 'app_studentcopy_getdetailedgradedcopy', methods: ['GET'])]
public function getDetailedGradedCopy([CurrentUser] User $user, int $copyID, EntityManagerInterface $entityManager): JsonResponse
{
    $copy = $entityManager->getRepository(StudentCopy::class)->findSingleGradedCopy($copyID);
    if (null === $copy) {
        $this->createNotFoundException();
    }
    if (!$this->isAllowedOnResource($copy, $user)) {
        $this->createAccessDeniedException();
    }

    if (null === $copy) {
        $this->json([], status: Response::HTTP_NOT_FOUND);
    }

    return $this->json($copy, context: ['groups' => 'studentCopy']);
}
```

Voici le test postman de cette fonctionnalité :

 Quiz / StudentCopy / Get Single Graded Copy

GET



{{baseUrl}}/studentCopy/8/graded

Voilà les vues de cette fonctionnalité :

Dernier quiz

Voir la copie →

Placement	Note	Moyenne de la classe	Evaluation
1 / 1	7 / 7	7/7	Quiz test numéro 1
			Formation BTS SIO SLAM

Vos copies corrigées

29 mai 2023	Quiz test numéro 1	Note: 7 / 7	Commentaire	Excellente copie	>
-------------	--------------------	-------------	-------------	------------------	---

Quiz : Quiz test numéro 1

Formation  
BTS SIO SLAM

Note : 7 / 7  
Appréciation:  
Excellente copie  
Position : 1 / 1  
Note moyenne : 7 / 7

Question :  
Quel est la bonne réponse ?

Réponse :  
  
**Note : 4 / 4**  
**Annotation :** Très bonne réponse.

Question :  
Question libre

Réponse :  
reponse  
**Note : 3 / 3**  
**Annotation :** Parfait, sujet maîtrisé.

#### h. Bilan de l'étape

Le cahier des charge est respecté et toutes les fonctionnalités sont présentes. Une des principale difficulté résidant dans le contrôle d'accès, et de vérifier si un utilisateur avait accès à la ressource demandée, et pour l'opération demandée.

### 5. Quatrième étape : Documentation, Déploiement et mise en production

#### a. Documentation

Plusieurs documentations sont nécessaires pour la bonne livraison du logiciel. La documentation technique crée plus tôt fut mise à jour tout au long du projet, elle est hébergée à l'adresse suivante :

<https://quiz-doc.netlify.app>

Il faut ensuite une documentation utilisateur. Je fais le choix d'en créer deux :

- Une au format vidéo (<https://matthieu-courrejou-portfolio-slam.netlify.app/projets/quiz>)
- Une au format texte (<https://github.com/MCkeydev/dubois-quiz-back>)

Vous pouvez aussi accéder à la collection de test Postman comme précisé plus haut :

<https://ticket-io.postman.co/workspace/Team-Workspace~b2a59d48-203a-4aa0-80eb-bc691672742a/collection/20766209-24cb22fa-17fa-4955-8bef-98c2ffcf547b?action=share&creator=20766209>

## b. Déploiement et mise en production

Maintenant que l'application est entièrement développée, il va être temps de la déployer. Pour ce faire il va nous falloir deux hébergeurs, un pour le front, ainsi qu'un pour le back.

Concernant la partie backend, je choisis de m'orienter vers l'hébergeur Heroku, car il est performant, et permet une intégration continue simple avec Github. De plus, il permet d'héberger une base de données relationnelle de manière assez facile.

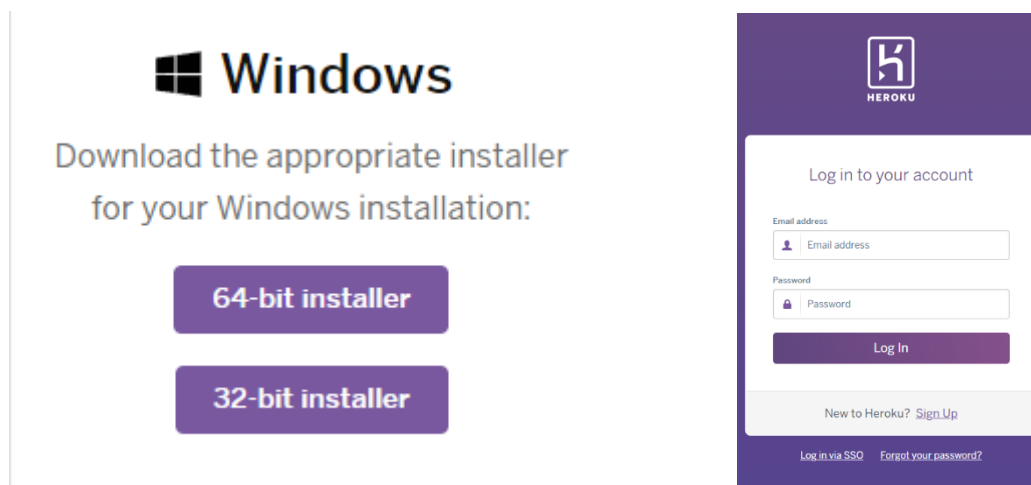
Pour la partie cliente, je m'oriente vers l'hébergeur netlify, qui permet d'héberger avec grande facilité des projets node, le tout gratuitement.

### i. Déploiement sur heroku (back) :

Avant toute chose, il nous faut installer le apache pack pour symfony, qui va se charger de générer un .htaccess cohérent :

```
$ composer require symfony/apache-pack
$ git add composer.json composer.lock symfony.lock public/.htaccess
$ git commit -m "apache-pack"
```

Nous commençons par installer l'invite de commande Heroku CLI, grâce à l'installateur fourni sur le site de l'hébergeur. Une fois installé, nous nous connectons.



Nous nous rendons ensuite dans le repository local de notre projet et créons le projet heroku :

```
PS E:\Linkweb\quiz> heroku create
```

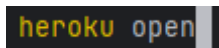
Il nous faut ensuite créer un fichier « Procfile » pour indiquer à heroku quel profil de server utiliser :

```
PS E:\Linkweb\quiz> echo 'web: heroku-php-apache2 public/' > Procfile
```

```
git add Procfile
```

```
git commit -m "Heroku Procfile"
```





L'application est maintenant déployée ! Cependant les variables d'environnement ne sont pas encore créées. Nous allons continuer sur l'interface graphique :

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some addons come with their own.

Hide Config Vars

APP_ENV	prod		
CORS_ALLOW_ORIGIN	^https?://zesty-alpaca-e3a0bf.netlify.app		
DATABASE_URL	mysql://wd1bin7wv1mxs5nz:g1340rsox8s25jrq		
JAWSDB_URL	mysql://wd1bin7wv1mxs5nz:g1340rsox8s25jrq		
JWT_PASSPHRASE	dc197904a344dbba4855ec89fb2ba557		
JWT_PUBLIC_KEY	%kernel.project_dir%/config/jwt/public.pe		
JWT_SECRET_KEY	%kernel.project_dir%/config/jwt/private.p		
KEY	VALUE		

L'application est presque prête, il faut générer les clés asymétrique permettant le chiffage des JWT, pour cela nous faisons dans la console du serveur la commande suivante :

```
$ php bin/console lexik:jwt:generate-keypair
```

Voilà ! Il ne reste dorénavant qu'à héberger la base de données, et importer le scrip sql permettant le bon fonctionnement de l'application. Heroku permet d'ajouter des « addons » au serveur, et parmi l'un d'eux il y a « JawsDB » qui permet d'héberger gratuitement une petite base de données MySQL. Nous l'installons alors :

Installed add-ons **\$0.00/month** [Configure Add-ons](#)

JawsDB MySQL Kitefin Shared  
dubois-quiz

Il nous faut alors créer la base de données :

```
heroku addons:create jawsdb:kitefin --name=dubois-quiz --version=5.7
```

Cette commande nous retourne la chaîne de connexion à la BDD, que nous allons renseigner en variable d'environnement serveur.

Il nous faut maintenant importer notre script sql en base :

```
PS E:\Linkweb\quiz> mysql -h=us-cdbr-east-06.cleardb.net -u=b1a12d5f4a845b -p=c5b23611 --reconnect=heroku_41022e08f29a94e < dump.sql
```

Voilà, nous pouvons voir les logs de l'application ici :

Personal > limitless-mountain-65739

GitHub MCKeydev/quiz main

Overview Resources Deploy Metrics Activity Access Settings

Activity Feed

mattcourrejou@gmail.com: Deployed 0ff8040b  
Today at 3:37 PM · v32 · [Compare diff](#)

Application Logs ALL PROCESSES

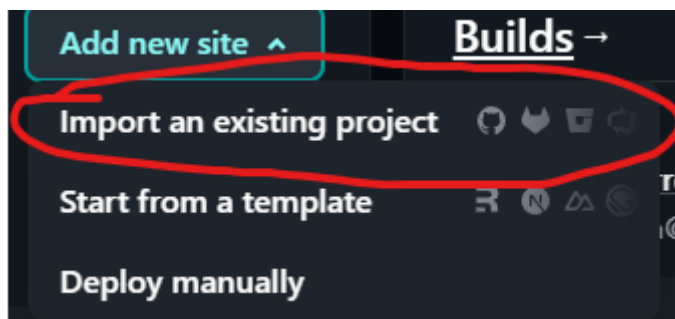
```
2023-05-29T13:47:03.438039+00:00 app[web.1]: 10.1.35.124 - - [29/May/2023:13:47:02 +0000] "POST /api/login HTTP/1.1" 200 114 "https://zesty-alpaca-e3a0bf.netlify.app/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36"
2023-05-29T13:47:03.438211+00:00 heroku[router]: at=info method=POST path="/api/login" host=limitless-mountain-65739.herokuapp.com request_id=5d098dd6-ee63-416b-906b-c1b2b4933072 fwd="31.35.99.45" dyno=web.1 connect=0ms service=595ms status=200 bytes=1146 protocol=https
2023-05-29T13:47:03.612983+00:00 app[web.1]: 10.1.52.186 - - [29/May/2023:13:47:03 +0000] "GET /api/home HTTP/1.1" 200 75 "https://zesty-alpaca-e3a0bf.netlify.app/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36"
2023-05-29T13:47:03.613388+00:00 heroku[router]: at=info method=GET path="/api/home" host=limitless-mountain-65739.herokuapp.com request_id=8aa85e7f-8102-4892-8804-bdad75407759 fwd="31.35.99.45" dyno=web.1 connect=0ms service=52ms status=200 bytes=418 protocol=https
2023-05-29T13:47:03.651041+00:00 app[web.1]: 10.1.35.124 - - [29/May/2023:13:47:03 +0000] "GET /api/studentCopy/preview/last HTTP/1.1" 200 1142 "https://zesty-alpaca-e3a0bf.netlify.app/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36"
2023-05-29T13:47:03.651376+00:00 heroku[router]: at=info method=GET path="/api/studentCopy/preview/last" host=limitless-mountain-65739.herokuapp.com request_id=4b33318e-94c0-4213-922e-d8a8edd4a394 fwd="31.35.99.45" dyno=web.1 connect=0ms service=91ms status=200 bytes=1486 protocol=https
2023-05-29T13:47:10.575307+00:00 app[web.1]: 10.1.35.124 - - [29/May/2023:13:47:10 +0000] "GET /api/studentCopies/graded HTTP/1.1" 200 608 "https://zesty-alpaca-e3a0bf.netlify.app/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36"
2023-05-29T13:47:10.575477+00:00 heroku[router]: at=info method=GET path="/api/studentCopies/graded" host=limitless-mountain-65739.herokuapp.com request_id=27e3444c-9632-4118-87ce-8a757049f2ef fwd="31.35.99.45" dyno=web.1 connect=0ms service=24ms status=200 bytes=952 protocol=https
2023-05-29T13:47:16.433279+00:00 app[web.1]: 10.1.35.124 - - [29/May/2023:13:47:16 +0000] "GET /api/studentCopy/25/graded HTTP/1.1" 200 606 "https://zesty-alpaca-e3a0bf.netlify.app/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36"
2023-05-29T13:47:16.433472+00:00 heroku[router]: at=info method=GET path="/api/studentCopy/25/graded" host=limitless-mountain-65739.herokuapp.com request_id=dafef01ad-7a53-47b5-89f8-034a96504e15 fwd="31.35.99.45" dyno=web.1 connect=0ms service=24ms status=200 bytes=950 protocol=https
```

☒ Autoscroll with output [Save](#)

Il ne reste plus qu'à tester que tout fonctionne, et c'est bon le côté backend de l'application est publié !

## ii. Déploiement sur Netlify (front) :

Les étapes sont beaucoup plus simples, il suffit de lier son compte github à son compte netlify, et effectuer les étapes suivantes :



## Import an existing project from a Git repository

From zero to hero, three easy steps to get your site on Netlify.

1. Connect to Git provider

2. Pick a repository

3. Site settings, and deploy!

### Connect to Git provider

Choose the Git provider where your site's source code is hosted. When you push to Git, we run your build tool of choice on our servers and deploy the result.

You can [unlock options for self-hosted GitHub/GitLab](#) by upgrading to the Business plan.


GitHub

GitLab

Bitbucket

Azure DevOps

Don't have a project yet? [Start from a template instead.](#)

 **MCkeydev** ▾

Search repos

MCkeydev/2nd-challenge

MCkeydev/Calculs

MCkeydev/cned\_portfolio Private

MCkeydev/Comment Private

MCkeydev/Comment-front Private

MCkeydev/Denombrements

MCkeydev/desktop-tutorial Private

MCkeydev/dubois-quiz-back

MCkeydev/dubois-quiz-front

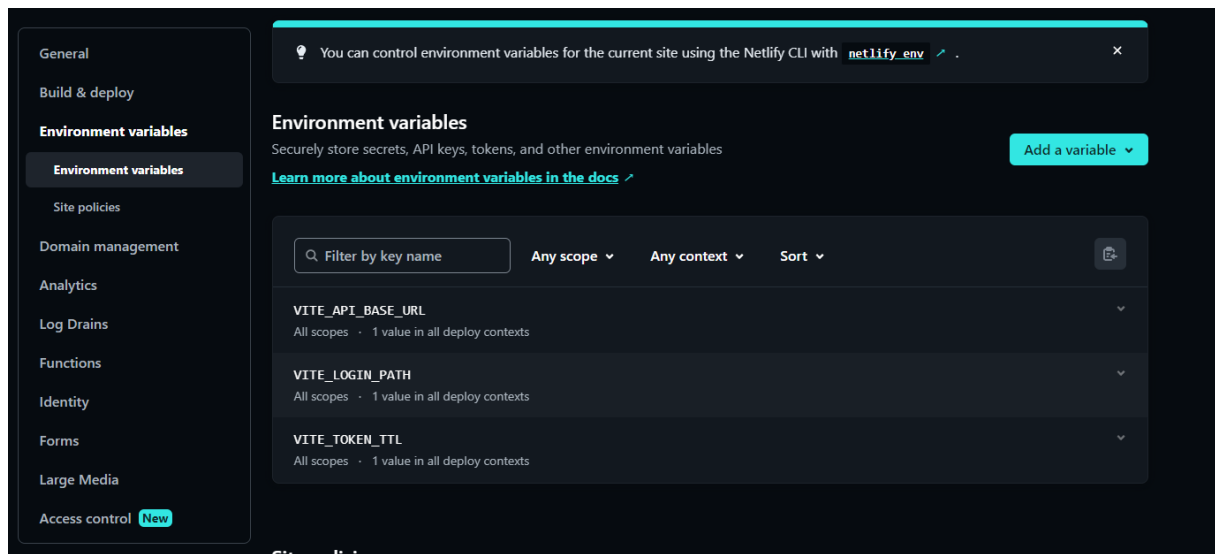
MCkeydev/E-Commerce-Project

MCkeydev/FrontEnd-Mentor

MCkeydev/FrontEndMentor-Github-API

*Il suffit de choisir le repository, et d'appuyer sur déployer.*

Il ne reste plus qu'à paramétrer les variables d'environnement du serveur :



**Bilan : Le front et le back sont déployés et testés, tout est fonctionnel. L'environnement est alors bien déployé en environnement de production et peut être utilisée par les utilisateurs.**

### c. Sauvegardes

Les bases de codes sont sauvegardées et versionnées chacune sur un repo git. Concernant la base de données, celle-ci est hébergée sur un service amazon AWS, et est sauvegardée tous les 24h dans la fenêtre suivante :

Server Status	
Property	Value
Status	AVAILABLE
Max Storage	0.005 GB
Backup Window	09:48-10:18 UTC
Maintenance Window	SAT:04:42-SAT:05:12 UTC

La rétention est d'une semaine, et l'on peut revenir à une sauvegarde préalable sur demande.

## 6. Conclusion du projet

En conclusion de ce projet, j'ai pu relever des défis significatifs tout en acquérant de précieuses connaissances. Mon objectif était de développer une application web pour le centre de formation "Formations Dubois", permettant la gestion d'évaluations en ligne par les professeurs et la participation des élèves.

La réalisation de cette mission m'a permis de mettre en pratique mes compétences en utilisant les technologies TypeScript, React, PHP et Symfony. Ces choix technologiques se sont avérés pertinents, offrant une base solide pour le développement et facilitant ma productivité tout au long du processus.

Un défi majeur a été de concevoir l'application de manière à permettre son évolutivité et sa scalabilité. La séparation du back-end et du front-end a été essentielle pour faciliter d'éventuels développements futurs, tels que l'expansion de l'application vers des plateformes mobiles ou de bureau. J'ai accordé une attention particulière à l'architecture de l'application afin de garantir sa flexibilité et sa capacité à s'adapter aux besoins à venir.

En conclusion, ce projet a été une expérience professionnelle enrichissante. J'ai pu relever des défis techniques, développer mes compétences et acquérir une meilleure compréhension des enjeux liés au développement d'applications web complexes. Je suis fier du travail accompli et reconnaissant des opportunités d'apprentissage que ce projet m'a offertes.