

Stage du 11 janvier au 04 mars 2022

DOSSIER PROJET

Réalisation d'une application de gestion de recrutements

Titre professionnel Développeur Web / Web Mobile

Yamina **JOUILLE**

Sommaire

SOMMAIRE.....	2
1. COMPETENCES DU REFERENTIEL COUVERTES PAR LE PROJET	5
A. DEVELOPPER LA PARTIE FRONT-END D'UNE APPLICATION WEB OU WEB MOBILE EN INTEGRANT LES RECOMMANDATIONS DE SECURITE	5
<i>Maquetter une application</i>	<i>5</i>
<i>Réaliser une interface utilisateur web statique et adaptable</i>	<i>5</i>
<i>Développer une interface utilisateur web dynamique</i>	<i>5</i>
B. DEVELOPPER LA PARTIE BACK-END D'UNE APPLICATION WEB OU WEB MOBILE EN INTEGRANT LES RECOMMANDATIONS DE SECURITE .	5
<i>Créer une base de données</i>	<i>5</i>
<i>Développer les composants d'accès aux données.....</i>	<i>5</i>
<i>Développer la partie back-end d'une application web ou web mobile</i>	<i>5</i>
2. RESUME DU PROJET	6
3. CAHIER DES CHARGES, EXPRESSION DES BESOINS OU SPECIFICATIONS FONCTIONNELLES DU PROJET	7
4. SPECIFICATIONS TECHNIQUES DU PROJET	9
A. OUTILS DE COMMUNICATION	9
B. OUTILS DE VERSIONING	9
C. ENVIRONNEMENT DE DEVELOPPEMENT	9
<i>Ordinateur.....</i>	<i>9</i>
<i>Editeur de code</i>	<i>9</i>
<i>Navigateur web.....</i>	<i>10</i>
<i>Symfony 5.3.....</i>	<i>10</i>
<i>Serveur de développement.....</i>	<i>11</i>
<i>Webpack encore.....</i>	<i>12</i>
<i>Bootstrap 5.....</i>	<i>12</i>
5. REALISATIONS COMPORTANT LES EXTRAITS DE CODE LES PLUS SIGNIFICATIFS.....	13
A. MODELE DE DONNEES	13
B. LES ENTITES	13
C. MANIPULATION DES DONNEES	16
D. GESTION DU FRONT, LES VUES	18
E. ÉDITION A LA VOLEE	21
F. FLATPICKER	23
6. PRESENTATION DU JEU D'ESSAI DE LA FONCTIONNALITE LA PLUS REPRESENTATIVE	24
<i>Création et gestion des candidats.....</i>	<i>24</i>
<i>Création et déroulement de la session de recrutement</i>	<i>27</i>
7. DESCRIPTION DE LA VEILLE.....	31
8. DESCRIPTION D'UNE SITUATION AYANT NECESSITE UNE RECHERCHE ANGLOPHONE.....	32



Textes en anglais.....	32
Textes en français	32
9. CONCLUSION.....	33
10. ANNEXES.....	34
A. MAQUETTES.....	34
B. VUE DE L'APPLICATION SUR TABLETTE ET MOBILE (MODE PAYSAGE)	38
C. ARBORESCENCE DU PROJET	39

Remerciements

Je tiens à remercier toutes les personnes et tous les organismes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce dossier de projet.

Merci à mon conjoint pour son aide et son soutien sans faille au quotidien. J'ai pu ainsi m'investir complètement dans ma formation.

Merci à mon formateur Monsieur Alain Merucci pour son aide, sa disponibilité et sa patience.

Merci à la Région Bourgogne Franche-Comté, Onlineformapro et l'Access Code School de Lons-Le-Saunier qui ont rendu cette formation possible.

Merci aux trois co-fondateurs de l'entreprise Débutant(e) accepté(e), mon tuteur Monsieur Michel Chevassu, Monsieur Christophe Boutet et Monsieur Thomas Boyer pour leurs conseils tant sur la gestion de projet, la méthodologie et les aspects plus techniques.

Merci à toute l'équipe de Débutant(e) accepté(e) pour leur accueil et leur disponibilité en cas de difficulté, les échanges avec eux étaient très enrichissants.

Enfin, merci à tous les apprenants de la promotion Lons 2022 de l'ACS pour leur aide et leur bonne humeur.



1. Compétences du référentiel couvertes par le projet

A. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Maquetter une application

Pendant mon stage, j'ai réalisé une application de gestion des candidatures. En me basant sur l'existant et après plusieurs échanges avec mon tuteur, j'ai réalisé des maquettes sur Whimsical (un logiciel d'espace de travail visuel collaboratif utilisé dans l'entreprise).

Réaliser une interface utilisateur web statique et adaptable

J'ai intégré le front-end de l'application (espace administration) avec Bootstrap 5 et le moteur de template Twig, les outils qui ont été utilisés pour faire le site vitrine de l'entreprise et qui sont majoritairement utilisés dans l'entreprise.

Développer une interface utilisateur web dynamique

Les interfaces sont générées à la demande et le contenu varie en fonction des actions des utilisateurs (clic sur les items de menus, boutons, remplissage de formulaire, etc.).

L'ensemble des pages est disponible qu'après connexion.

B. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Créer une base de données

J'ai utilisé la base de données MySQL créer au début du projet à laquelle j'ai apporté de nombreuses modifications pour répondre au cahier des charges : ajout ou modification de tables, création de nouvelles associations entre celles-ci.

Développer les composants d'accès aux données

J'ai utilisé Symfony 5.3 pour la partie back-end de l'application et donc Doctrine, l'ORM (Object Relational Mapping) de Symfony pour interagir avec la base de données.

Développer la partie back-end d'une application web ou web mobile

J'ai utilisé le back-end pour rendre l'application administrable afin de gérer les sessions de recrutement, les candidats et les entretiens.

2. Résumé du projet

Création d'un outil interne de gestion de candidatures pour Débutant(e) Accepté(e) en Symfony.

Afin de valider ma formation de Développeur Web et Web Mobile, j'ai effectué un stage de 8 semaines au sein de Débutant(e) accepté(e), une entreprise coopérative du numérique basée à Besançon. Cette entreprise permet aux développeurs diplômés ou autodidactes de passer la barrière de la "première expérience exigée" pour une montée en compétences progressive.

De par sa nature, l'entreprise propose des CDDs renouvelables pour une durée totale maximale de 2 années. Les sessions de recrutement sont donc fréquentes. Actuellement, un fichier Excel est utilisé, ce qui n'est pas pratique, d'où le besoin de développer un outil interne de suivi des candidatures de manière plus efficace, et de supprimer ainsi le risque d'oubli d'un candidat. L'outil est accessible aux seuls administrateurs.

Ce projet a été commencé par un précédent stagiaire. Par manque de temps, seule la gestion des candidats a été réalisée sans tenir compte de la notion de session de recrutement.

Une première phase d'analyse m'a permis d'étoffer le cahier des charges et définir le workflow de l'application qui décrit de manière exhaustive l'évolution de l'état d'un candidat lors d'une session de recrutement. J'ai réalisé les maquettes de l'ensemble des interfaces ainsi que les interactions entre celles-ci. Le modèle de données a été revu, corrigé et complété.

Cette étape a été cruciale pour préciser les besoins de l'entreprise et en exprimer de nouveaux au fur et à mesure de la réalisation des tâches. Les principales fonctionnalités que j'ai réalisées peuvent se résumer ainsi :

- Gestion des candidatures (CRUD),
- Démarrage d'un recrutement en sélectionnant des candidats dans la liste d'attente avec la possibilité d'en ajouter au fur et à mesure,
- Gestion de plusieurs sessions de recrutement simultanément et possibilité de passer de l'une à l'autre,
- Suivi d'un recrutement sous la forme d'un tableau de bord qui permet une vision d'ensemble des candidats, de leurs entretiens, et des actions à mener en fonction de l'étape dans laquelle ils se trouvent : ajout d'un entretien, sortie du recrutement avec retour à la liste d'attente si tel est leur souhait, ou archivage. Le tout rendu plus lisible grâce à un code couleur et des icônes,
- Clôture d'un recrutement : au terme des entretiens, les derniers candidats sont recrutés,
- Suppression des candidats archivés au bout de 3 mois de la base de données pour se conformer au RGPD,
- Possibilité de consulter l'historique des recrutements terminés.

De plus, j'ai réalisé la refonte du thème de l'application pour rester dans le graphisme du site vitrine de l'entreprise.



3. Cahier des charges, expression des besoins ou spécifications fonctionnelles du projet

Ce projet a été réalisé lors d'un stage de huit semaines au sein de l'entreprise Débutant(e) accepté(e) un organisme qui permet aux jeunes développeurs diplômés ou autodidactes de passer la barrière de la "première expérience exigée" et favorise la progression de leurs talents. Débutant(e) accepté(e) propose une gamme de services liés à la numérisation de l'entreprise, de la Smart City, du Smart territoire et à l'accompagnement de Start-up.

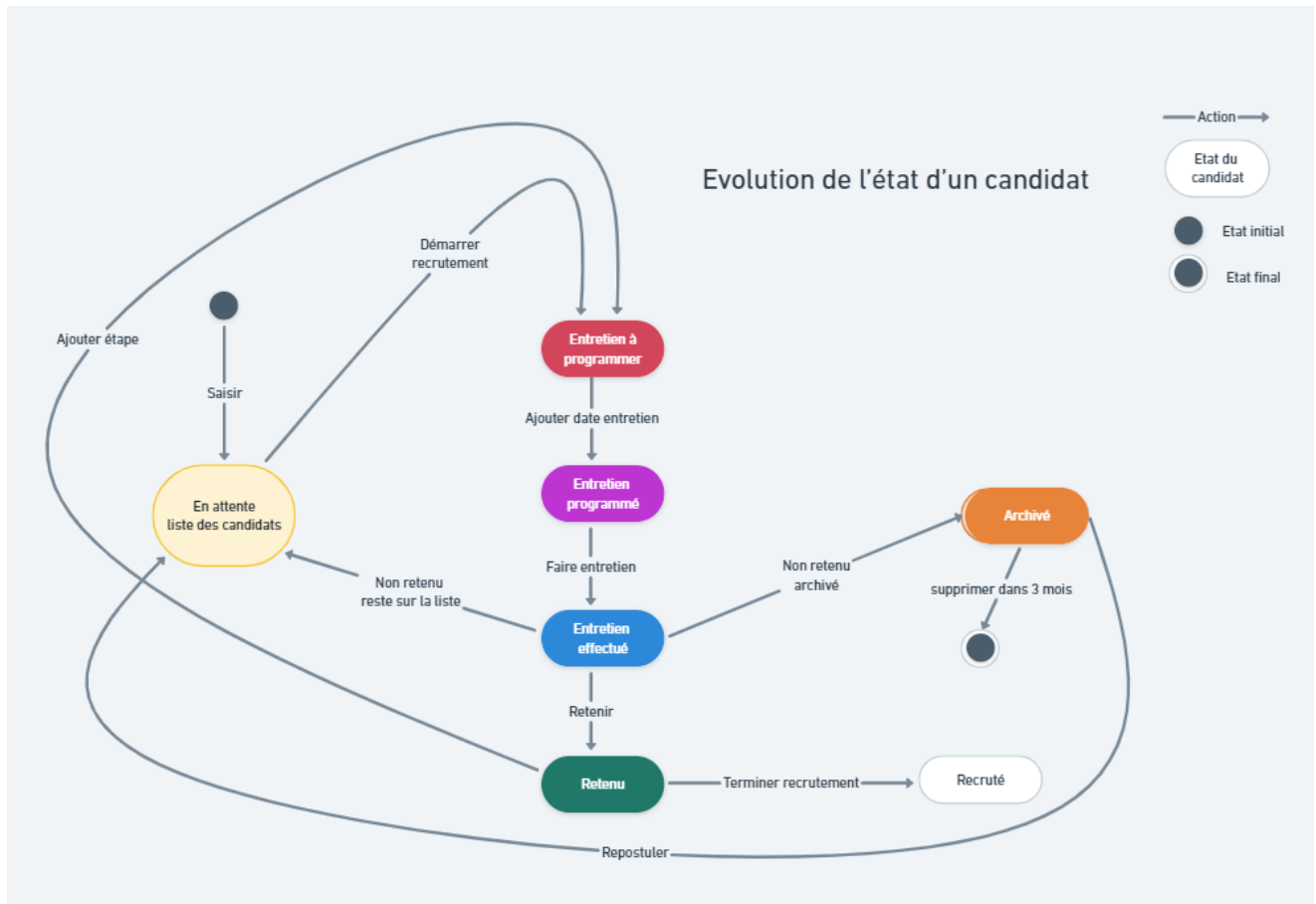
L'entreprise est dirigée par les 3 co-fondateurs, mon tuteur M. Michel Chevassu, M. Christophe Boutet (Direction) et M. Thomas Boyer (Lead développeur). Elle compte également quatre salariés et deux stagiaires.

L'objectif du stage est de réaliser un outil d'aide pour la gestion des sessions de recrutement. De par sa nature, le turn-over est important dans l'entreprise, les salariés sont embauchés pour une durée maximale de 2 ans, ce sont des contrats tremplins. Les candidatures arrivent tout au long de l'année, et il peut y avoir plusieurs sessions de recrutements par an. Il m'a été demandé de :

- Réaliser des maquettes de l'ensemble des interfaces et les interactions entre celles-ci,
- Intégrer l'ensemble des interfaces (administration facilitée à l'aide de tableaux de bord, de modales, d'édition à la volée, etc.),
- Réaliser une partie administrable pour permettre de :
 - Créer des candidats (Nom, prénom, numéro de téléphone, email, date de candidature, CV, lettre de motivation, réseaux sociaux, profil [non défini, back-end ou front-end], commentaire divers) et les gérer,
 - Créer des entretiens (date et heure de l'entretien, type [téléphone, visio, présentiel], motivation, technique, impressions générales) et de les gérer. Démarrer, clôturer des entretiens,
 - Lier les candidats à une session de recrutement et proposer les actions sur le candidat en fonction de l'avancement de ses entretiens (retenir, archiver, remettre en liste d'attente),
 - Pour les candidats recrutés, indiquer le type de poste sur lequel ils sont recrutés et la durée du contrat. Plusieurs candidats peuvent être embauchés pour le même type de poste.
- Le thème de l'application (charte graphique : logo, fonts, couleurs, etc.) doit être de préférence cohérent avec celui du site de l'entreprise,
- L'outil n'est accessible que par les administrateurs,
- Pour les candidats archivés, mettre en place un système qui les supprime automatiquement s'ils sont archivés depuis plus de 3 mois.

Lors de la phase d'analyse et de rédaction du cahier des charges, il y a eu plusieurs échanges avec mon tuteur de stage sur les besoins de l'entreprise. En formalisant, d'autres besoins ont été exprimés pour rendre l'outil plus intéressant comme la possibilité d'avoir plusieurs sessions de recrutement en parallèle.

J'ai proposé un workflow qui a été complété et validé par mon tuteur pour fixer la logique métier de l'outil à développer :



J'ai également réalisé l'ensemble des maquettes correspondant à l'enchainement des écrans (voir annexe A), celles-ci ont été également validées par mon tuteur de stage.

C'est à partir de ces documents que j'ai pu commencer à travailler. Il m'est arrivé d'apporter des modifications minimales et avoir ainsi un cycle de vie en spirale de mon application.



4. Spécifications techniques du projet

A. Outils de communication

Dans l'entreprise, un point quotidien est fait tous les matins avec toute l'équipe. Ainsi, chaque membre de l'équipe fait part de l'avancement de ses tâches et s'en voit assigné de nouvelles si besoin. Le reste de la journée, nous pouvons échanger car nous partageons le même bureau, nous utilisons également le logiciel Mattermost, un service de messagerie instantanée libre conçu comme un chat interne pour les organisations et les entreprises. J'ai utilisé cet outil pour demander de l'aide, des informations techniques ou organisationnelles à mes collègues.

Compte tenu du contexte sanitaire, il nous a été proposé deux jours de télétravail par semaine. Dans ce cas, l'outil Jitsi Meet est mis en place pour faciliter les discussions vocales et partages d'écrans à distance. L'équipe est connectée à un salon vocal pour échanger en direct si besoin.

B. Outils de versioning

L'entreprise utilise Gitlab, une plateforme de développement logiciel open source avec contrôle de version intégré, suivi des problèmes, revue de code, intégration, distribution et déploiement continu. Après avoir créé un compte, j'ai reçu les invitations nécessaires pour pouvoir travailler sur mon projet et en consulter d'autres.

L'utilisation de cet outil m'a permis d'effectuer des commits mes modifications régulièrement pendant la journée, et de pousser l'ensemble des développements de la journée tous les soirs sur une branche de développement créée pour moi durant le stage.

À la fin du développement d'une fonctionnalité par exemple, je faisais une merge request et ainsi mon travail était relu par le lead developer et intégrée à la branche principale de développement.

C. Environnement de développement

Ordinateur

J'ai utilisé l'ordinateur sous Windows 10 mis à ma disposition par l'Access Code School, avec Ubuntu en WSL car les projets de l'entreprise tournent sous docker et Linux.

WSL (Windows Subsystem for Linux) est une couche de compatibilité permettant d'exécuter des exécutables binaires Linux de manière native sur Windows 10 entre autres.

Editeur de code

J'ai utilisé Visual Studio Code, un éditeur de code extensible développé par Microsoft pour Windows et j'ai aussi installé des extensions utiles pour gagner en performance et rapidité de codage.

Navigateur web

J'ai développé principalement sur le navigateur Chrome. J'ai vérifié les fonctionnalités également sur Firefox et Edge.

Symfony 5.3

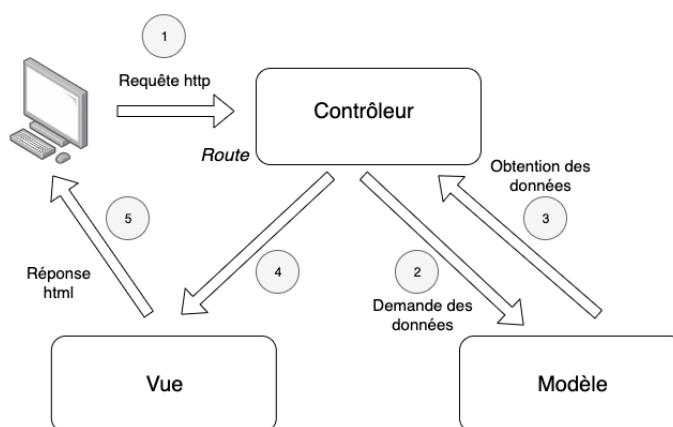
Lorsque j'ai commencé mon stage, le projet sur lequel j'ai travaillé possédait déjà une version sommaire développée avec Symfony. En accord avec mon tuteur, j'ai donc poursuivi le développement avec cette technologie.

Symfony est un ensemble de composants PHP ainsi qu'un framework MVC (Model View Controller) libre écrit en PHP. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d'accélérer le développement d'un site web.

Symfony¹ propose entre autres à partir de sa première version :

- une séparation du code en trois couches, selon le modèle MVC, pour une plus grande maintenabilité et évolutivité,
- des performances optimisées et un système de cache afin d'assurer des temps de réponse optimaux,
- une gestion des URL parlantes, permettant à une page d'avoir une URL distincte de sa position dans l'arborescence (routes),
- un système de configuration en cascade utilisant le langage de description YAML,
- l'internationalisation native,
- le support d'AJAX,
- une architecture extensible permettant créations et utilisations de plugins.

Revenons sur le modèle MVC illustré par le schéma suivant. Ce modèle permet de séparer les données, la couche métier qui gère ces données et les interfaces utilisateur.

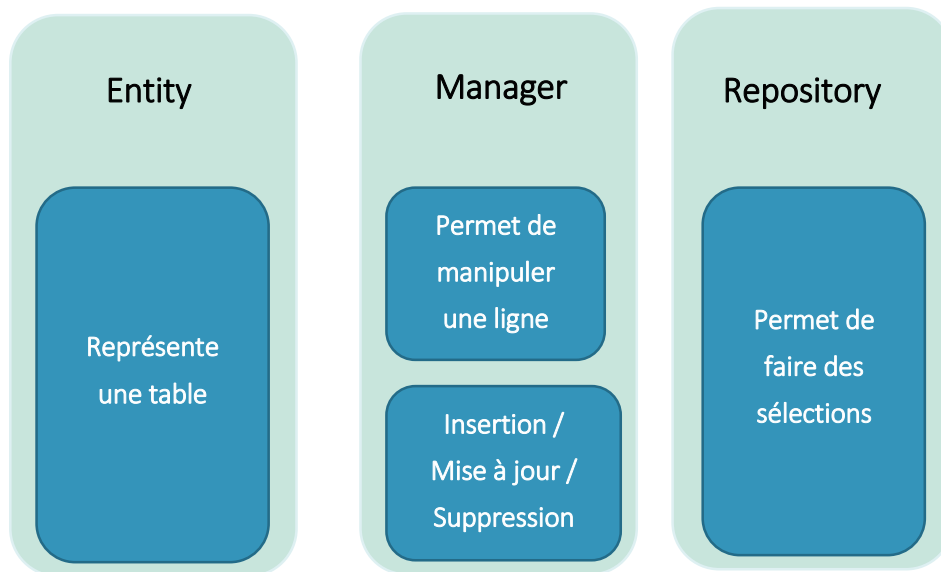


Modèle MVC - source : <https://fr.wikipedia.org/>

¹ Source : <https://fr.wikipedia.org/wiki/Symfony>



Pour le **M** de MVC, L'ORM (Object Relational Mapping) qu'utilise Symfony est Doctrine. Il permet de manipuler les données sans écrire directement des requêtes SQL mais en passant par des objets comme l'illustre le schéma suivant :



En cas de modification du modèle de données, Symfony permet de facilement mettre à jour la base de données grâce à des fichiers de migration générés. Cela est très pratique en particulier lorsqu'on travaille à plusieurs sur le même projet (pas besoin de partager la base de données, les fichiers de migration se trouvent déjà dans le projet).

Le **C** de MVC correspond à tous les contrôleurs : avec Symfony, on écrit généralement un contrôleur pour chaque entité, qui permet de gérer toutes les actions (routes) relatives à celle-ci.

Enfin, le **V** de MVC est l'ensemble des vues. Les vues sont les interfaces avec lesquelles l'utilisateur interagit avec l'application. Dans Symfony, on utilise le moteur de templates Twig.

Serveur de développement

L'équipe utilise un repository infra dev, afin d'initialiser les outils de développements relatifs aux projets de l'entreprise. Après avoir cloné le repository, j'ai pu administrer les services via docker-compose. Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels. Il ne s'agit pas de virtualisation, mais de conteneurisation. Ce système permet d'accroître la flexibilité et la portabilité d'exécution d'une application, laquelle va pouvoir tourner de façon fiable et prévisible sur une grande variété de machines hôtes.

Docker Compose est un outil permettant de définir et d'exécuter des applications Docker multi-conteneurs. Avec Docker Compose, un fichier YAML permet de configurer les services de l'application. Ensuite, une seule commande permet de créer et démarrer tous les services à partir d'une configuration.

Pour créer et démarrer les services j'ai utilisé la commande ***docker-compose up -d*** (-d active le mode détaché qui exécute les conteneurs en arrière-plan).

Pour stopper et détruire les services, j'ai utilisé la commande ***docker-compose down***. Je peux aussi lister les services avec la commande ***docker-compose ps***. Les services instanciés sont :

- MariaDB
- PostgreSQL
- Traefik qui sert de reverse proxy, il permet d'accéder aux conteneurs via un nom de domaine
- Portainer qui permet de gérer les conteneurs graphiquement
- PhpMyadmin qui permet de gérer la base de données MySQL
- PgAdmin qui permet de gérer la base de données Postgres
- Mailhog qui permet de gérer les emails

J'ai installé les conteneurs grâce à Docker. La commande ***docker-compose exec watcher yarn watch*** me permet de gérer la compilation des fichiers lorsque je modifie les fichiers scss ou que j'ajoute des assets dans le projet.

Webpack encore

Afin de gérer les différentes dépendances de mon projet, le module Webpack a été ajouté à Symfony. Il permet d'empaqueter automatiquement les assets (images, fichiers SCSS et JavaScript) lors du déploiement de l'application. Il génère des fichiers compilés à partir de ressources définies.

Bootstrap 5

Enfin pour la partie front-end, j'ai utilisé le framework Bootstrap, qui est composé de fichiers CSS et JavaScript. Ces fichiers contiennent des règles prédéfinies encapsulées dans des classes, et nous utilisons ces classes pour appliquer un ensemble de styles à des éléments HTML. De plus, des composants sont mis à disposition comme des barres de navigation, des modales, des accordéons, etc. qu'on peut directement implémenter dans nos projets.

Ce framework permet de développer facilement des interfaces adaptables à tous les formats d'écrans (responsive). Pour mon projet j'ai utilisé le thème bootswatch (<https://bootswatch.com/>).

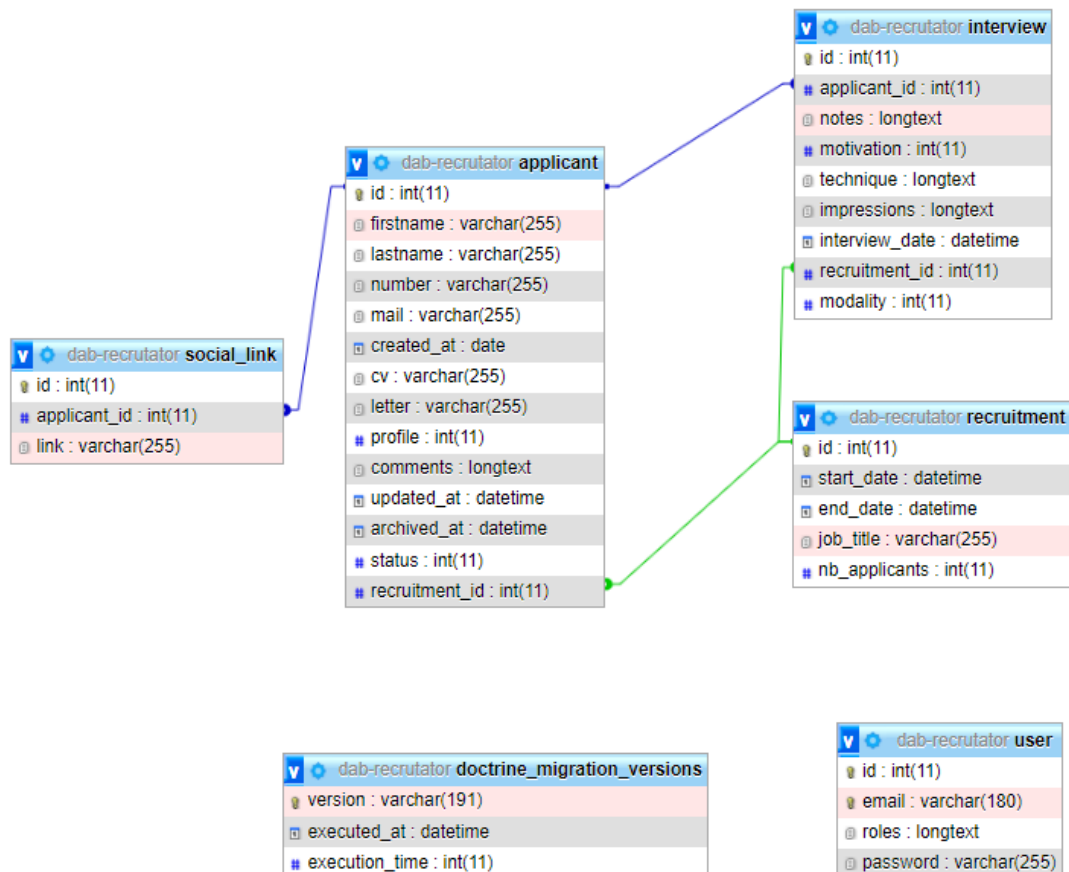
Il est vrai que Bootstrap permet de gagner du temps de développement mais il est souvent nécessaire de modifier le code pour répondre à nos besoins particuliers.



5. Réalisations comportant les extraits de code les plus significatifs

A. Modèle de données

En conformité avec les besoins exprimés dans le cahier des charges, le modèle de données suivant a été retenu :



B. Les entités

Symfony fournit une couche logicielle pour créer et modifier des entités en utilisant des lignes de commande. Les principales entités dans mon projet sont :

- *Applicant* (candidat : personne qui postule à un poste dans l'entreprise),
- *Interview* (entretien d'embauche),
- *Recrutement* (session de recrutement sur un type de poste).

Le bundle `makerBundle` disponible sur Symfony, permet de générer du code dans l'application en fonction de nos besoins. Ici pour créer une entité, il suffit de taper dans le terminal la commande :

`php bin/console make:entity` . Il faut ensuite écrire le nom de l'entité et définir toutes ses propriétés. Le bundle va automatiquement créer un fichier php contenant la classe correspondante avec le nom de l'entité, toutes les propriétés, les getters et les setters.

Pour mettre en place les entités dans la base de données, Doctrine utilise un système de migrations. À chaque changement, au niveau des entités, la commande **`php bin/console make:migration`** permet de créer un fichier de version qui stocke toutes les actions qui doivent être effectuées dans la base de données. Il faut ensuite appliquer ces changements avec la commande **`php bin/console doctrine:migrations:migrate`**. Ce système est pratique car il permet d'avoir un système de versionning sur Symfony.

Comme certaines de ces entités existaient déjà, la commande de création des entités m'a beaucoup servi car elle permet aussi de modifier les entités existantes pour ajouter de nouvelles propriétés ou relations.

```
yaminajouille@PORT201910-098:~/recrutator$ bin/console make:entity
Creating dab-recrutator_app_run ... done

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

Your entity already exists! So let's add some new fields!

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

Pour modifier ou supprimer des propriétés ou des relations, il est nécessaire de le faire directement dans la classe correspondant à l'entité et ensuite générer un fichier de migration.

Une des principales entités de mon projet est l'entité *Applicant* (candidat), je vais donc à présent détailler certains des fichiers du modèle MVC permettant sa gestion.

Pour commencer, voici le fichier décrivant l'entité *Applicant* :

```
class Applicant
{
    const PROFILE_UNDEFINED = 0;
    const PROFILE_FRONT_END = 1;
    const PROFILE_BACK_END = 2;
    const PROFILE_LABEL = [
        self::PROFILE_UNDEFINED => "Non défini",
        self::PROFILE_FRONT_END => "Front-end",
        self::PROFILE_BACK_END => "Back-end",
    ];
}
```



```
const STATUS_INTERVIEW_TO_SCHEDULE = 0;
const STATUS_INTERVIEW_SCHEDULED = 1;
const STATUS_INTERVIEW_DONE = 2;
const STATUS_APPLICANT_KEPT = 3;
const STATUS_PENDING = 4;
const STATUS_ARCHIVED = 5;
const STATUS_RECRUITED = 6;
const STATUS_LABEL = [
    self::STATUS_INTERVIEW_TO_SCHEDULE => "Entretien à programmer",
    self::STATUS_INTERVIEW_SCHEDULED => "Entretien programmé",
    self::STATUS_INTERVIEW_DONE => "Entretien effectué",
    self::STATUS_APPLICANT_KEPT => "Retenu",
    self::STATUS_PENDING => "En attente, liste des candidats",
    self::STATUS_ARCHIVED => "Archivé",
    self::STATUS_RECRUITED => "Recruté",
];

public function __construct()
{
    $this->socialLinks = new ArrayCollection();
    $this->interviews = new ArrayCollection();
    $this->status = self::STATUS_PENDING;
}

/**
 * @ORM\Id
 * @ORM\GeneratedValue
 * @ORM\Column(type="integer")
 */
private ?int $id = null;

/**
 * @ORM\Column(type="string", length=255)
 * @Assert\NotNull
 * @Assert\Regex("/^[a-zA-ZÀ-ü ]*$/", message="Le prénom ne doit comporter que
des caractères alphabétiques")
 * @Assert\Length(min=2)
 */
private string $firstname;

...
```

La constante STATUS correspond à l'état dans lequel se trouve le candidat dans le workflow. J'ai utilisé des correspondances entre le libellé et le nombre entier correspondant pour plus de lisibilité dans mon code.

On constate aussi que Symfony permet de positionner des contraintes sur les propriétés. Par exemple, le prénom (firstname) ne doit pas être null, sa longueur est de 2 caractères minimum et ne doit comporter que des caractères alphabétiques comme l'indique l'expression régulière associée.

Il est possible également de personnaliser les messages d'erreur.

C. Manipulation des données

Les contrôleurs dans Symfony permettent la manipulation des entités. Pour mon entité *Applicant*, j'ai écrit plusieurs méthodes dans le contrôleur associé (*ApplicantController.php*), chacune de ces méthodes correspond à un traitement sur cette entité. Par exemple : listes des candidats en fonction de leur statut, création, modification, visualisation, suppression, etc.

```
//APPLICANTS
#[Route('/', name: 'applicants_list', methods: ['GET'])]
public function index(): Response
{ ...
}

#[Route('/applicants/archived', name: 'applicants_archived', methods: ['GET'])]
public function archived(): Response
{ ...
}

#[Route('/applicants/recruited', name: 'applicants_recruited', methods: ['GET'])]
public function recruited(): Response
{ ...
}

#[Route('/applicant/new', name: 'applicant_new', methods: ['GET', 'POST'])]
public function new(Request $request): Response
{ ...
}

#[Route('/applicant/{applicant}/edit', name: 'applicant_edit', methods: ['GET', 'POST'])]
public function edit(Applicant $applicant, Request $request, EntityManagerInterface $em): Response
{ ...
}

#[Route('/applicant/{applicant}', name: 'applicant_show', methods: ['GET'])]
public function show(Applicant $applicant): Response
{ ...
}

#[Route('/applicant/{applicant}/delete', name: 'applicant_delete', methods: ['POST'])]
public function delete(Request $request, Applicant $applicant): Response
{ ...
}
```




Une méthode importante de ce contrôleur est celle permettant de changer de statut au candidat au cours d'une session de recrutement.

```
#[Route('/applicant/changeStatus', name: 'applicant_changeStatus', methods: ['GET'])]
public function changeStatus(Request $request): Response
{
    $applicant = $this->repository->find($request->query->get('idApplicant'));
    $status = $request->query->getInt('status');
    // add applicant in recruitment
    if ($status == Applicant::STATUS_INTERVIEW_TO_SCHEDULE) {
        // set recruitment in applicant
        $recruitment = $this->getDoctrine()->getRepository(Recruitment::class)->find($request->query->get('idRecruitment'));
        // update total number of applicants in database
        $recruitment->setNbApplicants($recruitment->getNbApplicants() + 1);
        $applicant->setRecruitment($recruitment);
    }
    // program date new interview
    if ($status == Applicant::STATUS_INTERVIEW_SCHEDULED) {
        // create interview
        $recruitment = $this->getDoctrine()->getRepository(Recruitment::class)->find($request->query->get('idRecruitment'));
        $interview = new Interview($applicant, $recruitment, new DateTime());
        $this->em->persist($interview);
        $this->em->flush();
    }
    // remove applicant in recruitment
    if ($status == Applicant::STATUS_PENDING || $status == Applicant::STATUS_ARCHIVED) {
        // unset recruitment in applicant
        $applicant->setRecruitment(NULL);
    }
    if ($status == Applicant::STATUS_ARCHIVED) {
        $applicant->setArchivedAt(new \DateTime());
    }
    $applicant->setStatus($status);
    $this->em->persist($applicant);
    $this->em->flush();
    $this->addFlash("success", "Statut du candidat " . $applicant->getFirstname() . " " . $applicant->getLastname() . " modifié !");
    if ($status == Applicant::STATUS_PENDING) {
        return $this->redirectToRoute('applicants_list');
    }
    return $this->redirectToRoute('applicant_show', [
        'applicant' => $applicant->getId(),
    ]);
}
```

Cet exemple de méthode nous montre la puissance de Symfony qui permet d'accéder et/ou de modifier l'ensemble des données du modèle (autres entités), cela contribue aussi à la clarté du code orienté objet.

Une autre méthode importante est l'édition à la volée des champs correspondant à l'entité *Applicant*.

Cette fonctionnalité n'était pas présente dans le cahier des charges initial. Après quelques recherches, j'ai trouvé une librairie : *X-editables* qui permet facilement la mise en œuvre de cette fonctionnalité. Malheureusement, la dernière version de cette librairie date de 2013, et après concertation avec mon tuteur de stage, nous avons décidé de ne pas l'utiliser et de développer cette fonctionnalité.

Ce choix m'a ainsi permis d'élargir la couverture du référentiel en me faisant coder plus côté client (JavaScript, JQuery et Ajax).

Voici donc la méthode *editField*, extraite du contrôleur de l'*Applicant* :

```

#[Route('/applicant/{applicant}/editField', name: 'applicant_editField', methods: ['POST'])]
public function editField(Request $request, ValidatorInterface $validator, Applicant $applicant): Response
{
    try {
        $errors = null;
        $fieldName = $request->request->get('name');
        $fieldValue = $request->request->get('value');
        // special cases for add and delete social link
        if ($fieldName == 'add-social') {
            // add social link
            $socialLink = new SocialLink();
            $socialLink->setLink($fieldValue);
            $errors = $validator->validate($socialLink);
            if (count($errors) > 0) {
                return $this->json(['code' => Response::HTTP_INTERNAL_SERVER_ERROR, 'errors' => $errors]);
            } else {
                $applicant->addSocialLink($socialLink);
                $this->em->persist($socialLink);
                $this->em->persist($applicant);
                $this->em->flush();
            }
        } else if ($fieldName == 'delete-social') { ...
        } else {
            // edit simple field
            $applicant->setField($fieldName, $fieldValue);
            // validate applicant
            $errors = $validator->validate($applicant);
            if (count($errors) > 0) {
                return $this->json(['code' => Response::HTTP_INTERNAL_SERVER_ERROR, 'errors' => $errors]);
            } else {
                $this->em->persist($applicant);
                $this->em->flush();
            }
        }
    } catch (\Throwable $th) {
        return $this->json(['code' => Response::HTTP_INTERNAL_SERVER_ERROR, 'errors' => $th->getMessage()]);
    }
    return $this->json(['code' => Response::HTTP_OK]);
}

```

D. Gestion du front, les vues

Dans *Symfony*, les vues sont gérées par *Twig* qui permet de générer des pages html dynamiques depuis les contrôleurs.

Par exemple, pour les candidats, la vue qui permet d'en afficher un est le fichier *showApplicant.twig.html*.



```
{# Fistname and Lastname #}      You, il y a 2 semaines • [feat] : aditable fields for applicant view - nex...
<li class="list-group-item">
  <div class="row">
    <div class="col-6">
      <h6 class="card-subtitle mb-2 text-muted">
        <i class="fas fa-pen fa-sm" id="firstname-pen"></i>
        Prénom :
      </h6>
      <span class="editable-text" id="firstname">{{ applicant.firstname }}</span>
      <div class="editable-form" id="firstname-form">
        <input type="text" id="firstname-value" name="firstname" value="{{ applicant.firstname }}" size="17">
        <a class="editable-submit" id="firstname-check" data-id="{{ applicant.id }}">
          <i class="fa-solid fa-check"></i>
        </a>
        <a class="editable-cancel" id="firstname-cancel">
          <i class="fa-solid fa-xmark"></i>
        </a>
        <p class="editable-error" id="firstname-error"></p>
      </div>
    </div>
    <div class="col-6">
      <h6 class="card-subtitle mb-2 text-muted">
        <i class="fas fa-pen fa-sm" id="lastname-pen"></i>
        Nom :
      </h6>
      <span class="editable-text" id="lastname">{{ applicant.lastname }}</span>
      <div class="editable-form" id="lastname-form">...
    </div>
  </div>
</li>
{# Modifié le #}
<li class="list-group-item">
  <h6 class="card-subtitle mb-2 text-muted">
    Modifié le :
  </h6>
  {{ applicant.updatedAt|date('d/m/Y') }}
</li>
```

De plus, on peut depuis une vue inclure le résultat complet de l'appel à un contrôleur.

J'ai fait la proposition de cette vue candidat à mon tuteur qui l'a validée. Cela a l'avantage de tout centraliser au niveau d'une seule vue et d'avoir en un simple coup d'œil les informations du candidat et l'historique de ses entretiens.


```
{% if applicant.status == constant('App\\Entity\\Applicant::STATUS_INTERVIEW_TO_SCHEDULE') %}
  {{ render(controller('App\\Controller\\InterviewController:dateInterview',{idApplicant:applicant.id})) }}
{% else %}
```

Cette fonctionnalité m'a permis donc de créer une vue qui regroupe l'ensemble des informations relatives au candidat :

- Propriétés du candidat,
- Recrutements éventuels auxquels le candidat a participé,
- Détails des entretiens éventuels effectués par le candidat.

J'ai utilisé pour les recrutements et les entretiens le composant *Boostrsatp* "accordion"

De plus, l’affichage est dynamique et prend en compte le statut du candidat : ainsi le recrutement apparaît que s’il est en cours (accordéon replié si le recrutement est terminé), et les actions disponibles reflètent l’état du candidat dans le workflow.



LISTES DES CANDIDATS ▾

RECRUTEMENTS EN COURS

RECRUTEMENTS TERMINÉS

↔

FICHE CANDIDAT

✎ Prénom :

Thierry ✓ ✕

✎ Nom :

Blot

✎ Modifié le :

13/03/2022

✎ Candidat depuis le :

19/02/2022

✎ Téléphone :

0102030405

✎ Email :

ferrand.richard@example.com

✎ Documents :

CV

LETTRE

✎ Réseaux sociaux :

LIEN

✎ Profil :

Non défini

✎ Commentaires :

Atque dignissimos velit culpa. Ab aut soluta sed nobis velit. Unde laudantium ut temporibus necessitatibus molestiae sapiente.

Suivi des entretiens pour le recrutement du 13/03/2022 pour le poste Développeur junior

MODIFIER UN ENTRETIEN

*Date de l'entretien

13/03/2022 21:43

*Modalité

Téléphone ▾

Notes

Quia id commodi illum. Enim ea sequi accusamus nam possimus cum.

Motivation

☆☆☆☆☆

Technique

Et impedit dolor facilis. In et illo est mollitia ratione corporis perspicatis. Libero nemo nam molestiae reprehenderit minima autem reiciendis. Dicta aliquam iste qui est est.

Impressions

Voluptatem omnis non aut consequatur ea quae. Libero quae est voluptas quis. Sed minus vitae ut tempora voluptatem excepturi praesentium. Aliquid aut accusamus adipisci fugit quaerat

Décision finale

SAUVEGARDER

RETENIR

RECANDIDATER

ARCHIVER

Motif

13/03/2022

Pariatur sit et natus ipsa. Beatae nemo eaque et sint saepe aut eius. Dolore expedita rerum in neque. Eos doloribus laudantium sint.

13/03/2022

Dolor voluptas laudantium voluptas sequi. Dolore et quidem quidem unde aut est adipisci. Necessitatibus sed cupiditate inventore cum qui voluptatibus. Dolores et qui dolorem eum ut quam vero.

Cette vue est celle d’un candidat dont l’entretien a été réalisé et qui est en attente de décision (status = STATUS_INTERVIEW_DONE).

Si le candidat a le statut INTERVIEW_TO_SCHEDULE, il faut donc lui programmer un entretien, la vue ci-dessus s’adapte en affichant un datepicker pour sélectionner une date.



Si le candidat a le statut `APPLICANT_KEPT` cela veut dire qu'il a été retenu pour cette étape, on peut lui ajouter une autre étape.

E. Édition à la volée

Dans la vue *Applicant*, les champs de l'entité correspondant ont la classe " *editable* ". Comme je l'ai mentionné plus haut, le besoin a été exprimé de rajouter, en plus de l'édition classique du formulaire rattaché à l'entité *Applicant* une édition à la volée.

Symfony permet de créer des formulaires grâce à son système de *Form Type*. La vérification se fait coté client avec la validation HTML5 et côté serveur en fonction du type de données attendu.

Par contre pour l'édition à la volée, j'ai dû ajouter une méthode au contrôleur et j'ai appelé moi-même le validateur de Symfony car il n'y avait pas d'objet formulaire. Cette méthode est appelée de façon asynchrone en utilisant Ajax depuis la fiche du candidat. Lorsque l'on clique sur un champ, celui-ci se transforme en formulaire qui permet de modifier un seul champ à la fois sans recharger la page.

Voici un extrait du code JavaScript :

```
// editable fields
$(".editable-submit").click(function () {
  if (currentNode) {
    let id = this.dataset.id;
    let route = "/applicant/" + id + "/editField";
    let datas = new FormData();
    datas.append("name", currentNode.id);
    datas.append("value", $("#" + currentNode.id + "-value").val());
    fetch(route, {
      method: "POST",
      body: datas,
    })
      .then(function (response) {
        return response.json();
      })
      .then(function (response) {
        if (response.code != 200) {
          //$("#" + currentNode.id + "-error").text(response.errors);
          $("#" + currentNode.id + "-error").text(response.errors.violations[0].title);
        } else {
          if (currentNode) {
            // test if select
            if ($("#" + currentNode.id + "-value").is("select")) {
              $("#" + currentNode.id).text(
                $("#" + currentNode.id + "-value option:selected").text();
              )
            } else if (currentNode.id == "createdAt") {
              // update display for date - flatpicker format
              var date = new Date($("#" + currentNode.id + "-value").val());
              $("#" + currentNode.id).text(new Intl.DateTimeFormat("fr-FR").format(date));
            } else if (currentNode.id == "add-social") {
              location.reload();
            } else {
              $("#" + currentNode.id).text($("#" + currentNode.id + "-value").val());
            }
          }
          hideCurrentForm();
        }
      });
  }
});
```

Tous les champs sont éditables à la volée sauf l'ajout du CV et de la lettre de motivation. Par manque de temps je n'ai pas pu terminer cette tâche. Mais il est tout de même possible d'ajouter ces documents en passant par le formulaire d'édition classique qui est géré par Symfony. La route suivante est utilisée

```
#[Route('/applicant/{applicant}/edit', name: 'applicant_edit', methods: ['GET', 'POST'])]
```

Au lieu de :

```
#[Route('/applicant/{applicant}/editField', name: 'applicant_editField', methods: ['POST'])]
```

Dans Recrutator, le téléchargement des pièces jointes utilise le bundle (paquet) *VichUploader*.



F. Flatpicker

Dans mon projet, j'ai utilisé un datepicker : *Flatpicker* (<https://flatpickr.js.org/>). C'est un sélecteur de date et d'heure léger et puissant. Ce composant est ergonomique et adaptable. Il ne dépend d'aucune librairie.

À l'ajout de cette dépendance avec la commande `bin/yarn add nom_librairie`, un dossier est créé dans `node_modules` et une ligne dans le fichier `package.json`. (extrait suivant), ça met également à jour le fichier `yarn.lock` pour appliquer la modification.

```
"dependencies": {
  "@fortawesome/fontawesome-free": "^6.0.0",
  "bootstrap": "^5.1.3",
  "bootswatch": "^5.1.3",
  "datatables.net-bs5": "^1.11.3",
  "datatables.net-select-bs5": "^1.3.4",
  "flatpickr": "^4.6.9",
  "raty-js": "^3.1.1"
}
```

Pour pouvoir utiliser ce composant, je l'ai importé dans un fichier JavaScript et j'ai positionné les options en fonction de mes besoins.

```
import "flatpickr";
import "flatpickr/dist/l10n/fr.js";
$(".datepicker").flatpickr({
  locale: "fr",
  enableTime: true,
  dateFormat: "d/m/Y H:i",
});

$(".datepicker-update-applicant").flatpickr({
  locale: "fr",
  dateFormat: "Y-m-d",
  altInput: true,
  altFormat: "d/m/Y",
});

$(".datepicker-new-applicant").flatpickr({
  locale: "fr",
  dateFormat: "d/m/Y",
});
```

En effet, pour la création d'un candidat par exemple, il n'est pas nécessaire d'indiquer l'heure alors que cette information est importante pour la programmation d'un entretien.

Un fichier `datepicker.scss` existe également dans mon projet pour adapter le style de ce composant au style de Recrutator.

Le principe est semblable pour l'utilisation d'un autre composant : les datatables de bootstrap5.

6. Présentation du jeu d'essai de la fonctionnalité la plus représentative

La fonctionnalité la plus représentative est la gestion d'une session de recrutement donc :

- Création et gestion des candidats,
- Création et déroulements de la session de recrutement.

Création et gestion des candidats

Il est possible de démarrer un recrutement avec une sélection de candidats dans la liste d'attente ou de les ajouter au fur et à mesure.

On commence donc par créer les candidats, ces derniers se retrouveront dans la liste d'attente.

Cette page est la page d'accueil de l'application. Les trois onglets contiennent l'ensemble des candidats présents dans la base de données, avec un statut différent par onglet.

The screenshot shows the 'Débutant-e AccePT' application interface. The top navigation bar is dark blue with the logo on the left and three menu items: 'LISTES DES CANDIDATS', 'RECRUTEMENTS EN COURS', and 'RECRUTEMENTS TERMINÉS'. The 'CANDIDATS EN ATTENTE' section is active, showing a sub-header with a green plus icon and a button 'Ajouter un candidat'. Below this are three tabs: 'Candidats en attente' (selected), 'Candidats recrutés', and 'Candidats archivés'. A search bar labeled 'Rechercher:' is on the right. The main table has three columns: 'NOM PRÉNOM', 'CANDIDAT DEPUIS LE:', and 'ACTIONS'. It lists two candidates: BERTRAND David (27/12/2021) and COHEN Hélène (30/12/2021). Each row has a checkbox and three action icons (eye, edit, delete). At the bottom, it says 'Affichage de 1 à 2 sur 2 éléments' and features a green button 'DÉMARRER UN RECRUTEMENT'.

⬆ NOM PRÉNOM	⬆ CANDIDAT DEPUIS LE :	ACTIONS
<input type="checkbox"/> BERTRAND David	27/12/2021	
<input type="checkbox"/> COHEN Hélène	30/12/2021	

Affichage de 1 à 2 sur 2 éléments

DÉMARRER UN RECRUTEMENT

Une modale propose le formulaire de saisie. Une validation est faite pour chaque champ côté client (champs requis) et côté serveur (en fonction des propriétés de l'entité *Applicant*).



CRÉER UN CANDIDAT

*Date de candidature

12/03/2022

*Prénom

Cette valeur ne doit pas être nulle.

*Nom

12test

Le nom ne doit comporter que des caractères alphabétiques

*Téléphone

031245578

Le numéro de téléphone n'est pas valide

*Email

testgmail.com

Le format du mail n'est pas valide

CV

CHOISIR UN FICHIER

CV.pdf

Lettre

CHOISIR UN FICHIER

Lettre_motivation.pdf

*Profil

Non défini

Commentaires

Test de saisie d'un candidat avec des champs non valides

Réseaux sociaux




AJOUTER

*Lien

<https://github.com/YJouille>



CRÉER

Lors de la saisie, des vérifications sont faites. Une fois les données correctes saisies, le candidat est effectivement créé et ajouté dans la liste des candidats en attente :

<input type="checkbox"/>	JOUILLE Yamina	14/03/2022	  
--------------------------	----------------	------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Dans la base de données, le candidat est bien ajouté dans la table "*applicant*". On note que pour le moment il n'est dans aucune session de recrutement (*recrutement_id* = *NULL*) et son statut est 4 donc en liste d'attente.

+ Options

	id	firstname	lastname	number	mail	created_at	cv	letter	profile	comments	updated_at	archived_at	status	recruitment_id
<input type="checkbox"/> Éditer  Copier  Supprimer	1494	Yamina	Jouille	0384731665	yamina.jouille@gmail.com	2022-03-14	NULL	NULL	2	En reconversion.	2022-03-14 17:19:04	NULL	4	NULL

Maxime saoliente iosam



Création et déroulement de la session de recrutement

Une fois les candidats créés, ils apparaissent dans la liste d'attente, c'est à partir de cette liste qu'on peut démarrer un recrutement.

CANDIDATS EN ATTENTE

Rechercher:

NOM PRÉNOM	CANDIDAT DEPUIS LE :	ACTIONS
BERTRAND David	27/12/2021	
COHEN Hélène	30/12/2021	
JOUILLE Yamina	14/03/2022	

Affichage de 1 à 3 sur 3 éléments

[DÉMARRER UN RECRUTEMENT](#)

Une modale demande de renseigner le poste à pourvoir, le champ doit être renseigné.

Poste à pourvoir ×

***Veuillez renseigner l'intitulé du poste pour le recrutement à démarrer :**

[FERMER](#) [DÉMARRER](#)

Le recrutement est ainsi créé avec les candidats sélectionnés.

Nous arrivons sur le tableau de bord du recrutement que nous venons de créer.

RECRUTEMENT DÉMARRÉ LE : 14/03/2022

Poste à pourvoir : Développeur web junior

Rechercher:

CANDIDATS	ENTRETIEN N°1	ACTION
COHEN Hélène		
BERTRAND David		
JOUILLE Yamina		

Affichage de 1 à 3 sur 3 éléments

Ajouter un candidat au recrutement : [AJOUTER](#)

[TERMINER LE RECRUTEMENT](#)

Dans la base de donnée, le recrutement a bien été créé dans la table " *recrutement* " avec comme nombre de candidat au départ 3

+ Options

			id	start_date	end_date	job_title	nb_applicants		
<input type="checkbox"/>		Éditer	Copier	Supprimer	108	2022-01-07 00:07:04	2022-02-05 19:37:12	Développeur junior	10
<input type="checkbox"/>		Éditer	Copier	Supprimer	109	2022-03-14 14:57:53	NULL	Lead développeur	10
<input type="checkbox"/>		Éditer	Copier	Supprimer	110	2022-03-14 14:57:53	NULL	Développeur junior	10
<input type="checkbox"/>		Éditer	Copier	Supprimer	111	2022-03-14 17:29:39	NULL	Développeur web junior	3

Dans la table " *applicant* " les trois candidats appartiennent bien au recrutement n°111 et leur statut est passé de 4 à 0 c'est-à-dire entretien à programmer (de status = STATUS_PENDING à status = INTERVIEW_TO_SCHEDULE).

id	firstname	lastname	number	mail	created_at	cv	letter	profile	comments	updated_at	archived_at	status	recruitment_id
1466	Hélène	Cohen	0102030405	barthelemy.alfred@example.net	2021-12-30	NULL	NULL	0	Vel molestiae amet sapiente vitae et nesciunt. Ape...	2022-03-14 18:08:14	NULL	0	111
1470	David	Bertrand	0102030405	philippe02@example.org	2021-12-27	NULL	NULL	2	Quaerat cupiditate autem expedita aliquid voluptat...	2022-03-14 18:08:31	NULL	0	111
1494	Yamina	Jouille	0384731665	yamina.jouille@gmail.com	2022-03-14	NULL	NULL	2	En reconversion.	2022-03-14 18:08:24	NULL	0	111

Nous pouvons donc commencer les entretiens en utilisant le tableau de bord du recrutement sélectionné. Voici un exemple de tableau de bord d'un recrutement en cours :

LISTES DES CANDIDATS
RECRUTEMENTS EN COURS
RECRUTEMENTS TERMINÉS

RECRUTEMENT DÉMARRÉ LE : 15/03/2022

Poste à pourvoir : Développeur junior

Rechercher:

CANDIDATS	ENTRETIEN N° 1	ENTRETIEN N° 2	ACTION
AUBERT Mathilde	15/03/2022	15/03/2022	>
THOMAS Marcelle	15/03/2022	15/03/2022	>
MULLER Gilles	15/03/2022	15/03/2022	>
BENOIT Agnès	15/03/2022 à 11:03		
PEREZ Laetitia	15/03/2022		
LECOMTE Bernadette	15/03/2022		>
MARIE Honoré	15/03/2022		>
LUCAS Marine	15/03/2022		>
ROCHER Monique	15/03/2022		>
POTTIER Marc	15/03/2022		>

Affichage de 1 à 10 sur 10 éléments

Ajouter un candidat au recrutement : AJOUTER

TERMINER LE RECRUTEMENT



Ce tableau de bord a été généré grâce à la bibliothèque *Faker* que j'ai utilisée tout au long du développement de mon projet, la bibliothèque *Faker* génère des jeux de fausses données (*fixtures*). Pour que ce jeu de données soit cohérent, il fallait que ça corresponde au modèle de donnée. Voici un extrait de ma classe `DataFixtures.php`

```
// Fixtures for current recruitments
for ($i = 0; $i < 2; $i++) {
    $recruitment = new Recruitment();
    ($i == 0) && $recruitment->setJobTitle('Lead développeur');
    ($i == 1) && $recruitment->setJobTitle('Développeur junior');
    $recruitment->setNbApplicants(10);
    $manager->persist($recruitment);

    for ($j = 0; $j < 10; $j++) {
        $applicant = new Applicant();

        $applicant->setFirstName($faker->firstName)
            ->setLastName($faker->lastName)
            ->setCreatedAt($faker->dateTimeBetween('-5 week', '-3 week'))
            ->setMail($faker->safeEmail)
            ->setNumber("0102030405")
            ->setProfile(mt_rand(0, 2))
            ->setStatus(mt_rand(0, 3))
            ->setComments($faker->text())
            ->setRecruitment($recruitment);
        $manager->persist($applicant);

        for ($k = 1; $k <= mt_rand(1, 3); $k++) {
            $interview = new Interview($applicant, $recruitment, $faker->dateTimeBetween('-3 week', '-1 week'));
            $interview->setModality(mt_rand(0, 2))
                ->setNotes($faker->text())
                ->setTechnique($faker->text())
                ->setImpressions($faker->text());
            $manager->persist($interview);
        }
    }
}
```

La commande suivante génère ce jeu de données.

```
yaminajouille@PORT201910-098:~/recrutator$ bin/console doctrine:fixtures:load
```

Une fois que tous les entretiens se sont déroulés, il est possible de clôturer la session de recrutement. Si au moins un des candidats restant dans la liste n'a pas le statut retenu (`status = STATUS_APPLICANT_KEPT`), un message d'avertissement l'indique.

Terminer ce recrutement



Impossible de terminer le recrutement. Il y a des candidats dans la liste qui ne sont pas retenus.

FERMER

Terminer le recrutement



Pour chaque candidat retenu, veuillez renseigner la durée et le type de contrat proposé.

LECOMTE Bernadette

Contrat AFPR - CDD 06 mois

AUBERT Mathilde

Contrat apprentissage - CDD 12 mois

THOMAS Marcelle

Contrat AFPR - CDD 12 mois

FERMER

VALIDER

Si tous les candidats sont retenus, le recrutement peut-être clôturé.



7. Description de la veille

J'ai fait de la veille technologique en rapport avec la sécurité d'une application Symfony. En particulier, l'authentification c'est-à-dire vérifier qui on est, et si on peut le prouver, ainsi que l'autorisation c'est-à-dire si on a les droits d'accès à une ressource. J'ai surtout consulté la documentation officielle sur ce point :

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

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

Pour mettre en place un système d'authentification sur Symfony, on utilise le composant *Security*. On crée d'abord une entité *User* qui contient un ID, un nom d'utilisateur et un mot de passe. Cette entité implémente l'interface *UserInterface* du composant *security* ainsi que l'interface *serializable*. Une partie de la configuration se fait dans le fichier *security.yaml*. Le mot de passe est chiffré grâce à *bcrypt*, une fonction de hachage de mot de passe, avec un cost de 12.

Recrutator est un outil de gestion des recrutements internes à l'entreprise, les utilisateurs ont le rôle d'administrateur. Cet outil sera accessible depuis le site vitrine de l'entreprise aux seuls administrateurs. Il faut donc sécuriser l'accès.

Ce module a été réalisé par le stagiaire m'ayant précédé sur ce projet mais je l'avais déjà développé de manière similaire pour un projet Symfony au cours de la formation. Cela comprend :

- Une page de connexion avec login et mot de passe avec gestion des erreurs (identifiants invalides),
- Une page d'enregistrement d'un administrateur.

Un autre aspect de la sécurité est la protection des applications des attaques XSS.

Dans mon projet, j'ai appliqué ce concept en mettant en place le plus possible de validation du côté serveur. Symfony permet cela en ajoutant des *asserts* au niveau des entités.

En plus de la documentation officielle (<https://symfony.com/doc/current/reference/constraints.html>), j'ai consulté des tutoriels vidéo, je citerai celle de grafikart par exemple.

8. Description d'une situation ayant nécessité une recherche anglophone

Durant ma formation et mon stage, j'ai consulté beaucoup de documentation officielle (*Symfony*, *Twig*, documentation des différentes librairies utilisées, etc.). Ces sites sont anglophones. Il en est de même pour les sites d'aide (Stackoverflow par exemple). Il m'a fallu m'adapter et utiliser ces sites en anglais et ne pas me contenter des sites en français.

Pour Recrutator, j'ai utilisé beaucoup de tableaux pour l'affichage des informations (listes des candidats, des recrutements, etc.).

J'ai opté pour l'utilisation de la librairie *datatable* de *Bootstrap*. Une simple recherche Google "*datatable*" me permet d'obtenir la documentation officielle et de choisir sur le site la version qui m'intéresse, celle de Bootstrap5 (<https://datatables.net/>).

En fonction des options que je voulais utiliser, j'ai fait plusieurs recherches sur le site.

Textes en anglais

Row selection (multiple rows)

It can be useful to provide the user with the option to select rows in a DataTable. This can be done by using a click event to add / remove a class on the table rows. The `rows().data()` method can then be used to get the data for the selected rows. In this case it is simply counting the number of selected rows, but much more complex interactions can easily be developed.

If you are looking for a more complete and easier to use row selection option, the Select extension provides an API that is fully integrated with DataTables for selecting rows and acting upon that selection.

Textes en français

Sélection de lignes (sélection multiple)

Il peut être utile de mettre à disposition de l'utilisateur l'option de sélectionner plusieurs lignes dans la table de données. Ceci est possible en utilisant un événement click (event) pour ajouter / supprimer une classe sur les lignes de la table. La méthode `rows().data()` peut ensuite être utilisée pour obtenir les données des lignes utilisées. Dans ce cas, c'est simplement le fait de compter le nombre de lignes sélectionnées mais beaucoup plus d'interactions plus complexes peuvent être développées.

Si vous recherchez une option de sélection de lignes plus complète et plus facile, l'extension *Select* fournit une API entièrement intégrée à DataTables pour sélectionner des lignes et agir sur ces dernières.



9. Conclusion

Le projet qui m'a été confié à couvert beaucoup d'aspects du développement d'une application web dynamique. Ça a été pour moi un projet complet, très enrichissant. J'ai pu approfondir mes connaissances du framework *Symfony* et son utilisation dans un contexte professionnel. J'ai également travaillé sur la partie front-end en consolidant mes acquis en *SCSS*, *JavaScript*, *jQuery* et *Bootstrap*.

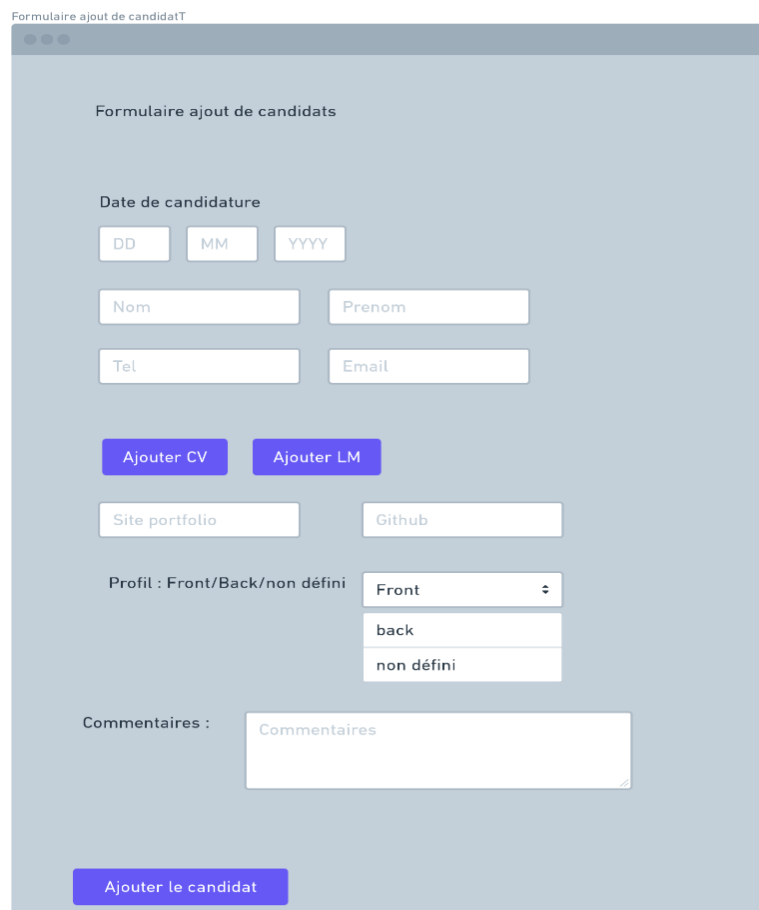
Symfony est un framework puissant mais complexe. Son utilisation pour des besoins spécifiques d'un projet a nécessité, dans mon cas, beaucoup de recherches par exemple pour l'utilisation de l'inclusion des contrôleurs dans des vues, l'utilisation des modales entre autres.

J'ai listé les améliorations à apporter à Recrutator que je vais pouvoir mettre en œuvre car l'aventure continue avec l'entreprise Débutant(e) accepté(e) au mois de mai 2022 !

10. Annexes

A. Maquettes

Voici quelques-unes des interfaces que j'ai réalisées au début du projet. Ces visuels servaient de supports pour comprendre les besoins de l'entreprise mais aussi pour en exprimer d'autres.



Formulaire ajout de candidats

Date de candidature

DD MM YYYY

Nom Prenom

Tel Email

Ajouter CV Ajouter LM

Site portfolio Github

Profil : Front/Back/non défini

Front
back
non défini

Commentaires : Commentaires

Ajouter le candidat

Figure 1 Formulaire d'ajout d'un candidat



Liste des candidats

[Candidats en attente](#) [Candidats recrutés](#) [Candidats archivés](#)

[Ajouter candidat](#)

Liste des candidats

Nom	Candidat ...	Voir	Editer	Archiver
DUPOND Jacques	... depuis le 09/10/2021			
MARTIN Odile	... depuis le 25/10/2021			
DUBOIS Pierre	... depuis le 02/11/2021			
BLANC Marcelle	... depuis le 04/11/2021			
MERCIER Gilles	... depuis le 01/11/2021			
DURAND Magalie	... depuis le 10/10/2021			
LATUILE Jean-Mi	... depuis le 05/10/2021			

[Démarrer un recrutement](#)

Figure 2 Listes des candidats

Popup démarrer recrutement

Intitulé du type de poste

[Valider](#) [Annuler](#)

Figure 3 Popup démarrer un recrutement

Recrutement démarrée le : 01/12/2021

Nom	Etape1	Etape2	Etape 3	Ajouter etape
DUPOND Jacques	☎ 10/12/21	☎ 15/12/21	🕒	+ recruter retenir archiver
MARTIN Odile	☎ 09/12/21	👤 12/12/21	💻 19/01/22 9h30	+ recruter retenir archiver
BLANC Marcelle	💻 10/12/21	👤 13/12/21	☎ 20/12/21	+ recruter retenir archiver
MERCIER Gilles	☎ 08/12/21	👤 10/12/21	👤 24/12/21	+ recruter retenir archiver
LATUILE Jean-Mi	🕒			+ recruter retenir archiver

Terminer recrutement

DUBOIS Pierre ▾

Ajouter un candidat

DURAND Magalie

Figure 4 Tableau de bord d'un recrutement

Popup terminer recrutement

Terminer le recrutement

Pour chaque candidat retenu, renseigner la durée et le type de contrat proposé. (exemple : CDI avec proposition de sociétaire immédiat. AFPR+CDD 12 mois)

DUPOND Jacques :

MARTIN Odile :

BLANC Marcelle

Valider

Annuler

Figure 5 Popup terminer un recrutement



Fiche Candidat

Nom Prenom

Candidat depuis le

Tel :

Email :

Voir le CV

Voir la LM

Site portfolio :

Github :

Profil : Front/Back/non défini

Front

back

non défini

Réseaux sociaux :

Commentaires :

Ponam in culpa idiota alii pravitatis. Principium ponere culpam in se justum praeceptum. Neque impropere et alii qui non perfecte ipse docuit.

Suivi des entretiens en bas!!

10/12/21

Notes prises pendant l'entretien ...

20/12/21

Notes prises pendant l'entretien ...

05/01/21

Notes prises pendant l'entretien ...

Nouvelle étape

Type entretien :

Visio

Telephonique

présentiel

Motivation

Commentaire

Technique

Impressions générales

Sauvegarder

Recandidater

Archiver

Retenir

Figure 6 Vue du candidat

B. Vue de l'application sur tablette et mobile (mode paysage)

Recrutator est un outil interne, le besoin que le l'outil soit adaptable aux différents formats d'écrans n'était pas une priorité mais j'ai quand fait en sorte que l'outil soit adaptable sur tablette et sur mobile en mode paysage. La majorité des interfaces comportent des tables de données plus lisible avec ce mode d'affichage. (Exemple : Samsung Galaxy S8).

Débutant-e AccepT MENU

CANDIDATS EN ATTENTE +

Candidats en attente Candidats recrutés Candidats archivés

Rechercher:

↑↓ NOM PRÉNOM ↑↓ CANDIDAT DEPUIS LE :

<input type="checkbox"/>	CHEVALLIER Frédérique	01/01/2022			
--------------------------	-----------------------	------------	--	--	--

Débutant-e AccepT MENU

LISTES DES CANDIDATS
RECRUTEMENTS EN COURS
RECRUTEMENTS TERMINÉS

PARIS Gilles	04/03/2022	13/03/2022 à 00:03
LAMY Jacques	22/02/2022	20/03/2022 à 00:03
GALLET Anouk	28/02/2022	04/03/2022 25/03/2022 à 00:03



C. Arborescence du projet

Voici l'architecture de mon projet. J'ai respecté le plus possible les conventions de nommage (CamelCase, snake_case, kebab-case, etc) en fonction du type des fichiers. J'ai utilisé des noms de fichiers et variables significatifs et en anglais.

Le code est aussi documenté le plus possible en anglais.

