Ce dossier projet a été rédigé à l'issue d'un stage professionnel effectué du 7 juin au 19 juillet 2021 au sein de la société CODE COLLIDERS au Mans

DOSSIER DE PROJET

ASSEMBLÉE PARLEMENTAIRE DE LA FRANCOPHONIE

DÉVELOPPEUR WEB ET WEB MOBILE

EMMANUEL CLEMENT

TABLE DES MATIÈRES

1. PRÉSENTATION

Après l'école secondaires et muni d'un Certificat d'Etudes Secondaires Supérieures en Sciences industrielles électromécaniques (C.E.S.S . - ISCED niveau 3), je me suis intéressé de près à la communication, et plus spécialement à la technologie de celle-ci.

J'ai acquis un certificat de technique en audiovisuel, où j'ai exercé pour une TV locale, puis un autre en infographie 2D/PAO. Le monde d'internet ouvrant grand ses portes, je me suis pris dans le développement Web en fullstack PHP de façon autodidacte. J'ai mis toutes mes compétences en œuvre pour différentes entreprises, pour ouvrir ma propre agence de communication, que j'ai tenu pendant 5 ans dans le Luxembourg belge (Arlon).

Peu après mon emménagement au Mans, j'ai eu un trou dans mon parcours, et ai dû laisser mes compétences de développeur/infographiste plus ou moins de côté pendant 4 ans, pour passer un certificat de création de contenus digitaux en développement 3D sur systèmes immersifs et mobiles à Laval (Mayenne). J'y rejoins l'équipe de Eon Reality pour arrêter presque 2 ans plus tard suite à la liquidation judiciaire de l'entreprise. Pendant 4 ans je continue l'infographie print/web et le développement web pour des associations, mais je sens que la technologie évolue plus vite que moi.

Mes méthodes de travail sont de plus en plus désuètes et je manque cruellement de veille technologique, même si mes bases sont bonnes.
Ma curiosité sur de nouvelle choses me démange et c'est ainsi que depuis le 12 avril 2021 et jusqu'au 22 octobre 2021, je suis les cours de Via Formation en vue du titre de Développeur Web et Web Mobile.

Il me faut une remise à niveau 2021, assimiler les méthodes modernes, améliorer mes connaissances, et apprendre les frameworks actuels utilisés en entreprise.

Pour valider le titre, j'ai proposé mes services à Code Colliders en tant que stagiaire, et c'est ici que je vais vous décrire ma réalisation professionnelle.

2. LE STAGE

Pour valider le titre de Développeur Web et Web Mobile, j'ai travaillé en tant que stagiaire du 7 juin au 19 juillet 2021 chez Code Colliders, entreprise qui me correspondait par ses activités en développement web Full Stack dans lesquelles il m'est possible d'y mettre en pratique les compétences acquises. Lors de mon entretien, il m'a été précisé qu'un projet était à réaliser avec Symfony, framework dont je n'avais aucune connaissances, ce qui m'a motivé en terme de challenge.

2.1 PRÉSENTATION DE L'ENTREPRISE

Créée en 2017, l'entreprise Code Colliders est spécialisée dans le développement de sites internet, et a pour vocation d'apporter à ses clients une expertise dans le domaine du numérique. A ce titre, son personnel réalise également des missions de conseil, d'audit, de formation, de gestion de projet, de référencement, de sécurité informatique et d'hébergement / infogérance.

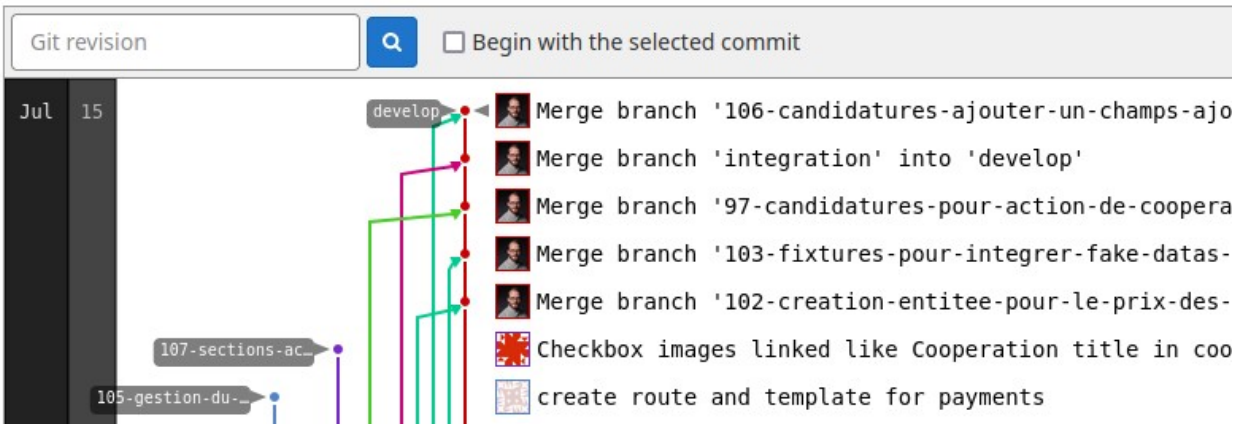
En interne, l'équipe est composée de

- Romain Virmaux : **Développeur** - Gérant, tuteur de mon stage
- Sandra Grémy : **Assistante de direction**
- William Lefebvre : **Développeur** - Associé
- Christine Gunsenheimer : **Développeuse** indépendante
- Emilie Viel : **Développeuse** en alternance
- David Remond : **Stagiaire** comme moi dans la même session de formation chez Via Formation

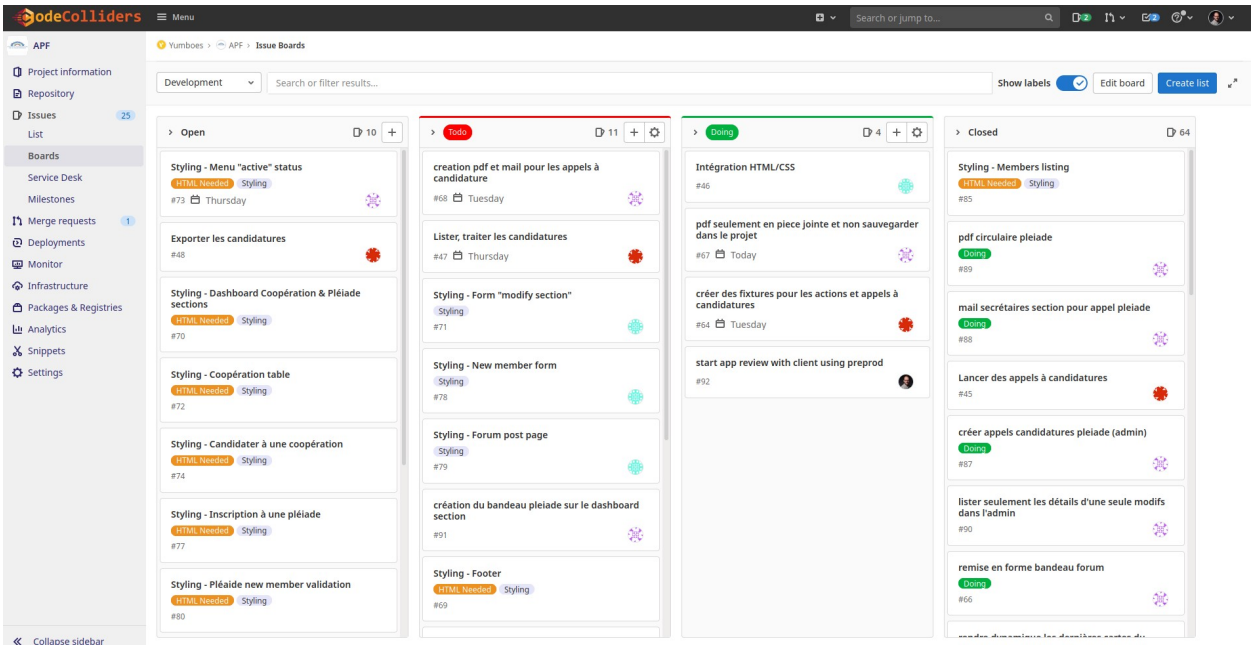
2.2 ENVIRONNEMENT DE TRAVAIL ET LOGICIELS

Pour rester cohérent avec la mise en situation, je continue à utiliser l'ordinateur portable qui m'a été confié par l'établissement de formation, configuré avec Linux Lubuntu 20.04. D'ailleurs, tous les développeurs de Code Colliders utilisent Linux.
Le tuteur de mon stage m'a remis le cahier des charges du projet, j'ai pu donc y jeter un oeil.
Comme les autres employés, j'utilise l'IDE PhpStorm pour développer le PHP, JavaScript et CSS éventuel

L'ensemble du code source produit par l'entreprise est versionné sur une plateforme GitLab hébergée par l'entreprise. Mon tuteur de stage m'en a donné l'accès et grâce à ma clé SSH, je suis préparé à faire mes commits en me servant de PhpStorm, avec un joli rond rouge en guise d'avatar.



Ainsi, les processus de développement élaborés en interne sont inspirés des métodes Agiles et utilisent des outils de planification de Gitlab, de la même manière qu'un Kanban



2.3 LE PROJET

L'Assemblée parlementaire de la Francophonie, association qui regroupe 88 sections constituées de parlements ou d'organisations interparlementaires membres, a besoin d'un outil pour gérer sa base de données et automatiser de nombreuses tâches, réalisées aujourd'hui de façon disparate.

Elle possède déjà un site internet (http://apf.francophonie.org), qui comprend un espace membre (permettant de mettre les documents pour les réunions de façon sécurisé) et une plateforme de consultation de ses membres (sécurisée également). Elle a développé en 2018 une plateforme d'inscription à ses événements, indépendante de son site internet.

L’objectif est de développer un nouvel outil de gestion de base de données, indépendant de ces autres plateformes, avec un lien sur son site internet pour que les sections puissent y accéder directement.

Ce projet est dirigé par Clément Meunier de Yumboes, agence de Digital Marketing, qui nous a fourni la maquette disponible à cette adresse : https://xd.adobe.com/view/0e2527c3-8be3-4180-abd2-c608489bb52f-5b76/grid

Code Colliders gère la réalisation de l'application, avec Maxime Senzamici, indépendant qui gère la partie CSS.

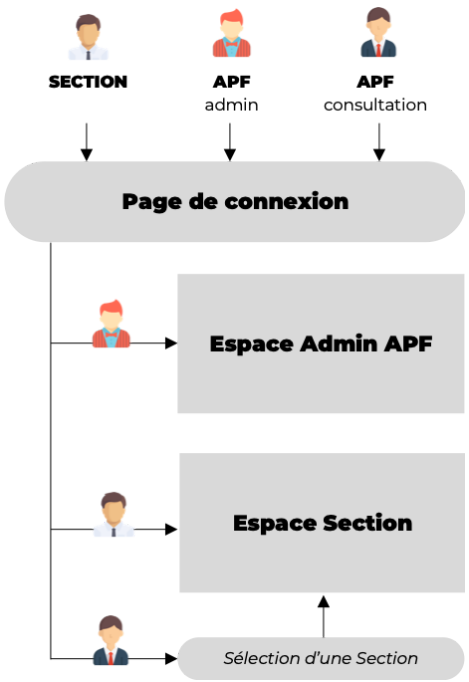
Nous sommes deux stagiaires à développer le projet en entier, avec l'aide de Emilie sur certains points.

2.4 TECHNOLOGIE DE RÉALISATION

Symfony a été choisi comme technologie de développement par le client Yumboes

2.5 STRUCTURE DU SITE

La plateforme sera divisée en 2 espaces distincts ayant chacun leurs fonctionnalités spécifiques : Espace dédié aux Sections et Espace dédié à l’Admin (back office)
3 types de comptes permettront une connexion sécurisée à ces espaces sécurisés



- Sections - Accès au Tableau de Bord :

- Consultation et modifications des coordonnées de la Section et de son Bureau.
- Paiement des cotisations et historique de paiement.
- Consultation des programmes de coopération et candidatures aux programmes ouverts.
- Soumettre des candidats à la Pléiade et paiement des médailles en ligne.
- Accès au forum.

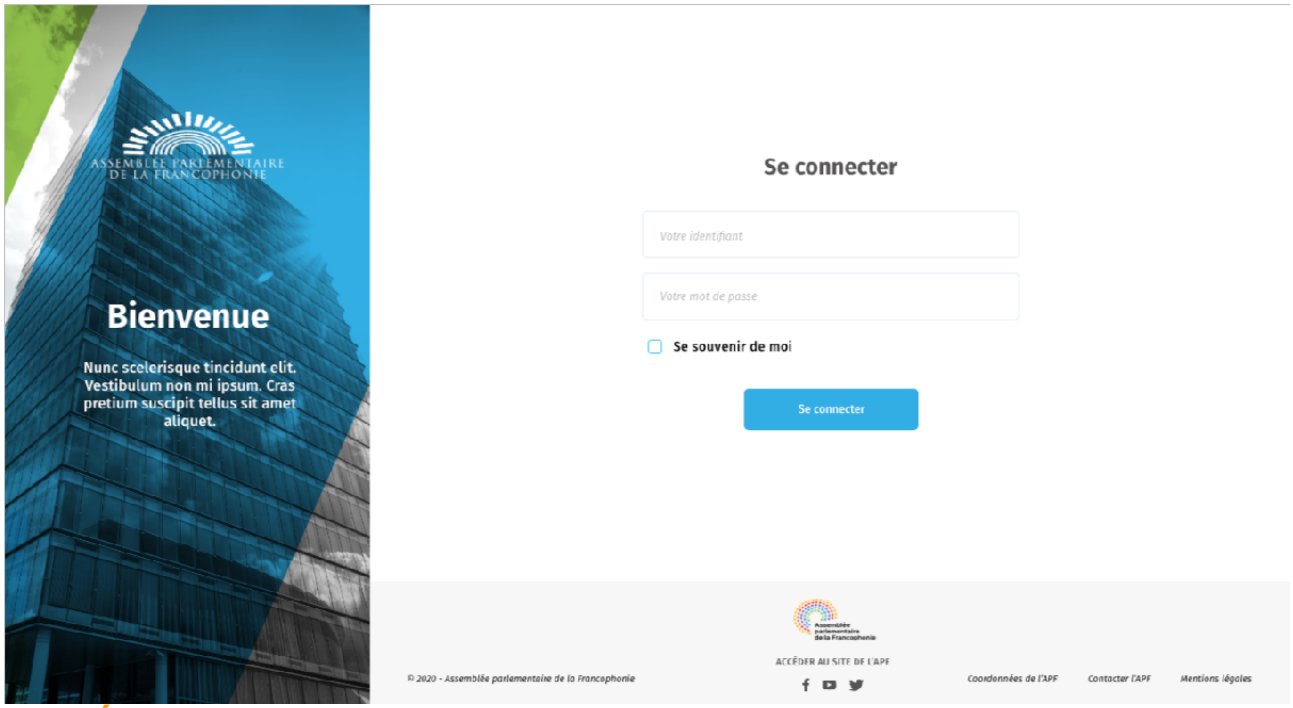
- Administrateur - Accès au Back Office :

- Consulter et accepter les demandes de modifications des coordonnées des Sections.
- Créer/modifier/supprimer d'autres comptes « Admin » ayant les mêmes droits et plusieurs comptes Sections
- Créer des cotisations pour chaque Section.
- Sélectionner une Section pour accéder à l'ensemble des données de cette Section (coordonnées, bureau, coopération, cotisations, pléiade).
- Créer et gérer des programmes de coopération et valider les candidatures des Sections aux programmes ouverts.
- Émettre des appels pour la Pléiade, valider les demandes des Sections, envoyer les factures.
- Gérer le Forum de discussion.

- Consultant - Accès à la lecture du tableau de bord des sections :

- Compte Utilisateur permettant d'accéder à l'ensemble des pages des Sections sans possibilité de modification.

L'entrée sur la plateforme s'effectuera via une page de connexion commune aux 3 types de comptes.



LE DÉBUT

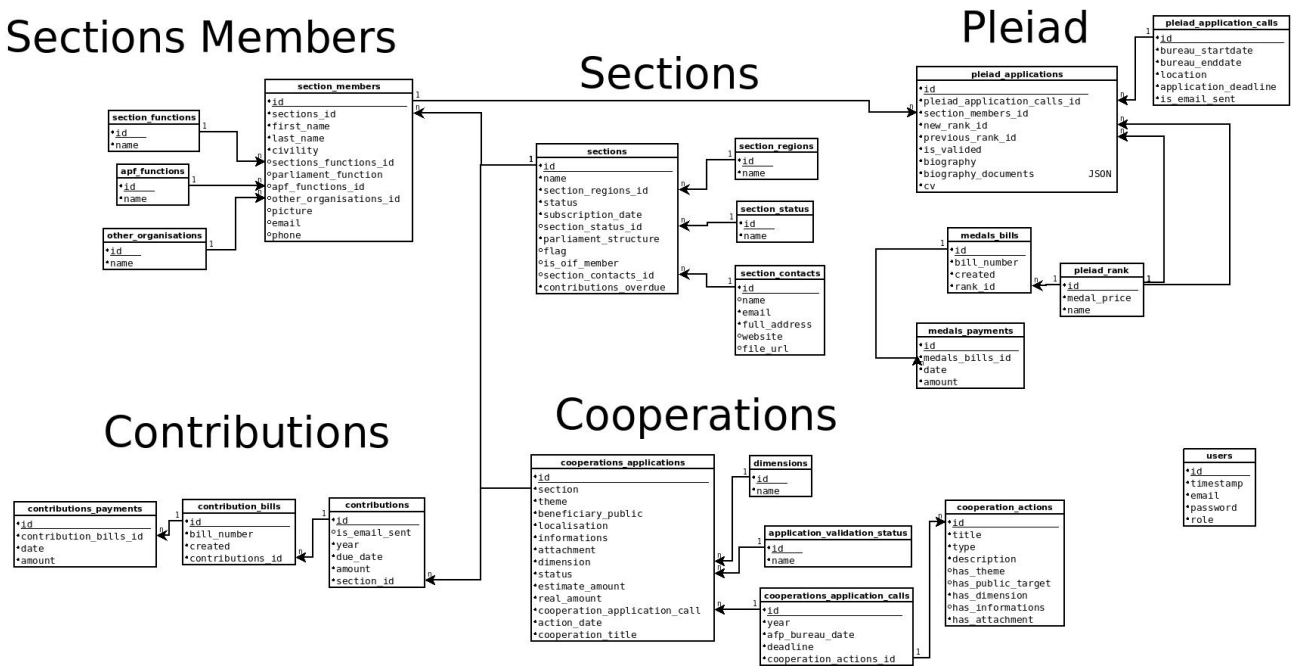
Le premier jour du stage fût consacré à l'installation de Symfony sur ma machine. Il fallait y ajouter les bundles :

- **Webpack Encore**, pour compacter (minify & uglify) et interpréter le JavaScript avec tests try/catch et CSS proprement, en compilant le SCSS. On le lance avec Yarn (npm).

- **EasyAdmin**, ensemble d'outils permettant de créer l'interface d'administration, en SCRUD (tableau avec fonctions Search, List, Show, Edit et New). sur les entités Doctrine qui gèrent la base de donnée.

- **Vich Uploader**, qui permet de faciliter les téléchargements de fichiers qui sont attachés aux entités ORM de Doctrine en tant qu'instances des classes "File" de Symfony. Ce bundle crée des modèles pour générer des URL publiques vers les fichiers.

Ensuite il a fallu faire une lecture approfondie du projet et la représentation d'une base de données permettant d'y voir plus clair dans le cahier des charges.



2.7 APPRENTISSAGE

Sur base du cahier des charges et du schéma de la base de donnée, je me suis entraîné pendant deux semaines sur Symfony en créant des entités, puis en les intégrant dans EasyAdmin pour pouvoir sortir un premier jet de l'interface d'administration. Une façon de me familiariser avec le système et l'architecture de Symfony :

- Les classes de Symfony
- Le développement PHP orienté objet
- Les différents bundles cités
- L'architecture Modèle-Vue-Contrôleur de Symfony
- Les entités
- Les dépôts de données (Repositories)
- Les templates en TWIG

Évidemment, il m'était impossible d'assimiler tout ça de façon naturelle, j'ai donc consulté la documentation officielle en ligne sur le site https://symfony.com/ en anglais. Quelques aides m'ont été consultables sur le site forum anglophone de Stack Overflow https://stackoverflow.com/

2.8 DÉVELOPPEMENT MORCEAU PAR MORCEAU

2.8.1 L'INTERFACE ADMINISTRATEUR

En gros, l'administrateur a accès à tout. Il aura donc sa propre route ("/admin") qui le mènera vers l'interface d'administration de EasyAdmin, contrôleur créé en ligne de commande.
Ce même contrôleur « Dashboard » est composé d'un menu qui mène vers tous les contrôleurs dont il a besoin, qui gèrent les entités attribuées.

```php
public function index(): Response
{
    return parent::index();
}

public function configureDashboard(): Dashboard
{
    return Dashboard::new()
        ->setTitle('<img src="/assets/img/logo-admin.png" width="65%">')
        ;
}

public function configureMenuItems(): iterable
{
    yield MenuItem::linktoDashboard('Accueil', 'fa fa-home');
    yield MenuItem::linkToCrud('Utilisateurs', 'fa fa-users-cog', User::class);
    yield MenuItem::linkToCrud('Sections', 'fa fa-globe', Section::class);
    yield MenuItem::linkToCrud('Logs', 'fa fa-clipboard-list', LoginLogs::class);
    yield MenuItem::linkToRoute('Modifs', 'fas fa-history', 'audit_history', [
        'entity' => 'App-Entity-SectionMembers',
    ]);
    yield MenuItem::section("Coopérations");
    yield MenuItem::linkToCrud('Actions', 'fas fa-handshake', CooperationActions::class);
    yield MenuItem::linkToCrud('Appels', 'fas fa-bullhorn', CooperationApplicationCalls::class);
    yield MenuItem::linkToCrud('Candidatures', 'far fa-file-alt', CooperationApplication::class);
    yield MenuItem::section("Pléiade");
    yield MenuItem::linkToCrud('Appels', 'fas fa-bullhorn', PleiadApplicationCalls::class);
    yield MenuItem::linkToCrud('Candidatures', 'far fa-file-alt', PleiadApplication::class);
    yield MenuItem::linkToCrud('Demandes Paiements', 'fa fa-shopping-bag', MedalsContribution::class);
    yield MenuItem::section();
    yield MenuItem::linkToCrud('Cotisations', 'fa fa-shopping-bag', Contribution::class);
    yield MenuItem::linkToCrud('Règlements', 'fa fa-shopping-cart', ContributionPayment::class);
    yield MenuItem::linkToRoute('Forum', 'fa fa-comment', 'forum_index');
}
```

Par exemple, une entité « Section » a été créée en ligne de commande, à laquelle un contrôleur CRUD a été rattaché.
Dans ce contrôleur, on configure la classe en ajoutant dans ce cas-ci une action « Voir » qui sera visible dans le tableau CRUD du template TWIG de EasyAdmin.
Le lien « Voir » créé comme action va permettre de sortir de l'interface Admin pour atterrir dans le tableau de bord consacrée aux sections, qui sera développé ultérieurement.

```php
public function configureFields(string $pageName): iterable
{
    return [
        TextField::new('name', 'Nom')
        ->setHelp('ex: france'),
        ChoiceField::new('region', 'Région')
            ->autocomplete()
            ->setChoices([  'Afrique' => 'Afrique',
                    'Amérique' => 'Amérique',
                    'Asie-Pacifique' => 'Asie-pacifique',
                    'Europe' => 'Europe']
            ),
        ChoiceField::new('status', 'Statut')
            ->autocomplete()
            ->setChoices([  'Membre' => 'Membre',
                    'Associé' => 'Associé',
                    'Observateur' => 'Observateur']
            ),
        TextField::new('membershipDate', "Date d'adhésion")
        ->setHelp('ex: 1967 (présent 1ere AG)'),
        ChoiceField::new('active', 'Actualité du statut')
            ->autocomplete()
            ->setChoices([  'Suspendu' => 'Suspendu',
                    'Sous observation' => 'Sous observation',
                    'Autre' => 'Autre']
            ),
        TextField::new('parliamentStructure', 'Structure du parlement')
        ->setHelp('ex: Monocaméral'),
        BooleanField::new('isOifMember', 'Membre oif')->hideOnIndex(),
        VichImageField::new('imageFile')->hideOnForm(),
        VichImageField::new('imageFile')->onlyOnForms(),
    ];
}
```

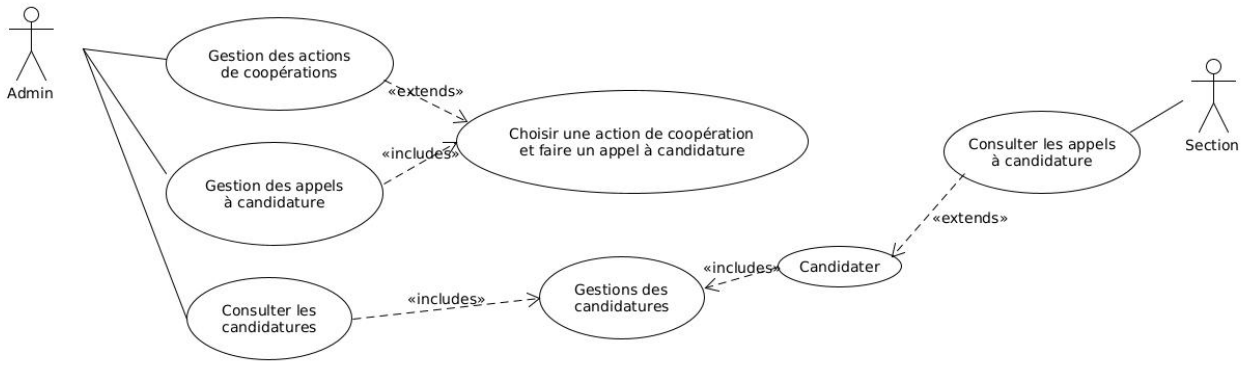
```php
public function configureActions(Actions $actions): Actions
{
    $linkSection = Action::NEW('Voir', 'Voir')
    ->linkToCrudAction('idSection');

    return $actions
        ->add(Crud::PAGE_INDEX, $linkSection)
    ;
}
```

chaque utilisateur fait partie d'une section.

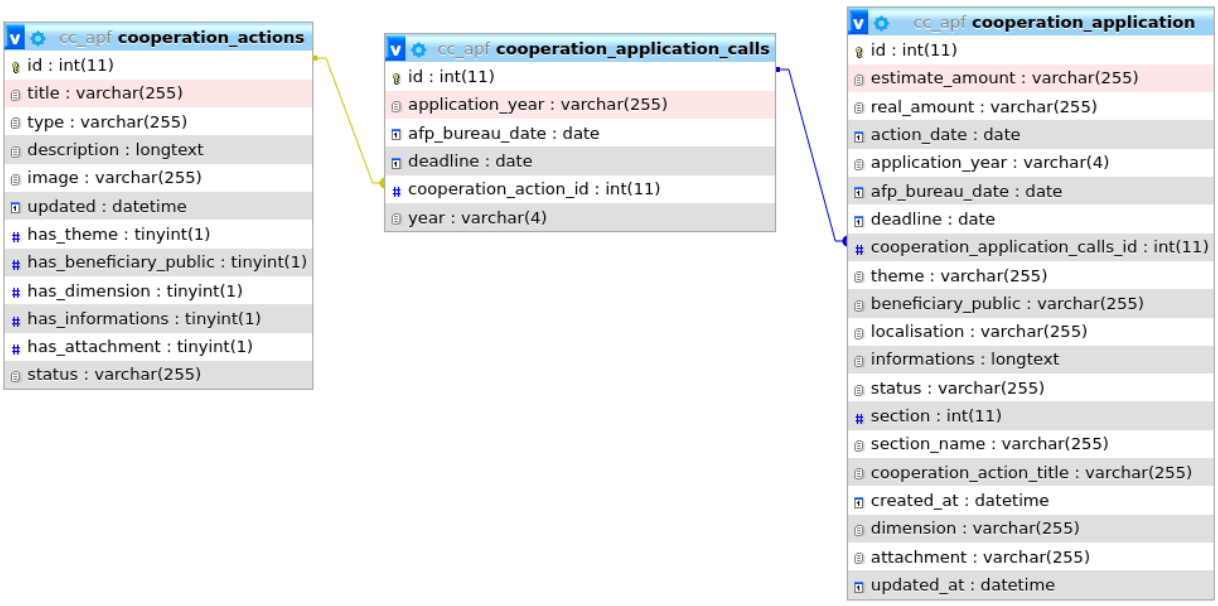
2.8.2 LA BASE DE DONNÉES

Je me suis attaqué à toute la partie de la gérance des coopérations. Pour y voir plus clair, il me fallait faire un UML d'activités :

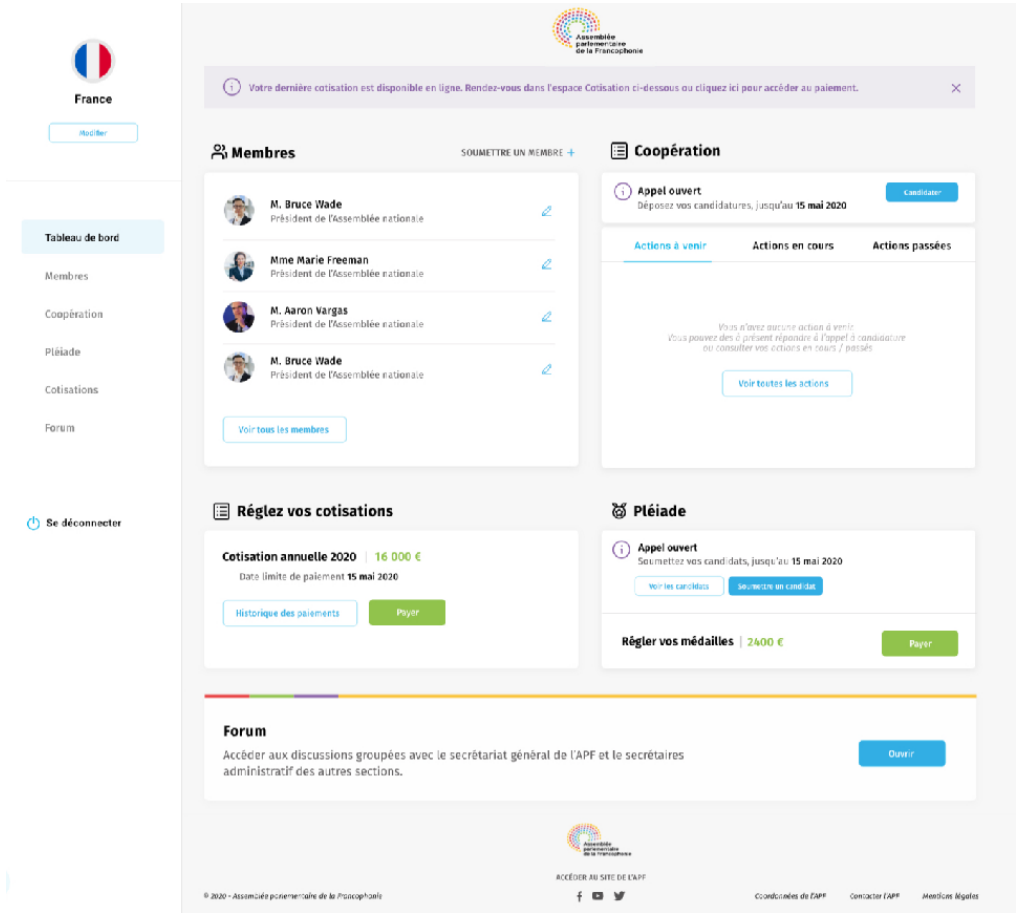


L'administrateur décrit une action de coopérations et/ou fait un appel à candidature d'une action de coopération créée. Une section peut consulter son tableau de bord et y voir le ou les appels à candidature pour y postuler si intéressé.
L'administrateur peut voir dans son interface les candidatures en cours, les valider ou les rejeter.

La difficulté n'était pas matérielle, ni provenant d'un manque de connaissance mais était plutôt un problème de logique. En effet, il aurait été inopportun de créer une table « retour de candidature ». Ce jeu de va-et-vient pouvait juste se concrétiser par la mise en place de champs non requis dans la table, qui viendraient se mettre à jour suivant les réponses des utilisateurs.



2.8.3 LE TABLEAU DE BORD



Ce tableau de bord est consacré aux sections. Il est donc évident d'attribuer une route « /section » pour ce tableau. Sachant que la partie « Consultation » ne pourra qu'accéder aux informations des sections, il suffira selon le rôle (section ou consultation) de gérer l'affichage du tableau avec des conditions.

Dans un premier temps, il a fallu constituer un CSS de base qui ressemblerait plus ou moins à la maquette donnée, pour pouvoir d'un côté avoir une vision fluide des résultats et de l'autre côté pouvoir montrer au client le travail en cours.
Mettre en place une simple feuille de style contenant des classes nommées dans le même esprit que Bootstrap était chose aisée en attendant le travail de Maxime, chargé de toute la partie CSS.
Le Webpack Encore permet de créer des feuilles SCSS qui sont importées dans une seule feuille.

Pour faire le tableau de bord, on n'utilise plus le bundle EasyAdmin et on reste dans les seules couches d'abstraction de Symfony.

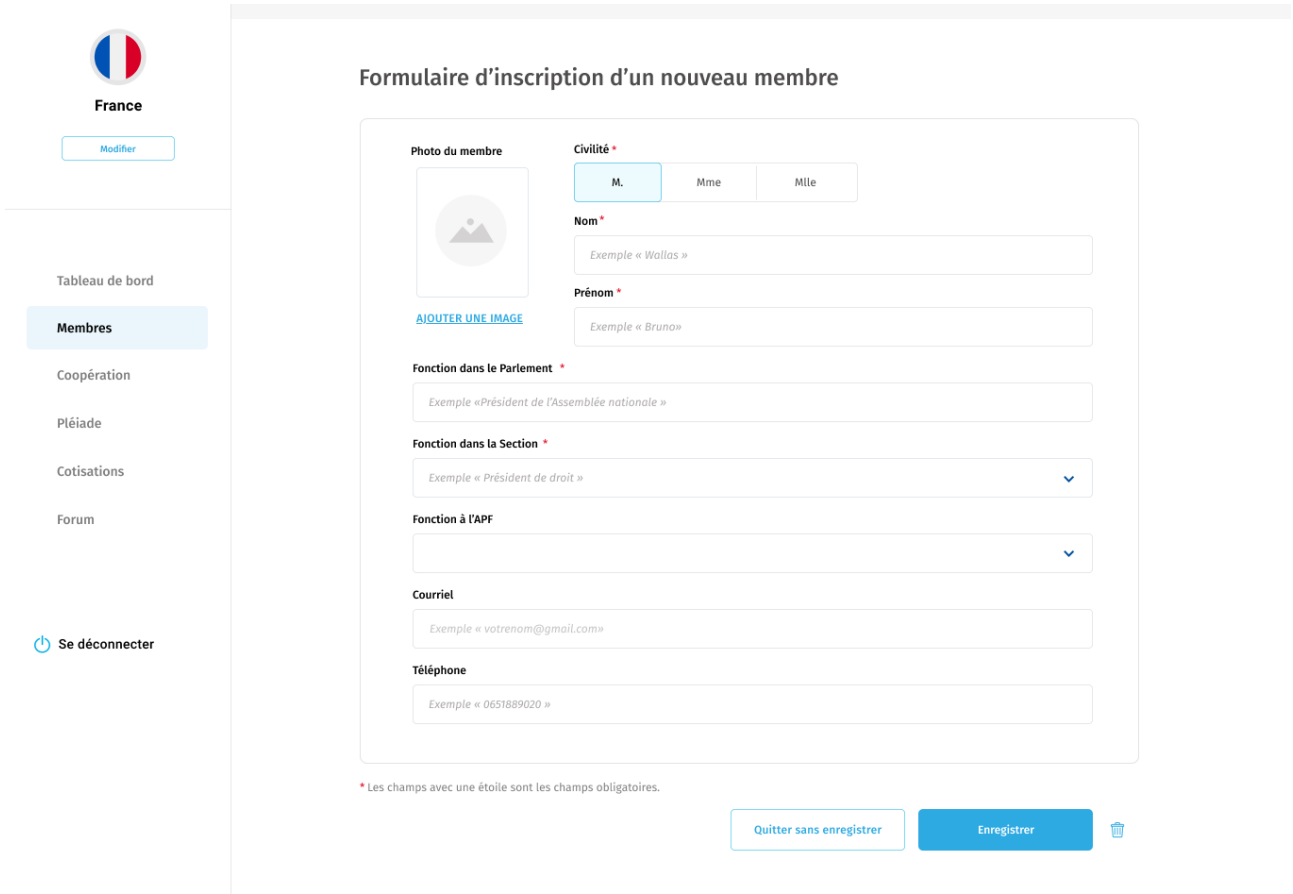
Le tableau de bord est consultable par un administrateur (dans ce cas il aura un lien de retour vers son interface), par les sections qui utiliseront automatiquement leur identifiant pour

afficher les données leur correspondant, et par un consultant, qui n'aura que des accès en lecture.
Ces accès sont gérés par les conditions dans les templates sur le rôle de l'utilisateur, car maintenant nous pouvons agir directement sur les templates, sans devoir faire une surcharge comme précédemment avec EasyAdmin.

Les contrôleurs et templates associés sont fait en ligne de commande. Dans ces contrôleurs, on peut configurer les classes et accéder aux entités afin de retourner les variables utilisées par les templates correspondant.

Grâce au makeCRUD de Symfony, on obtient directement des contrôleurs pour les formulaires avec des routes spécifiques pour les actions du CRUD, ainsi que des templates en TWIG pour les pages de formulaires. Ne reste plus qu'à les configurer.

Pour me familiariser avec cette méthode, je constitue les formulaires pour l'inscription de nouveaux membres d'une section.



```php
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('imageFile', VichImageType::class, [
            'required' => false,
        ])
        ->add('civility', ChoiceType::class, [
            'choices'  => [
                'M.' => "M.",
                'Mme' => "Mme",
                'Mlle' => "Mlle",
            ],
            'multiple' => false,
            'label' => 'Civilité',
            'expanded' => true,
        ])
        ->add('firstName', TextType::class, [
            'label' => 'Nom',
            'attr' => ['placeholder' => 'Exemple « Wallas »']
        ])
        ->add('lastName', TextType::class, [
            'label' => 'Prénom',
            'attr' => ['placeholder' => 'Exemple « Bruno »']
        ])
        ->add('sectionFunctions', ChoiceType::class, [
            'choices'=>[
                'Président de Parlement' => 'Président de Parlement',
                'Parlementaire (Député(e))'=> 'Parlementaire (Député(e))',
                'Parlementaire (Sénateur(trice))' => 'Parlementaire (Sénateur(trice))',
                'Président de section' =>'Président de section',
                'Fonctionnaire' => 'Fonctionnaire',
            ],
            'label' => 'Fonction dans la Section',
            'attr' => ['placeholder' => 'Exemple « Président de droit »'],
        ])
        ->add('parliamentFunctions', TextType::class, [
            'label' => 'Fonctions dans le Parlement',
            'attr' => ['placeholder' => 'Exemple « Président de l\'Assemblée nationale »'],
        ])
        ->add('apfFunctions', ChoiceType::class, [
```

Dans la partie « Coopérations » qui m'a été attribuée, Je dois récupérer la date d'échéance de toutes les candidatures, pour en sortir la date la plus éloignées et l'afficher dans l'entête de la carte, avec un bouton « candidater ». Si toutes les dates sont périmées, alors un bandeau signifiant qu'aucune candidature n'est disponible se met en place.

Cela vaut pour la première page du tableau de bord. Le même travail sera réalisé pour la page « Coopération »

Dans le cahier des charges, l'entête est un menu de trois liens constitués comme suit :

- Actions À VENIR = toutes les actions dont le statut est « Validé par le Bureau ».

- Actions EN COURS = toutes les actions dont la date de l'action à été renseignée par l'APF dans l'admin.

- Actions PASSÉES = toutes les actions dont le montant réel à été renseignée par l'APF dans l'admin.

J'ai dû faire un JavaScript pour l'affichage du contenu de chaque lien dans la carte « Coopérations , stocké dans le dossier « assets » qui est géré par le Webpack Encore.

```
// SHOW COOPERATION CARD IN SECTION DASHBOARD

const cooperation = {
    navlinks: document.querySelectorAll("#cooperation-nav a"),
    items: document.getElementById("cooperation-items").children,
    show: (element) => element.style.display = "block",
    showItem: (event) => {
        const target = event.target.nodeName === "NOBR" ? event.target.parentNode : event.target;
        const selectedIndex = Array.from(cooperation.navlinks).indexOf(target);
        for (let i = 0; i < cooperation.navlinks.length; i++) {
            cooperation.items[i].style.display = "none";
            cooperation.navlinks[i].classList.remove("active");
        }
        cooperation.navlinks[selectedIndex].classList.add("active");
        cooperation.show(cooperation.items[selectedIndex]);
        event.preventDefault();
    }
};
for (let navlinks of cooperation.navlinks)
    navlinks.addEventListener("click", cooperation.showItem, false);

// WHILE WAITING CSS HIDDEN "#COOPERATION-ITEMS" DIV

for (let item of cooperation.items){
    if(item.id!=="cta-div"){
        item.style.display = "none";
    }
}

////////

cooperation.show(cooperation.items[0]);

//////////////////////
```

La dernière partie de ce script « While waiting CSS » est destinée à être effacée après que Maxime, qui s'occupe du CSS du site, ait bien stipulé dans la feuille de style que l'identifiant « #cooperation-items » de la balise ne doit pas être affiché (display : hidden).

Comme je ne savais pas si le site était enclin à évoluer, j'ai conçu ce script afin que peu importe si on enlève un lien ou si on en ajoute un, il fonctionnera toujours.

Les requêtes SQL sont fait avec la couche d'abstraction de Symfony. Certaines sont aisées, d'autres requêtes plus compliquées doivent être appelées avec le « createQueryBuilder »

```
// GET COOPERATION CARD

$maxResultsDisplayed = 3;

$applicationsApprovedByApf = $cooperationApplicationRepository->createQueryBuilder('a')
    ->setParameter('apf','apf')
    ->setParameter('sectionId',$section->getId())
    ->where('a.action_date IS NULL')
    ->andWhere('a.real_amount IS NULL')
    ->andWhere('a.status = :apf')
    ->andWhere('a.section = :sectionId')
    ->orderBy("a.createdAt","DESC")
    ->setMaxResults($maxResultsDisplayed)
    ->getQuery()
    ->getResult()
;

$applicationsWithActionDate = $cooperationApplicationRepository->createQueryBuilder("a")
    ->setParameter('sectionId',$section->getId())
    ->where('a.action_date IS NOT NULL')
    ->andWhere('a.real_amount IS NULL')
    ->andWhere('a.section = :sectionId')
    ->getQuery()
    ->getResult()
;

$applicationsWithRealAmount = $cooperationApplicationRepository->createQueryBuilder("a")
    ->setParameter('sectionId',$section->getId())
    ->where('a.real_amount IS NOT NULL')
    ->andWhere('a.section = :sectionId')
    ->getQuery()
    ->getResult()
;

$applicationsForThisSection = $cooperationApplicationRepository->findBy(["section" => $section]);
$applicationCalls = $cooperationApplicationCallsRepository->findByNewApplicationCalls($applicationsForThisSection);
$applicationsDeadline = "";
$needBanner = false;
```

Reste à savoir si on doit afficher le bandeau « pas de candidatures », et ensuite retourner une

```php
foreach ($applicationCalls as $applicationCall) {
    if (!$applicationCall["isApplied"]) {
        $applicationsDeadline = $applicationCall["frenchDeadline"];
        break;
    }
}

foreach ($applicationCalls as $applicationCall) {
    if (!$applicationCall["isApproved"]) {
        $needBanner = true;
        break;
    }
}

        //VALUES TO RETURN TO TWIG TEMPLATE

$cooperations = [
    'applications' => [
        'upcomingActions' => $applicationsApprovedByApf,
        'currentActions' => $applicationsWithActionDate,
        'pastActions' => $applicationsWithRealAmount
    ],
    'applicationCalls' => $applicationCalls,
    'applicationsDeadline' => $applicationsDeadline,
    'needBanner' => $needBanner,
];
```

valeur sous forme de tableau pour le template.

Lors d'un clic sur ce bouton, la section pourra voir la liste des candidatures en cours. La section peut cocher la candidature pour postuler.

2.9 JEU D'ESSAI

Les fixtures (jeu de données) sont accessibles dans Symfony avec l'ORM Doctrine. On peut les installer dans le projet via Composer.

Dans le fichier « bundles » du dossier « config », on configure de la manière suivante :

```php
<?php

return [
    Symfony\Bundle\FrameworkBundle\FrameworkBundle::class => ['all' => true],
    Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle::class => ['all' => true],
    Symfony\Bundle\TwigBundle\TwigBundle::class => ['all' => true],
    Symfony\Bundle\WebProfilerBundle\WebProfilerBundle::class => ['dev' => true, 'test' => true],
    Symfony\Bundle\MonologBundle\MonologBundle::class => ['all' => true],
    Symfony\Bundle\DebugBundle\DebugBundle::class => ['dev' => true],
    Symfony\Bundle\MakerBundle\MakerBundle::class => ['dev' => true],
    Doctrine\Bundle\DoctrineBundle\DoctrineBundle::class => ['all' => true],
    Doctrine\Bundle\MigrationsBundle\DoctrineMigrationsBundle::class => ['all' => true],
    Symfony\Bundle\SecurityBundle\SecurityBundle::class => ['all' => true],
    Twig\Extra\TwigExtraBundle\TwigExtraBundle::class => ['all' => true],
    Symfony\WebpackEncoreBundle\WebpackEncoreBundle::class => ['all' => true],
    EasyCorp\Bundle\EasyAdminBundle\EasyAdminBundle::class => ['all' => true],
    Doctrine\Bundle\FixturesBundle\DoctrineFixturesBundle::class => ['dev' => true, 'test' => true],
    Vich\UploaderBundle\VichUploaderBundle::class => ['all' => true],
    Knp\Bundle\SnappyBundle\KnpSnappyBundle::class => ['all' => true],
    DH\AuditorBundle\DHAuditorBundle::class => ['all' => true],
];
```

Ne reste plus qu'à créer un fichier de fixtures, dans une classe qui fait l'extension de la classe « Fixture », et y inscrire les commandes. Le fichier est à inclure dans le dossier « DataFixtures »

```php
//Add Cooperation Action and its application call

$cooperationAction = new CooperationActions();
$cooperationAction->setTitle("Le Programme de coopération Noria");
$cooperationAction->setType("Promotion de la Francophonie");
$cooperationAction->setDescription("Créé en juillet 2002, lors de la réunion du Bureau de l'APF à Be
$cooperationAction->setHasAttachment(false);
$cooperationAction->setHasBeneficiaryPublic(false);
$cooperationAction->setHasDimension(false);
$cooperationAction->setHasInformations(false);
$cooperationAction->setHasTheme(false);
$manager->persist($cooperationAction);

    $cooperationApplicationCall = new CooperationApplicationCalls();
    $cooperationApplicationCall->setAfpBureauDate(new \DateTime("2020-09-08T10:00:00"));
    $cooperationApplicationCall->setApplicationYear("2021");
    $cooperationApplicationCall->setCooperationAction($cooperationAction);
    $cooperationApplicationCall->setDeadline(new \DateTime("2021-11-08T10:00:00"));
    $cooperationApplicationCall->setYear("2021");
    $manager->persist($cooperationApplicationCall);

$cooperationAction = new CooperationActions();
$cooperationAction->setTitle("Programme de développement parlementaire");
$cooperationAction->setType("Développement");
$cooperationAction->setDescription("Le programme multilatéral de développement parlementaire est un
$cooperationAction->setHasAttachment(false);
$cooperationAction->setHasBeneficiaryPublic(false);
$cooperationAction->setHasDimension(false);
$cooperationAction->setHasInformations(false);
$cooperationAction->setHasTheme(false);
$manager->persist($cooperationAction);

    $cooperationApplicationCall = new CooperationApplicationCalls();
    $cooperationApplicationCall->setAfpBureauDate(new \DateTime("2020-09-08T10:00:00"));
    $cooperationApplicationCall->setApplicationYear("2021");
    $cooperationApplicationCall->setCooperationAction($cooperationAction);
    $cooperationApplicationCall->setDeadline(new \DateTime("2021-10-08T10:00:00"));
    $cooperationApplicationCall->setYear("2021");
    $manager->persist($cooperationApplicationCall);
```

La touche finale est le chargement des fixtures en ligne de commande.

2.10 SÉCURITÉ

LES ROUTES

Dans ce projet, nous avons deux routes : "admin" et "sections".
Celles-ci sont protégées par un firewall, un système d'authentification d'utilisateur de Symfony qui permet de donner l'accès qu'aux utilisateurs authentifiés par leur rôle défini dans la base de données, et qui reconnaît les moyens de connexion utilisés (login form, API token, etc). L'utilisateur est authentifié dès son arrivée sur le site. En cas de premier accès, l'utilisateur sera identifié comme anonyme.
Cela signifie que toute requête peut avoir un jeton anonyme pour accéder à une ressource, tandis que certaines actions (c'est-à-dire certaines pages ou certains boutons) peuvent encore nécessiter des privilèges spécifiques.

LA CONNEXION

Par défaut, les tentatives de connexion sont limitées à 5 demandes échouées pour l'adresse IP + nom d'utilisateur et 25 demandes échouées pour l'adresse IP. La deuxième limite permet d'éviter qu'un attaquant utilisant plusieurs noms d'utilisateur ne contourne la première limite, sans perturber les utilisateurs normaux des grands réseaux (comme les bureaux avec plusieurs utilisateurs simultanés).

VULNÉRABILITÉ XSS

Les attaques de scripts sont déjouées par des fonctions sanitaires qui empêchent les balises HTML d'être interprétées.

INJECTION SQL

Les injections SQL sont filtrées grâce aux requêtes faites en PDO dans la couce d'abstraction de Symfony, qui contiennent la méthode « prepare(sql) » et « execute(parameters) ». Les paramètres sont ainsi filtrés.

```
$applicationsApprovedByApf = $cooperationApplicationRepository->createQueryBuilder('a')
    ->setParameter('apf','apf')
    ->setParameter('sectionId',$section->getId())
    ->where('a.action_date IS NULL')
    ->andWhere('a.real_amount IS NULL')
    ->andWhere('a.status = :apf')
    ->andWhere('a.section = :sectionId')
    ->orderBy("a.createdAt","DESC")
    ->getQuery()
    ->getResult();

;
```

Cet exemple est la couche d'abstraction de cette méthode PDO, qui filtre les injections SQL :

```
$sql = "SELECT *
FROM cooperation_application AS a
WHERE ISNULL(a.action_date) = 1
AND ISNULL(a.real_amount) = 1
AND a.status =:apf
AND a.section = :sectionId
ORDER BY a.createdAt DESC";

$applicationsApprovedByApf = $this

        ->connection→prepare($sql)

        ->execute([
                'apf' => apf,
                'sectionId' => $section->getId(),
        ])

        ->fetchAll()
```

VULNÉRABILITÉ CRSF

Sur chaque formulaire, un jeton est inséré dans un champ caché. Celui-ci est créé aléatoirement au chargement de la page, est stocké dans les paramètres de session PHP et est envoyé avec la reqûete. La page qui reçoit la requête analyse le jeton et le compare avec celui de la session.

LE MOT DE PASSE

Les mots de passe sont automatiquement encodé par une couche d'abstraction nommée "passwordHasher", qui encode en hachant avec du sel le mot de passe originel.

3. COMPÉTENCES COUVERTES PAR LE STAGE

Vue synoptique de l'emploi-type

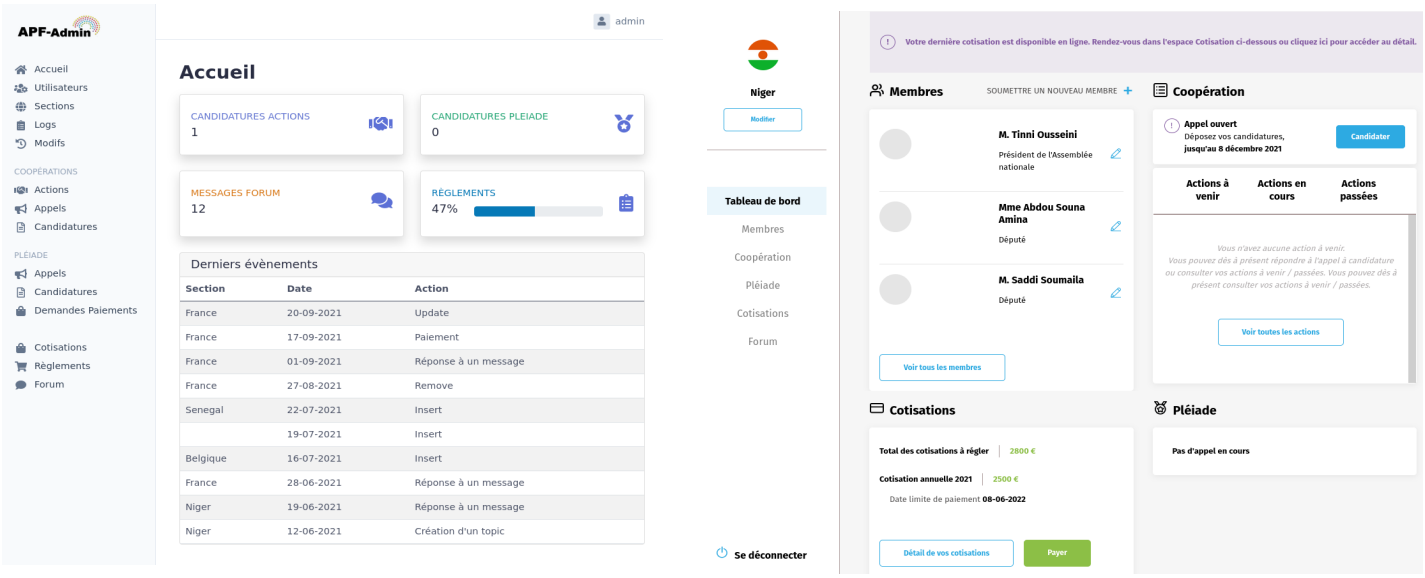
N° Fiche AT	Activités types	N° Fiche CP	Compétences professionnelles
1	Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité	1	Maquetter une application
		2	Réaliser une interface utilisateur web statique et adaptable
		3	Développer une interface utilisateur web dynamique
		4	Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce
2	Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité	5	Créer une base de données
		6	Développer les composants d'accès aux données
		7	Développer la partie back-end d'une application web ou web mobile
		8	Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

3.1 MAQUETTER UNE APPLICATION (CP1)

Bien que le wireframe soit donné par le client, j'ai dû faire des diagrammes afin de facilité le process de développement (CFR UML d'activités)

3.2 RÉALISER UNE INTERFACE UTILISATEUR (CP3-CP4)

Le framework Symfony utilisé dans ce projet est une solution de gestion de contenu qui a permis de réaliser deux interfaces web dynamiques.
Ces interfaces sont sécurisées par les points cités dans la partie « Sécurité ».



3.3 CRÉER UNE BASE DE DONNÉES (CP5)

J’ai créé une base de données avec phpMyAdmin, dont j’ai noté les paramètres dans le fichier « .env » du dossier projet de Symfony.

DATABASE_URL="mysql://USER:PASSWORD@DB_HOST/DB_NAME?DB_VERSION
"

La couche d'abstraction ORM Doctrine a permis de créer des tables dans la base de donnée sous forme d'objets PHP directement liées à la base réelle MariaDB créée en même temps que cette classe par ligne de commande.

```php
/**
 * @ORM\Entity(repositoryClass=CooperationActionsRepository::clas
 */
class CooperationActions
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $status;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $title;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $type;

    /**
     * @ORM\Column(type="text")
     */
    private $description;
```

3.4 L'ACCÈS AUX DONNÉES (CP6)

Le dossier "Entity" présent dans le MVC de Symfony stocke les classes crées par Doctrine lors de la création des classes cités au point précédent. Ces classes comprennent des fonction d'appel qui se codent simplement ainsi :

$title = getTitle ;

```php
public function getId(): ?int
{
    return $this->id;
}

public function getStatus(): ?string
{
    return $this->status;
}

public function setStatus(string $status): self
{
    $this->status = $status;

    return $this;
}

public function getTitle(): ?string
{
    return $this->title;
}

public function setTitle(string $title): self
{
    $this->title = $title;

    return $this;
}

public function getType(): ?string
{
    return $this->type;
}
```

Pour des accès aux données plus complexes, un dossier « Repository » présent dans le MVC stocke les dépôts sous forme de fonctions de classes. Ainsi, on peut ajouter des actions.

Par exemple dans le projet, pour créer un nouveau mot de passe :

```php
public function upgradePassword(PasswordAuthenticatedUserInterface $user, string $newEncodedPassword): void
{
    if (!$user instanceof User) {
        throw new UnsupportedUserException(sprintf('Instances of "%s" are not supported.', \get_class($user)));
    }

    $user->setPassword($newEncodedPassword);
    $this->_em->persist($user);
    $this->_em->flush();
}
```

3.5 DÉVELOPPER LA PARTIE BACK-END + COMPOSANTS (CP7-CP8)

Tout le projet est du développement back-end dans Symfony qui est une application gestion de contenu fournissant des couches d'abstractions de PHP. Les composantes ont été ajoutées soit manuellement, soit directement en ligne de commande.

4. REMERCIEMENTS

Je tiens à remercier l'équipe de Code Colliders, de m'avoir fait confiance, pour leur accueil, leur bonne humeur et pour cette excellente ambiance de travail.

Particulièrement, je remercie :

- **Romain Virmaux** pour sa bienveillance, ses explications claires lors de mes gros blocages, et ses conseils même après le stage.

- **William Lefebvre** d'avoir pris le temps de me donner des coups de pouce et conseils alors qu'il était concentré dans son travail.

- **Christine Gunsenheimer** pour ses cours à Via Formation et le suivi de mes exercices, qui a remonté mon niveau de connaissances en SQL et liens des tables de bases de données.

- **Sandra Grémy** qui m'a fait rire au téléphone lors de l'entretien, ce qui m'a permis de ne pas trop angoisser.

- **David Remond**, mon collègue stagiaire, avec qui j'ai beaucoup échangé afin de régler les problèmes que nous rencontrions, et souvent les miens.