

Projets 1A 2021

Présentation &
Déroulement





Table des matières

Table des matières	2
Présentation	3
Déroulement	3
Différentes parties	3
Projets proposés	3
Organisation	4
Ressources de formation	5
Ressources spécifiques à la partie 1 :	5
Ressources spécifiques à la partie 2 :	8
Ressources spécifiques à la partie 3 :	10
Autres ressources :	10
Todo-list des différentes parties	11
Initialisation du projet	11
Gestion des utilisateurs	11
Mise en place des contrôleurs (front-office)	11
Administration du site (back-office)	12





Présentation

Les projets autonomes 1A ont pour but de se familiariser avec l'usage de Symfony 4 en développant une petite application en autonomie en se basant sur les tutoriels vidéos ainsi que sur le guide de chaque mission.

Déroulement

Plusieurs projets seront proposés et pourront être repris plus tard dans des applications réelles ou pourront devenir des applications autonomes selon la qualité de celles-ci. Les projets seront séparés en différentes parties qui seront toutes les mêmes entre les différents projets pour pouvoir proposer des jalons et revoir le code à intervalle régulier.

Chaque personne devra mettre son projet sur son GitHub perso pour qu'un membre du Glnfo puisse revoir le code et faire des commentaires sur celui-ci pour pouvoir s'améliorer.

Différentes parties

- 1) Initialisation du projet et création des entités & fixtures
- 2) Mise en place d'un système de gestion des utilisateurs
- 3) Création d'un controller pour afficher et gérer les données
- 4) Administration du site et gestion des formulaires

Projets proposés

- Appli pour suivre les anciens du Glnfo
- Application pour éditer des notes de frais
- Leaderboard style TRACS/NC
- Suivi de parc informatique
- Feedback de cours
- Application suivi formations Glnfo
- Gestion des emprunts
- BDD de stages effectués
- BDD de mobilités internationales
- Site de petites annonces

Les descriptions détaillées des projets sont disponibles, vous pouvez nous les demander.





Organisation

Chacun des participants proposera ses propres deadlines pour réaliser les différentes parties. De même, les participants mettront sur le google sheet les tâches effectuées en "En review" quand les commits auront été push sur GitHub pour qu'ils puissent être revus et commentés.

Les différentes tâches d'une partie devront être toutes (ou la majorité) passées au moins en "En review" avant la deadline fixée pour la partie.

Vous pouvez bien entendu demander de l'aide à votre tuteur quand vous le voulez mais une fois votre travail terminé, à la fin de chaque partie vous préviendrez votre tuteur afin qu'ils fassent une revue globale de votre site. Les tâches passeront alors en "Fait".



Ressources de formation

Ressources spécifiques à la partie 1 :

- Tout d'abord, commencez par [installer PHPStorm](#).
- Vérifiez que vous avez bien la version full de Laragon pour ceux qui sont sous Windows.
- Créez un dossier. Il contiendra votre projet. Pour Windows, il est bien de le créer dans "C:\Users\moi\PhpstormProjects".
- [Initialiser un projet Symfony 4](#) ← clique !
 - Installer bien Composer version **1.10.22** → télécharger le .exe, ça sera une version 2.*, et pour downgrade lancer la commande :

```
composer self-update 1.10.22
```

- Installer [Node.js](#)
- Ouvrez le terminal puis déplacez vous dans le dossier de votre projet et lancer la commande d'initialisation du projet Symfony **4.4** ⚠ :

```
composer create-project symfony/website-skeleton . "4.4.*"
```

- S'il vous demande des choses sur Docker, dites oui.
- Ouvrez ensuite PHPStorm, cliquez sur "Open" et sélectionnez le dossier de votre projet. Ouvrez le le fichier ".gitignore" et **ajoutez une ligne avec écrit ".idea"**. Et enfin lancez les deux commandes suivantes.

```
composer require symfony/webpack-encore-bundle  
npm install --save-dev @symfony/webpack-encore
```

- Mettre en place son projet sur GitHub
 - [Créez une clé SSH](#) si vous n'en avez pas déjà une puis [mettez la sur GitHub](#).
 - Une fois, sur GitHub, cliquez sur "+" et sur "New repository". Donnez lui un nom, par ex "PC <nom de mon projet>". Mettez une description si vous le souhaitez et choisissez si votre repo est public ou privé. Un repo privé sera accessible uniquement aux personnes autorisées alors qu'un repo public sera accessible par tout le monde. Ne rien cocher dans la partie "Initialize".
 - Une fois le repo créé, cliquez sur "Clone" et copiez le lien SSH.



- De nouveau sur PHPStorm, rendez-vous dans le terminal et taper les commandes suivantes :

Pour s'identifier la première fois :

```
git config --global user.name 'jeanloutre007'  
git config --global user.email 'jeanloutre@centrale-marseille.fr'
```

Et ensuite :

```
git init -b master  
git add .  
git commit -m "First commit"  
git remote add origin <Le lien que vous venez de copier>  
git remote -v #checker que le lien correspond  
git push origin master
```

Voilà votre projet initialisé ! 😊

Attention : Il ne faut pas mettre le dossier `.idea/` sur le dépôt, ce dossier est un dossier de fonctionnement de PHPStorm.

Si vous n'avez toujours pas fait de commit, vous pouvez mettre `.idea` dans le `gitignore` (git ignorera la liste des fichiers dans le fichier `gitignore`).

En cas de problème voir [cette technique](#).

- Mise en place et lancement du serveur

- `composer require server --dev` #installation

A chaque fois, pour lancer votre site web, vous allez démarrer Laragon ou MAMP puis exécuter la commande :

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

- [Créer ses entités](#)

Vous pouvez vous inspirer de [la vidéo de Lior Chamla](#) pour cette partie. (à partir de 31:08).

Cependant, il y a quelques points où vous ne devez pas faire comme Lior, les voici :

- ⚠ Nous n'utilisons pas [PhpMyAdmin](#) ⚠ Pour visualiser les bases de données, il faut se rendre dans Laragon et cliquer sur "Base de données" puis cliquer sur "Ouvrir".
- Dans le fichier `.env`, remplacer `DATABASE_URL` par :

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/db_name"
```





avec "db_name" le nom que vous voulez donner à votre base de données.

- Pour créer une base de données il faut utiliser la commande :

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

- A chaque fois que vous modifiez vos entités, pour que les changements soient pris en compte dans la DB, il faut faire une migration. On crée ainsi un fichier de migration et on l'exécute.

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

- [Gérer les champs uniques](#)
- [Comprendre les contrôleurs](#)

Vous pouvez vous aider de [la vidéo de Lior](#) pour comprendre et créer vos controllers. Comme précédemment, il y a quelques points différents :

- Vous n'allez pas installer un thème Bootstrap comme Lior mais vous allez [installer AdminBSB](#), qui est le thème que nous utilisons sur les sites du GInfo.
- Une fois Admin BSB installé, vous allez changer le fichier "base.html.twig" par [ça](#).
- Dans votre dossier "templates" vous créerez un fichier "bootstrap_3_layout.html.twig" et y copier [ça](#).

- [Créer sa première page](#) (ATTENTION À BIEN UTILISER LES ANNOTATIONS ROUTE)
- [Installation de webpack Encore](#)

- Création des fixtures

Vous pouvez vous inspirer de [la vidéo de Lior](#) pour créer vos fixtures. Créez un fichier de fixtures par entité. Ne faites pas tout dans un seul et même fichier, utilisez [cette partie](#) de la doc de Symfony !

Avant de mettre vos changements en review, assurez vous que votre code fonctionne !

Pour tester les migrations, fixtures et entités :

```
php bin/console doctrine:database:create  
php bin/console doctrine:migrations:migrate  
php bin/console doctrine:fixtures:load
```


Pour tester webpack encore et AdminBSB :

```
npm run watch
```

Ou juste faire une page d'accueil... :)



ATTENTION : (en cas de problème avec webpack)

La documentation a été mise à jour mais certains bouts de code traînent encore avec de vieilles versions de webpack. Si jamais vous avez une erreur avec un `.createSharedEntry()` dans le `webpack.config.js`, remplacez-le par `.addEntry()` 





Ressources spécifiques à la partie 2 :

Vidéo 4 de Lior Chamla +

- [La bible pour la gestion des utilisateurs](#) avec quelques points particuliers :
 - [Faire une hiérarchie des rôles](#)
 - [Donner le rôle minimal aux utilisateurs](#)
 - [L'utilisation du simple form_login](#) (ne pas oublier les msg d'erreur)
- [Sécuriser les pages](#) (attention, utiliser de préférence les annotations)
- [Ajouter la connexion via My](#)
- [Les messages flash pour faire de jolies notif](#)
- [Modifier les fixtures avec l'encoder](#)

Petit point sur le formulaire :

- Si vous avez un Token qui apparaît dans votre rendu de formulaire vous pouvez le retirer en suivant ces indications :
 - Créer un fichier nommé "messages.fr.yaml" dans le dossier translations
 - Y ajouter la ligne : Token: ""
 - Aller dans config/packages/translation.yaml et changer "en" en "fr" dans default_locale

Deux points sur la connexion via My :

- Le tuto porte sur une connexion avec l'utilisation du module FOSUserBundle. Vous n'en avez pas besoin, la seule ligne où le module est utilisé, c'est pour récupérer l'utilisateur. Vous pouvez le faire à l'aide du repository ! Vous n'avez donc pas besoin de l'UserManager.
EDIT : J'ai mis le code sans FOSUserBundle sur le wiki si vous avez la flemme de le modifier vous même.
- Les clés pour la connexion via My sont à nous demander directement.

Point de vigilance extrême : Faites attention à ce que vous mettez sur votre dépôt GitHub. Les clés OAuth de My ne doivent pas se retrouver sur un dépôt public.

Pour se prémunir de ça, ne mettez pas les clés dans le fichier .env, mais mettez les champs vides pour le dépôt et créez une copie de ce fichier qui sera appelé .env.local dans lequel vous mettrez les clés. Le .env.local est déjà ajouté dans le fichier .gitignore donc ignoré de git, et sera lu en priorité.





Ressources spécifiques à la partie 3 :

- [Utilisation des voters](#)
- [L'utilisation du repository](#) et le [QueryBuilder](#)
- [Documentation supplémentaire sur les QueryBuilders](#)
- [Mettre en place une pagination](#)

Autres ressources :

Les vidéos de l'excellent Lior Chamla :

- Vidéo 1 : <https://www.youtube.com/watch?v=UTusmVpwJXo>
- Vidéo 2 : https://www.youtube.com/watch?v=_cgZheTv-FQ
- Vidéo 3 : <https://www.youtube.com/watch?v=e5udJTjbYzw>
- Vidéo 4 : https://www.youtube.com/watch?v=_GjHWa9hQic

Ces vidéos sont essentielles pour comprendre la base de Symfony et ce que l'on fait.

Tutoriels made in GInfo :

- [Installer PHPStorm & Git](#)
- [Initialiser un projet Symfony 4](#)
 - Pour les projets claqués, on n'installera **pas FOSUserBundle**
 - Vérifiez bien que votre projet est en **Symfony 4** (en allant dans le composer.json et en regardant les lignes qui commencent par symfony/ : il faut que pour la plupart, ce soit écrit 4.4.* ou ^4.4)
- [CheatSheet Git & Symfony \(DWEB6\)](#)

Pages de la documentation utiles :

- [Le module sécurité, ou gérer ses utilisateurs](#)
- [L'upload de fichiers](#)



Todo-list des différentes parties

Le sheet de suivi prévaut à ce qu'il y a ci-dessous.

Initialisation du projet

- ☐ Initialisation du projet avec Symfony 4
- ☐ Mise en place du projet sur GitHub
- ☐ Création d'un fichier README.md pour présenter le projet
- ☐ Création des entités avec la commande make:entity
- ☐ Création des migrations associées aux entités
- ☐ Relations entre les entités
- ☐ Création d'un jeu de fausses données (Fixtures)
- ☐ Installation de webpack Encore
- ☐ Installation de AdminBSB
- ☐ Créer un base.html.twig pour son application (squelette)
- ☐ Création d'un contrôleur pour la page d'accueil et de la vue associée

Gestion des utilisateurs

- ☐ Création de l'entité utilisateur
- ☐ Création de la fonction d'inscription (Avec Form & Security)
- ☐ Création d'un formulaire de login
- ☐ Création d'une fonction de logout
- ☐ Ajout des éventuelles relations entre l'user et les autres entités
- ☐ Création des différents rôles
- ☐ Système de connexion via MyCA

Mise en place des contrôleurs (front-office)

- ☐ Création d'un contrôleur relatif à l'entité principale de l'application
- ☐ Création d'une action pour lister ces entités
- ☐ Création d'une action pour consulter une entité en particulier
- ☐ Création d'un form pour créer une entité
- ☐ Création d'une action pour créer une entité
- ☐ Sécurisation des différentes pages à l'aide de voters



Administration du site (back-office)

- ☐ Panneau d'administration avec chiffres clés
- ☐ Ajouter/Retirer des administrateurs du site
- ☐ Création d'un rôle intermédiaire pour gérer des parties du site
- ☐ Création d'une tâche automatisée pour récup des données ou notifier

