RAPPORT DE STAGE

Développeur Web Junior

SOMMAIRE

I. Contexte

1. Présentation personnelle

Avant de commencer la présentation de mon stage, des missions qui m'ont été confié et du bilan que j'ai pu en faire. Je souhaiterais commencer mon rapport de stage par une présentation me concernant.

Je m'appelle Abdenour BELKACEMI et j'ai 31 ans. Après une expérience de 12 ans dans le secteur social, où j'ai occupé des postes d'éducateur dans diverses structures avant d'occuper un poste de directeur de structure associative où j'avais comme mission la gestion des nouveaux projets et des projets présents, les ressources humaines, la relation avec les divers partenaires et financeurs (Préfecture, Mairie, etc.) j'ai fait le choix d'une orientation professionnelle. Passionné depuis toujours par la technologie (encore plus les nouvelles technologies) et l'univers du Web et de l'informatique, j'ai fait le choix de m'orienter vers le métier de développeur Web.

J'ai commencé la formation de Développeur Web Junior sur OpenClassRoom au mois d'octobre 2019. Durant cette formation et les divers projets rendus, j'ai appris plusieurs langages : HTML, CSS , JavaScript, PHP, SQL et plusieurs notions : algorithme, programmation orientée objet, pattern MVC, les API et Web Service, etc. J'avais pour motivation, après avoir finalisé mon projet 4 (construit à partir de PHP et SQL) d'aller plus loin dans mon apprentissage et de m'initier aux Frameworks.

C'est dans cette perspective que j'ai choisi d'effectuer un stage en lieu et place d'un projet 5 personnel afin de finaliser ma formation. Ce stage me permettrait une immersion en entreprise avec les exigences et les contraintes qui y sont liées. Mais aussi d'avoir une première expérience au sein d'une équipe. J'ai donc cherché un stage me permettant cela et qui me permettrait de m'initier aux Frameworks afin de monter en compétences.

Après plusieurs recherches, j'ai trouvé un lieu de stage utilisant un outil développé avec Symfony 4 et Angular 7 demandant de nouvelles fonctionnalités.

C'est dans ce cadre de fin de formation et d'une envie de monter en compétence que j'ai choisi ce stage. Par intérêt personnel, mais aussi parce que la connaissance de Framework intéresse nombre d'entreprises, ce lieu de stage me permettait de m'initier à Symfony et à Angular, ce qui correspond à mon envie de monter en compétence.

2. Présentation du lieu de stage

L'entreprise se nomme ***** et se situe dans la ville de Levallois-Perret en Ile de France. C'est une petite entreprise, ouverte en 2015, constituée actuellement de deux personnes. L'activité principale de l'entreprise ****** est la maintenance réseau et les deux collègues sont diplômés technicien réseaux informatiques. Parallèlement à la maintenance des ordinateurs et du réseau, qu'ils effectuent auprès de diverses entreprises, ****** créait depuis deux ans des sites Web pour différents clients à partir du CMS Wordpress.

Les deux salariés (dont un est le directeur de l'entreprise) se relayent pour la gestion des parcs informatiques, des clients et la création des sites Web.

Pour la gestion des parcs informatique, ils utilisent une application Web, appelée SimpleIT, développée en interne par un ancien développeur web, auparavant dans l'équipe. Cette application Web, en Single Page Application, permet la gestion de parcs informatique et la gestion des interventions auprès de clients (inventaire des modèles d'ordinateurs et du Software pour un client donné) ainsi que la gestion des interventions (création d'une intervention, mise à jour et résolution d'une intervention).

Parallèlement à cet outil, l'entreprise utilise divers outils pour répondre à différents besoins non inclus dans SimpleIT. Nous comptons :

- Un CRM : EspoCRM qui permet la gestion des relations avec les clients ;
- Outlook : permettant l'envoi de mail et la gestion de planning ;
- Excel : permettant de reporter l'ensemble des interventions basées sur l'application SimpleIT afin d'établir les factures.

La multiplication des outils utilisés et l'équipe en nombre restreint créaient une déperdition de l'information qui se répercute sur la qualité de gestion, de relation avec les clients, mais aussi sur l'exactitude des interventions menées et donc facturées.

II. Problématique

Comment améliorer la qualité de gestion en réduisant le nombre d'outils utilisés ?

III. Missions confiées

1. Objectif

Après réflexion en équipe sur la stratégie la plus adaptée afin de répondre à la problématique initiale, il a été décidé de reprendre l'application SimpleIT afin d'ajouter une fonctionnalité de « Reporting ».

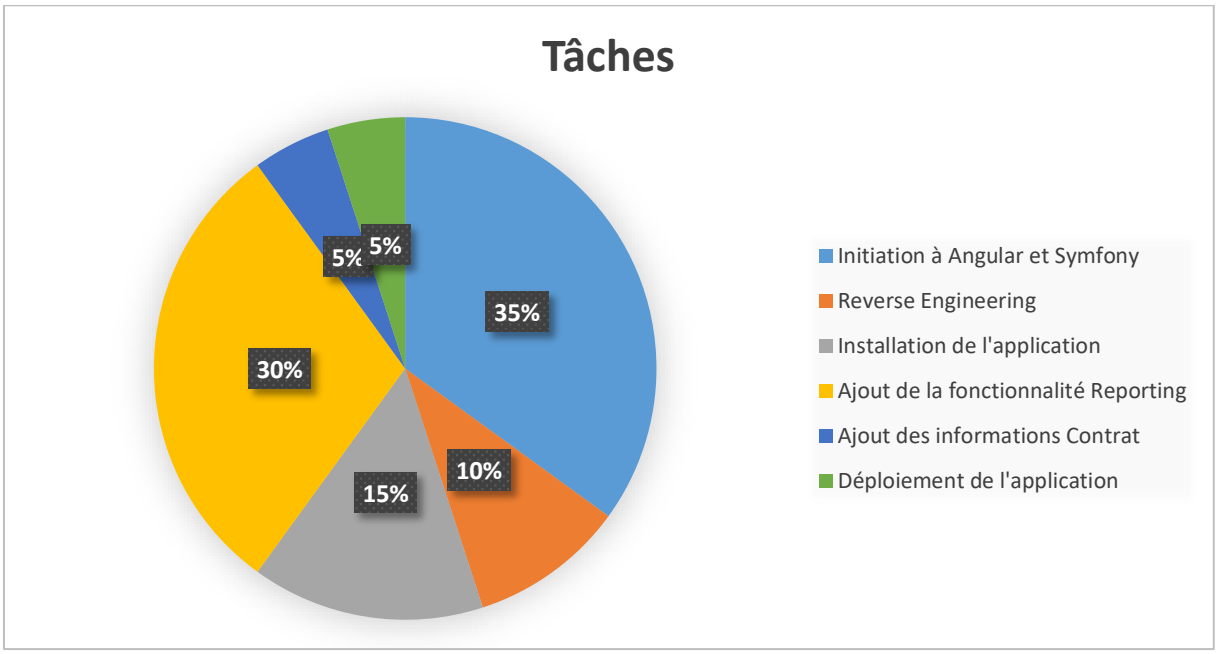
Cette fonctionnalité a été demandée par le directeur d'***** afin de pouvoir récupérer automatiquement les interventions enregistrées en base de données pour un client donné et de pouvoir en faire un export PDF avec la possibilité de choisir les interventions sur une période donnée.

Cela permettra à l'entreprise de ne plus utiliser l'outil Excel, de centraliser le Reporting (support pour la facturation) autour de l'application SimpleIT et d'être au plus près des interventions effectuée.

Pour arriver à cet objectif, il m'a fallu suivre plusieurs étapes indispensables à la bonne réalisation de mon objectif. Étapes que je vais détailler ci-dessous.

2. Répartition des tâches

Afin d'avoir une vue globale des différentes étapes que j'ai dû suivre pour arriver à réaliser mon objectif, je propose un diagramme qui montre l'ensemble des tâches initiales et le temps que cela m'a pris (en pourcentage). Cette répartition initiale a été modifiée en cours de stage. Je le détaillerais par la suite.



3. Réalisation de mes objectifs

a. Initiation aux Framework Symfony 4 et Angular 7

Reprenant un code complexe d'un développeur n'étant plus dans l'équipe et n'ayant pas de développeur dans l'équipe pouvant m'aider dans le fonctionnement de Symfony et Angular, j'ai dû consacrer une partie importante de mon temps dans l'initiation à ces deux Frameworks. Condition *sine qua non* pour atteindre mon objectif.

Cela m'a permis une compréhension et une utilisation de Symfony et Angular, mais aussi comment elle pouvait interagir entre elles.

Le point de départ a été la compréhension d'un Framework. En suivant différents cours et tutoriels, j'en ai compris qu'un Framework est un ensemble d'outils offrant un certain cadre de travail. Ce cadre offre du code préconçu et utilisable dans son application web (ou site) Cela améliore la productivité en simplifiant le développement grâce notamment aux diverses librairies à dispositions (en natif ou à installer) permettant au développeur de ne pas partir de zéro. J'ai, par

exemple, utilisé un « bundle (ou module pour Angular)» pour transformer du HTML en PDF. Il m'a fallu maîtriser l'outil, mais cela m'a évité de coder l'ensemble de la fonctionnalité à partir de zéro.

Ils sont d'autant plus intéressants qu'ils exigent de respecter les bonnes pratiques de développement (POO, MVC, namespaces, etc.).

J'ai observé plusieurs particularités dans ces frameworks :

- Ils installent un ensemble de fichiers et dossiers prêt à l'emploi,
- On les utilise en codant de manière classique, mais aussi en ligne de commande dans la console grâce à « composer » pour Symfony et « npm » pour Angular. Ils permettent de gérer simplement les dépendances (librairies) dont le projet a besoin pour fonctionner, mais aussi la gestion de la base de données avec Doctrine ainsi que d'autres fonctionnalités.

Symfony

J'ai donc commencé par l'apprentissage de Symfony qui est un Framework PHP permettant de faire du Backend autant que du Frontend. Il permet de réaliser des sites web, applications web, API, et autres. Basée sur le pattern MVC, Symfony fonctionne autour de trois éléments :

- Doctrine (avec les Entity et les Repository) : gestion de la base de données
- Les Controllers : contenant la logique du code,
- Twig : permettant l'affichage de la vue (non utilisé dans l'application SimpleIT puisque le Frontend est géré par Angular).

Autre notion importante que j'ai apprise est la gestion de la sécurité, authentification et autorisation des visiteurs ce qui donnera accès à tout ou une partie de l'application (ou site) web selon la configuration et l'identité du visiteur.

Mon apprentissage s'est fait d'une part par les cours, mais surtout par la pratique en faisant des petites applications tests.

Symfony est riche en fonctionnalité et possibilité, il serait inapproprié d'en parler plus longuement. Je parlerai dans la suite du rapport de stage de mon utilisation pratique dans le cadre de ma mission.

Angular

J'ai poursuivi mon apprentissage avec Angular, un Framework JavaScript. Tout comme Symfony, à l'installation, nous avons des fichiers et dossiers installés.

Angular fonctionne autour des « components » qui sont les composants de base d'une application Angular. L'application étant une arborescence de plusieurs components. Créer un component permettra l'installation de trois fichiers :

- Le template HTML (.html),
- La feuille de style CSS ou SCSS (.css ou .scss),
- Le fichier TypeScript qui contiendra la logique du code (.ts)

Par un système de « routing » grâce aux différentes balises <app-root> ou par exemple <app-interventions> nous pouvons créer l'ensemble du Frontend de notre application avec un format

Single Page Application. Lors de la lecture par le navigateur, ces balises seront remplacées par leurs contenus respectifs.

J'ai appris différentes notions propres à Angular, tels que la gestion des données dynamiques entre le fichier Typescript et le template grâce au « databinding », la structure du document grâce aux directives (*ngIf, *ngFor par exemple) et aux « services », modifier dynamiquement les données grâce aux « pipes ».

Ils seraient là aussi ardus de parler de l'ensemble des possibilités offertes par Angular dans ce rapport de stage tant cela demanderait de nombreuses pages (et approfondir mes connaissances).

Comme Symfony, mon apprentissage s'est déroulé avec des cours et des applications tests.

b. Reverse Engineering

À mon arrivée sur le lieu de stage, le directeur m'a transmis le code source et un export de la base de données.

Il m'a fallu prendre le temps de comprendre le fonctionnement global de l'application. Pour cela j'ai utilisé l'application actuelle, pour comprendre les fonctionnalités et commencer le « reverse engineering ». Partant d'une application existante, j'ai dû en comprendre le fonctionnement et sa construction.

Pour cela j'ai répertorié les différents bundles présents dans Symfony et dans Angular. Afin d'avoir un aperçu des fonctionnalités, mais aussi de la construction de l'application Web. Cela m'a permis de comprendre que le Backend, fait avec Symfony, est une API REST avec la présence des bundles « FOSRestBundle » et « Nelmio API ». L'authentification et les autorisations selon le rôle du visiteur sont gérées par un système de « token » avec le bundle LexikJWTAuthenticationBundle.

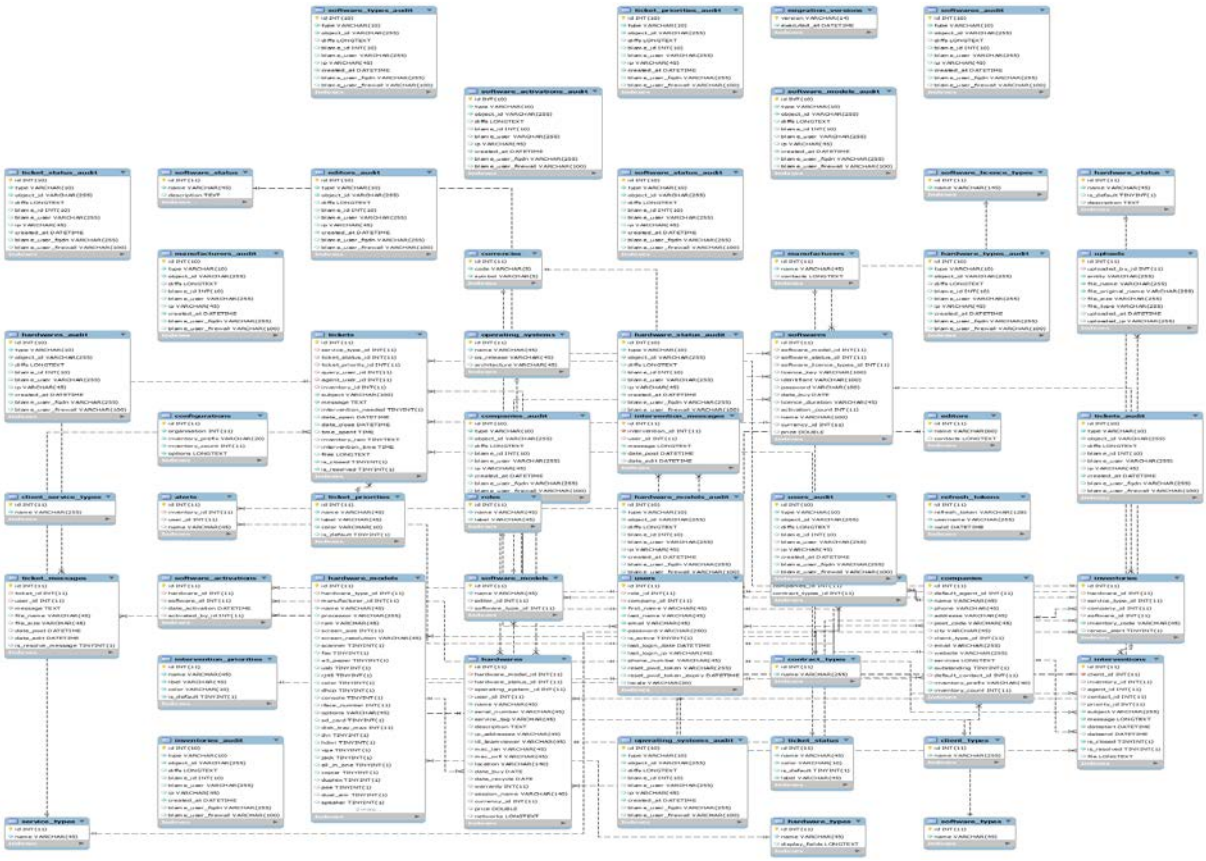
N'étant pas familier des API REST, je me suis renseigné sur leur fonctionnement. Basée en majorité sur le protocole HTTP et sur les verbes d'action, une API REST est un style architectural qui va permettre de centraliser des services et d'interagir avec eux grâce à des verbes HTTP « GET, POST, PUT, DELETE » qui reprennent les éléments d'un CRUD : create, read, update, delete.

Dans cette continuité, j'ai appris à utiliser POSTMAN, qui est un logiciel permettant de tester différentes requêtes vers une API REST.

Dans cette phase de compréhension de l'application, j'ai relevé qu'Angular communique avec l'API REST sous Symfony grâce à des requêtes et récupère ou modifie les informations en base de données. Il se charge ensuite d'afficher sur le navigateur web du client.

Pour compléter cette partie, j'ai continué la compréhension de l'application par la base de données en le schématisant grâce au logiciel MySQL WorkBench. J'ai pu avoir la base de données et les différentes relations entre les tables.

Je mets ci-dessous une représentation, sous forme de diagramme relationnel, de la base de donnée que j'ai obtenue avec le logiciel.



c. Installation de l'application

Cette étape m'a permis de comprendre l'importance de l'environnement de travail et qu'une installation d'un Backend sous Symfony et d'un Frontend sous Angular nécessite de définir certaines variables d'environnements.

Je vais détailler ci-dessous le processus que j'ai suivi et les problèmes que j'ai rencontrés afin d'installer l'application :

- Vérification et adaptation de l'environnement de travail pour correspondre à celui du code source :
  - Utilisation de **Symfony version 4.3.2**
  - Utilisation de **Symfony CLI version 4.16.0** pour la partie Backend (partie API)
  - Mettre l'**environnement** en **« dev »** et non en « prod » dans le .env
  - Configurer les bonnes informations de **connexion à la BDD** dans le **.env**

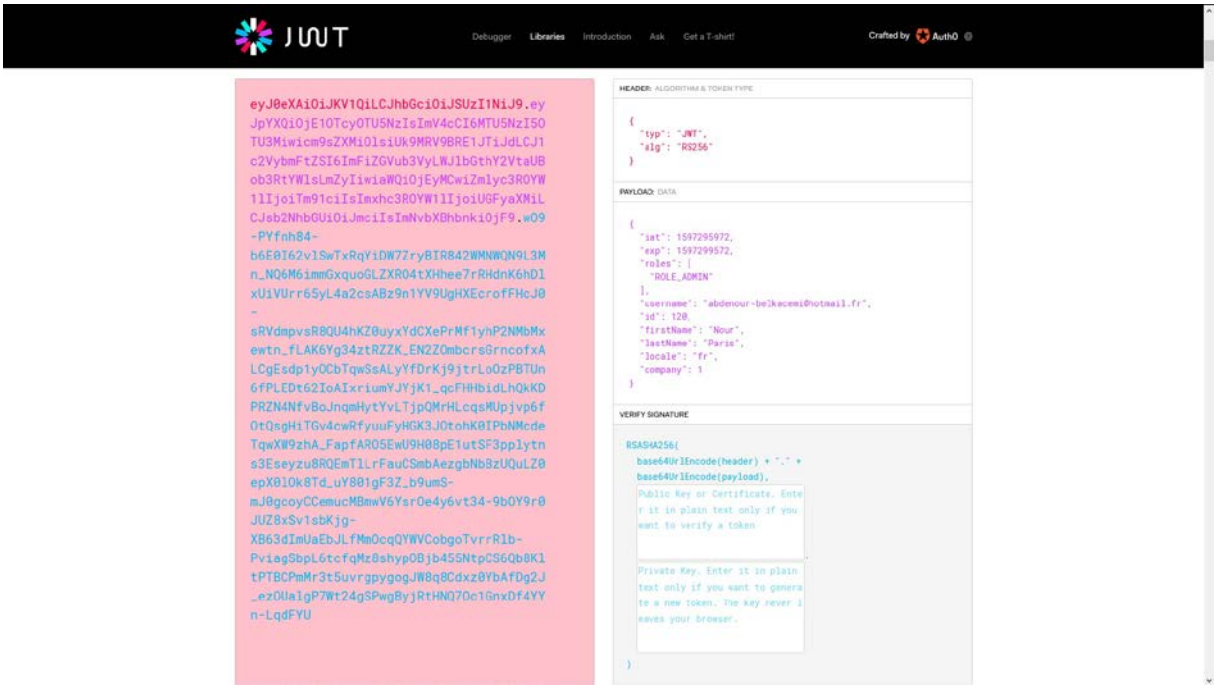
    "DATABASE_URL=mysql://Identifiant:Password@IP:Port/DBName"
  - Utiliser une version **PHP version 7.3**
  - **Supprimer dossier Vendor**
  - Lancer la commande : **composer install –optimize-autoload**

- Installer le **Frontend ANGULAR** :
    - Vérification et adaptation de mon environnement de travail :
        - **NPM version 6.14.5**
        - **Angular CLI version 7.3.9** pour la partie Frontend
        - **Angular version 7.2.15**
        - Le **Frontend** communique avec **l'API Backend** via des **requêtes AJAX**
    - Configuration de la communication avec **l'API Backend** :
        - Configurer la bonne route pour envoyer la requête AJAX : dans le fichier « environments.ts » / ligne 44 apiUrl: 'https://127.0.0.1:8000/api', (à modifier selon son URL Localhost).
- Dans la base de données :
    - Création d'un utilisateur avec droit d'admin (mettre dans le champ « is-active » la valeur 1 signifiant que le compte est actif. Dans « rôle » la valeur 4 pour avoir les droits d'admin

d. Authentification et autorisation

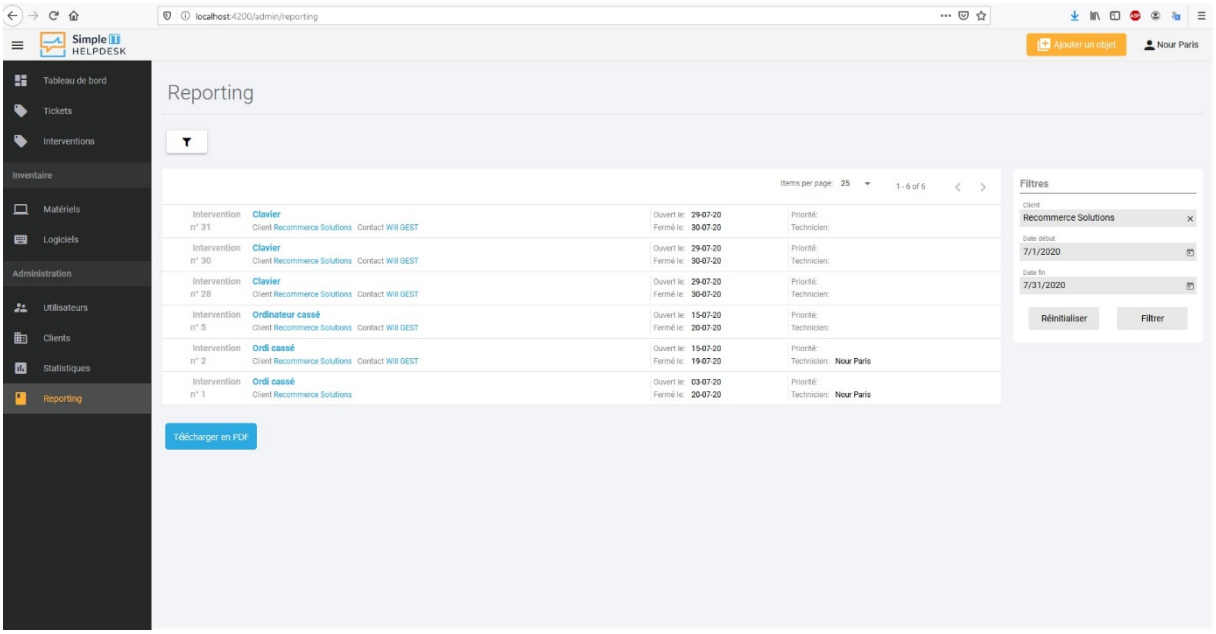
L'authentification et les différents niveaux d'autorisations sont gérés par le bundle LexikJWTbundle.

Elle fonctionne par un système de jeton JWT qui sera émis à chaque requête d'authentification faite auprès de l'API. Ici, avec la ressource dans l'URL : api/login_check. Ce jeton contient plusieurs informations : l'en-tête, qui contient l'algorithme utilisé pour la signature du jeton, la charge utile du jeton qui va contenir nos informations utilisateurs et le rôle de l'utilisateur et pour finir, la signature du jeton qui va le sécuriser. Ce jeton sera validé ou non par le serveur avec une clé secrète. Le serveur décodera par la suite le contenu JSON du jeton, notamment le « ROLE » ce qui va permettre de signaler dans le code les accès selon le rôle de l'utilisateur (dans le Controller).



e. Ajout de la fonctionnalité Reporting

Une fois l'environnement de travail configuré, j'ai pu commencer l'ajout de la fonctionnalité. Avant de détailler le code permettant la construction de cette fonctionnalité, j'aimerais commencer par une capture d'écran du résultat final et à partir de ça détailler les attentes de l'entreprise.



Ce qui était attendu :

- Ajouter une nouvelle section « Reporting » à l'application web tout en conservant la forme d'une Single Page Application ;
- Dans cette section, retrouver :
    - Une liste des interventions;
    - Un formulaire de filtrage selon la compagnie et la plage de temps voulu pour les interventions;
    - Un bouton permettant l'export PDF

Je détaillerai chacun des points ci-dessus dans la suite de ce chapitre.

Ajout de la section « Reporting » en format Single Page Application

- Ajout dans le « component « menu » et le fichier « menu.service.ts », qui permet la configuration du menu, une nouvelle entrée qui sera un sous-item de la section Administration. Cette entrée sera interdite au « ROLE_USER »:

```
children: [
      {state: 'users', name: 'Users', icon: 'supervisor_account'} as ChildrenItems,
      {state: 'clients', name: 'Clients', icon: 'business', restrictedRole: 'ROLE_MANAGER'} as ChildrenItems,
      {state: 'statistics', name: 'Statistics', icon: 'assessment', restrictedRole: 'ROLE_MANAGER'} as ChildrenItems,
      {state: 'reporting', name: 'Reporting', icon: 'book', restrictedRole: 'ROLE_MANAGER, ROLE_ADMIN'} as ChildrenItems,
    ]
```

- Création du component Reporting

En ligne de commande à l'aide d'Angular CLI

```
ng generate component reporting
```

Cette commande permet la création des fichiers reporting.component.html, reporting.component.scss et reporting.component.ts sur lesquelles je travaillerai afin d'avoir le résultat attendu.

- Import de la nouvelle classe ReportingComponent dans :
  - Admin.module.ts :

```
import { ReportingComponent } from '../interventions/reporting/reporting.component';
```

Et sa déclaration :

```
declarations: [
    UsersComponent,
    ClientsComponent,
    StatisticsComponent,
    ReportingComponent,
  ],
```

Ce qui me permet par la suite de configurer le « routing » en faisant l'ajout du path « reporting » dans « admin.routing.ts » et de la déclaration de class

```
{
   path: 'reporting',
   component: ReportingComponent
}
```

Cette configuration du « routing » me permet d'avoir une nouvelle entrée dans le menu avec une forme Single Page Application

API REST via Symfony

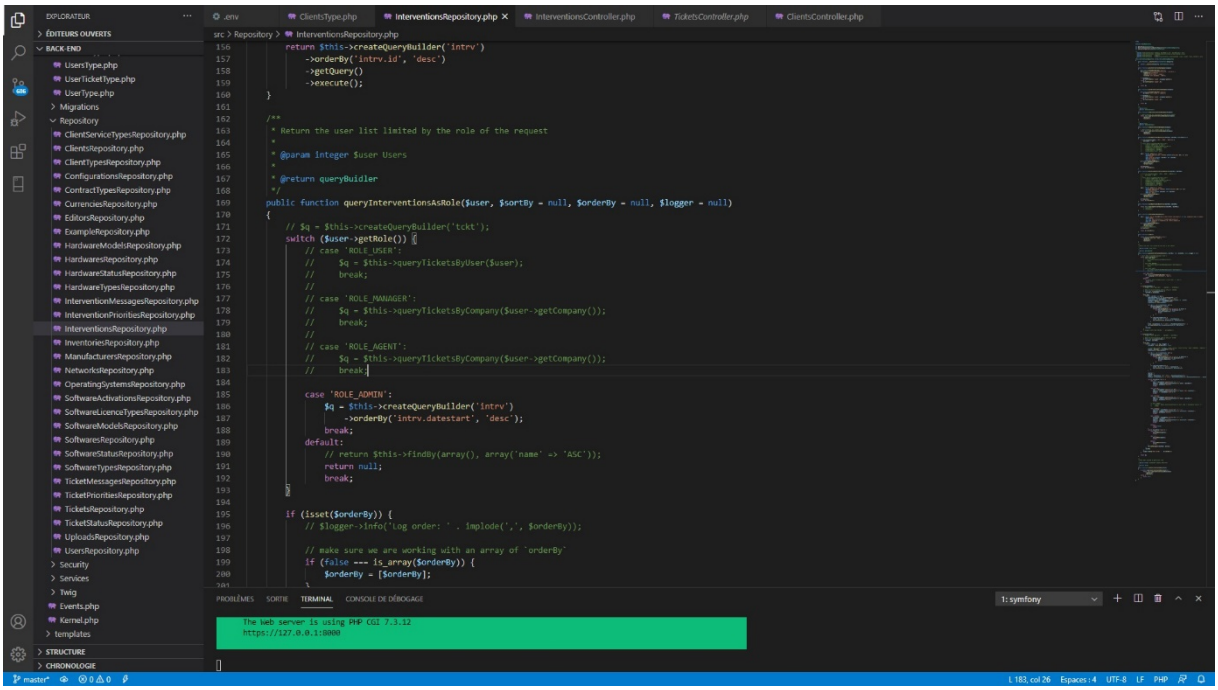
Une fois la configuration de la nouvelle section sur Angular faite, il me faut opérer plusieurs ajouts dans l'API REST sur Symfony afin d'avoir les interventions ainsi que la possibilité de sélectionner une plage de temps concernant les interventions.

Configuration du backend API pour créer une nouvelle ressource accessible via requête avec Swagger, HttpFoundation, FOSRESTbundle

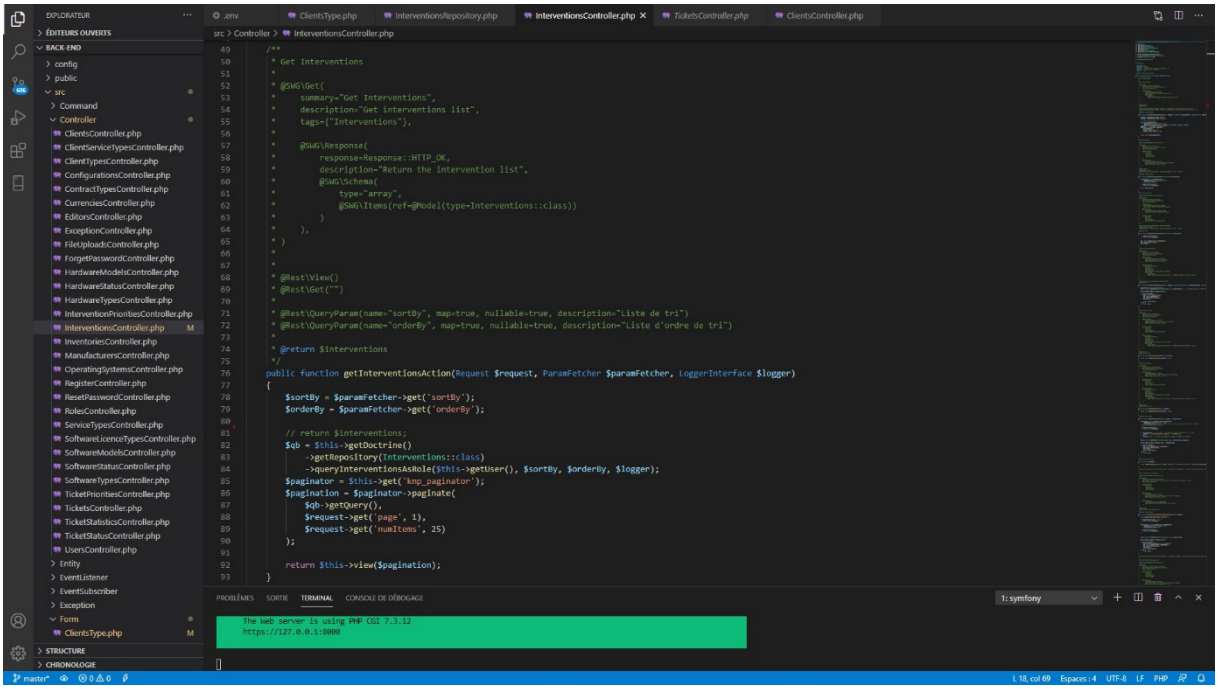
- Récupérer l'ensemble des interventions selon le code déjà établi par l'ancien développeur

- Création d'une nouvelle ressource : ensemble des interventions par compagnies :

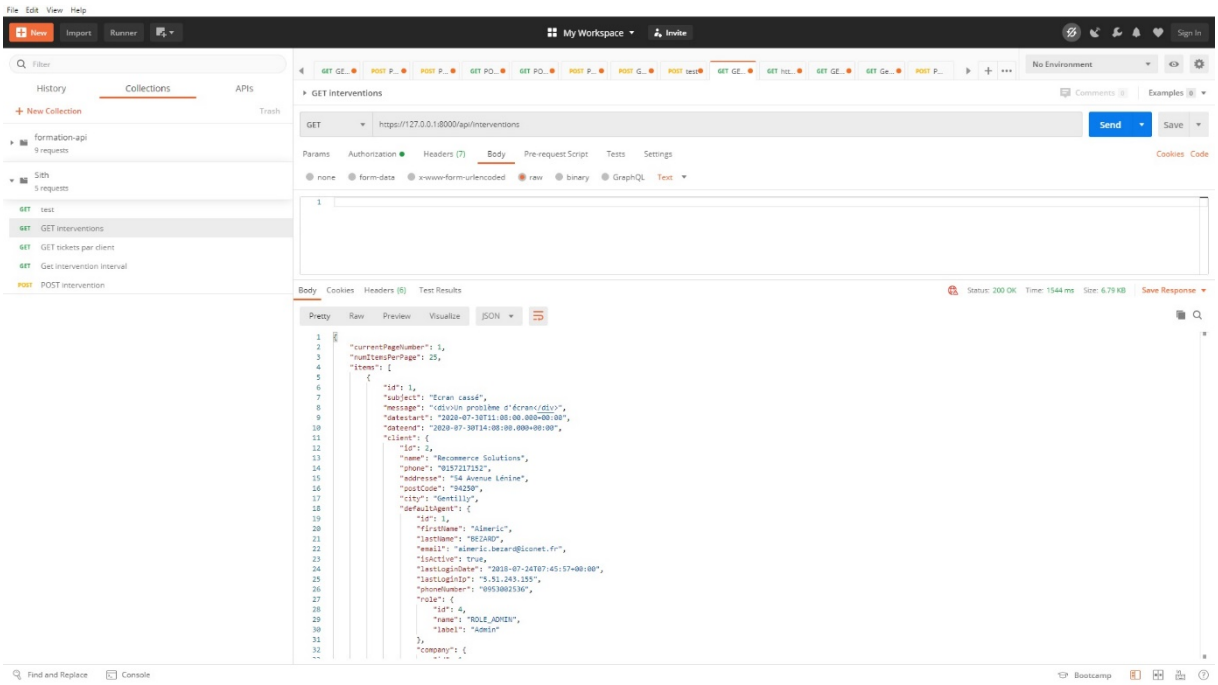
- Dans les Repository, InterventionRepository, crée deux méthodes afin de lié les interventions à l'id d'une compagnie



- Dans le Controller « InterventionController » créez une nouvelle méthode qui va permettre de configurer une nouvelle route accessible via requête (et créer la doc) et appelle de la méthode du Repository « queryInterventionsAsRole() » pour avoir la liste d'intervention.

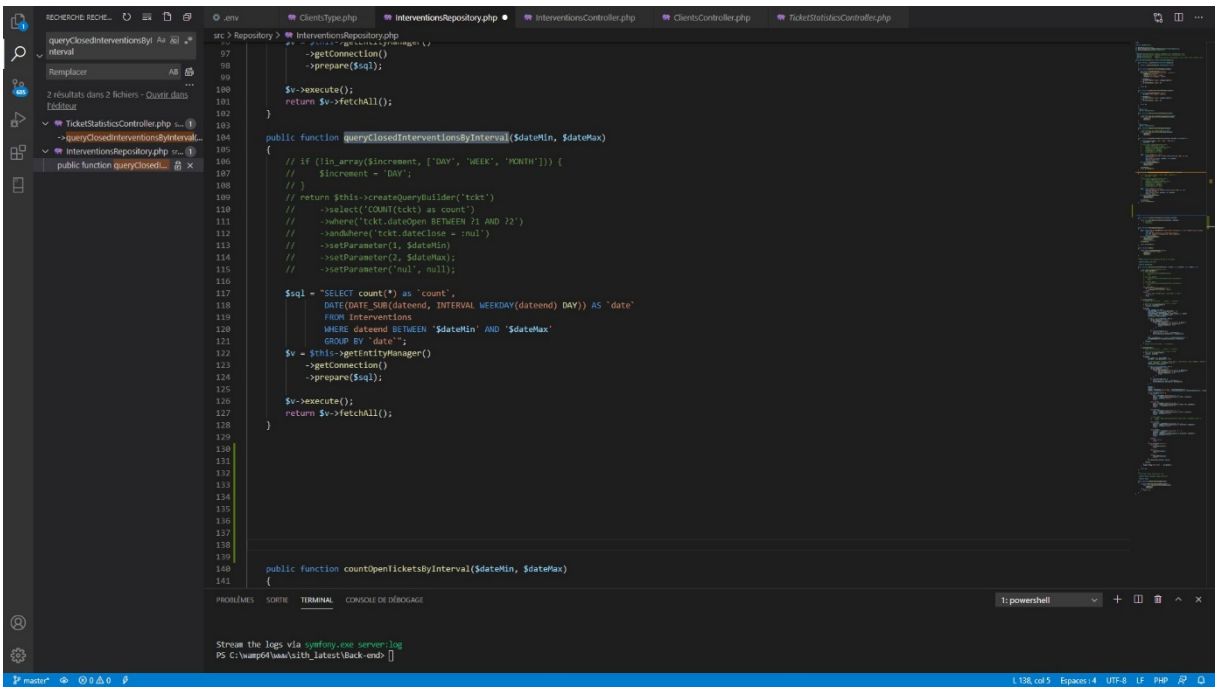


- Test de la requête avec le verbe HTTP GET (pour récupérer en format JSON) sur PostMan :

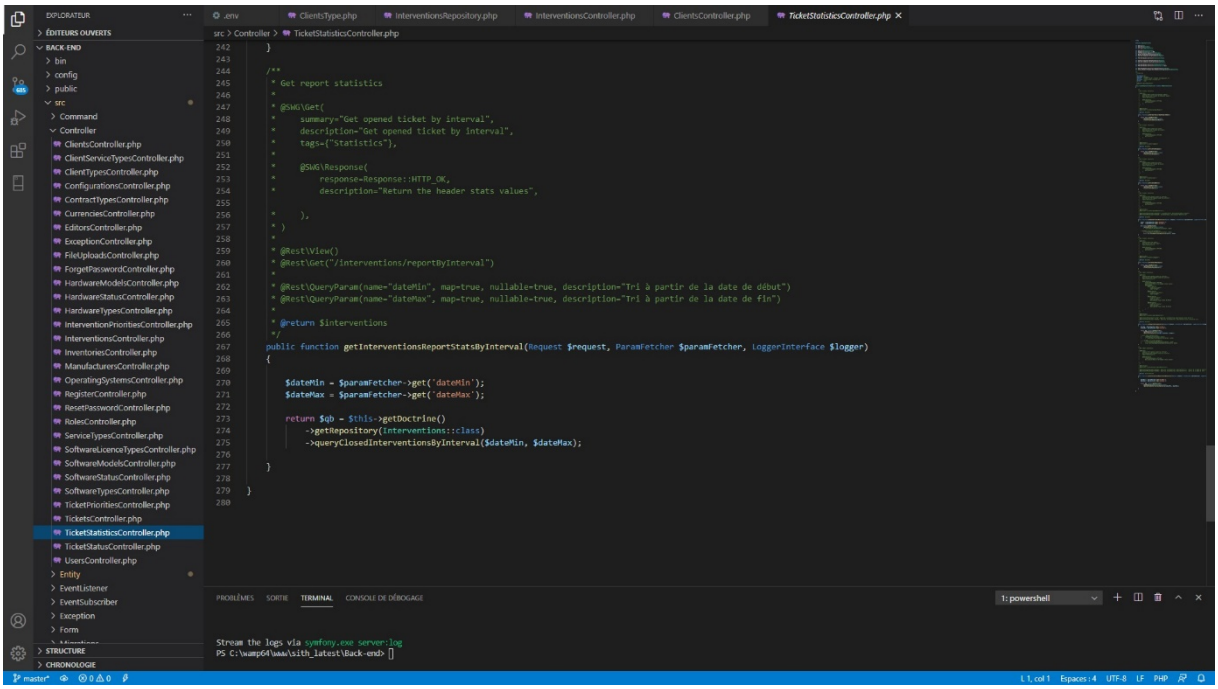


Création d'une ressource permettant le choix d'une plage de temps pour les interventions

- Création d'une méthode dans le Repository permettant la sélection dans la base de données

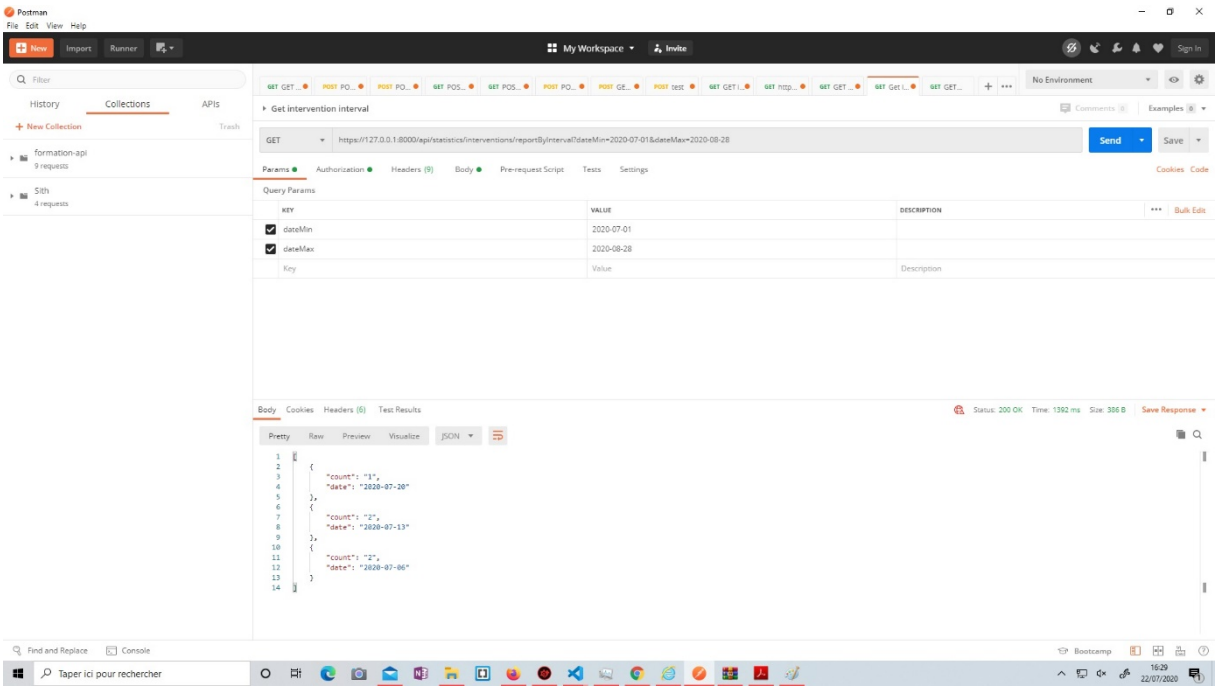


- Configuration d'une route accessible via l'API REST avec le système d'annotation



```
    }

    /**
     * Get report statistics
     *
     * @SWG\Get(
     *     summary="Get opened ticket by interval",
     *     description="Get opened ticket by interval",
     *     tags={"Statistics"},
     *
     *     @SWG\Response(
     *         response=Response::HTTP_OK,
     *         description="Return the header stats values",
     *
     *     ),
     * )
     *
     * @Rest\View()
     * @Rest\Get("/interventions/reportByInterval")
     *
     * @Rest\QueryParam(name="dateMin", map=true, nullable=true, description="Tri à partir de la date de début")
     * @Rest\QueryParam(name="dateMax", map=true, nullable=true, description="Tri à partir de la date de fin")
     *
     * @return $interventions
     */
    public function getInterventionsReportStatsByInterval(Request $request, ParamFetcher $paramFetcher, LoggerInterface $logger)
    {

        $dateMin = $paramFetcher->get('dateMin');
        $dateMax = $paramFetcher->get('dateMax');

        return $qb = $this->getDoctrine()
            ->getRepository(Interventions::class)
            ->queryClosedInterventionsByInterval($dateMin, $dateMax);

    }
}
```

- Test de la requête via POSTMAN :



Angular Frontend

Configurer l'environnement et Créer la requête

- Dans « environment.ts » ajouter la ligne (URL de l'API) :

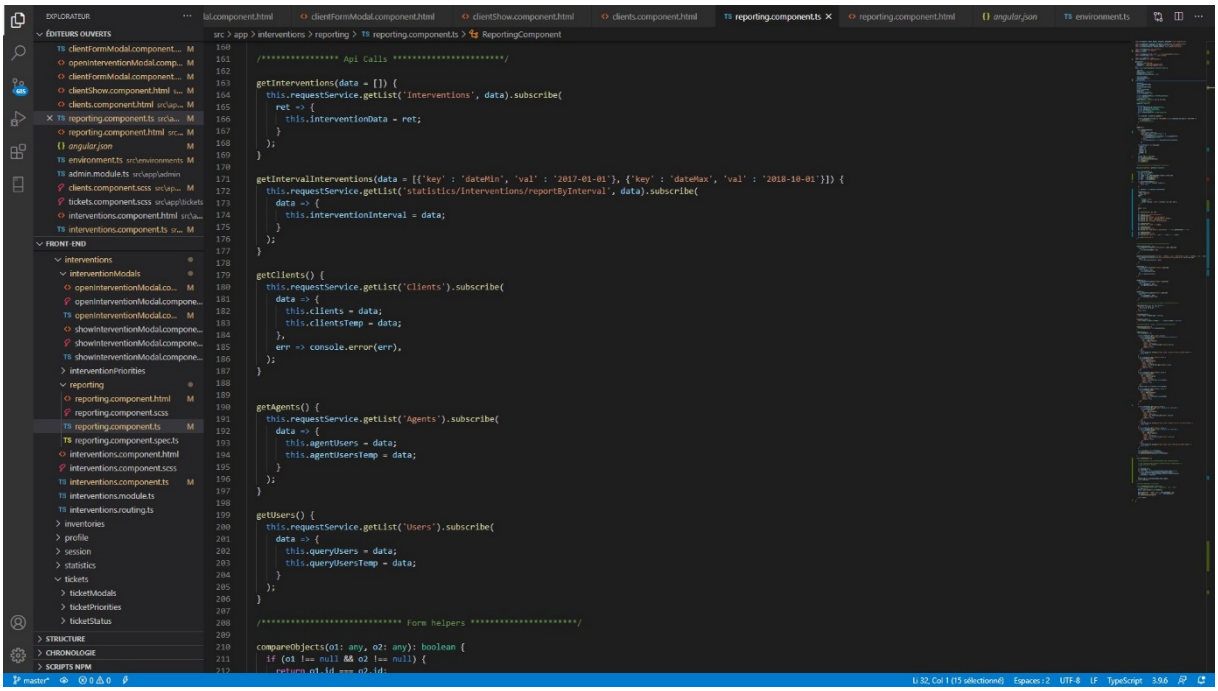
```
apiUrl: 'https://127.0.0.1:8000/api',
```

- Configurer les URL des requêtes dans le fichier :

```
apiFunctionPath: {
'Interventions': '/interventions',
'statistics/interventions/reportByInterval' : '/statistics/interventions/reportByInterval',
```

- Gérer la logique et les requêtes AJAX dans « reporting.component.ts »

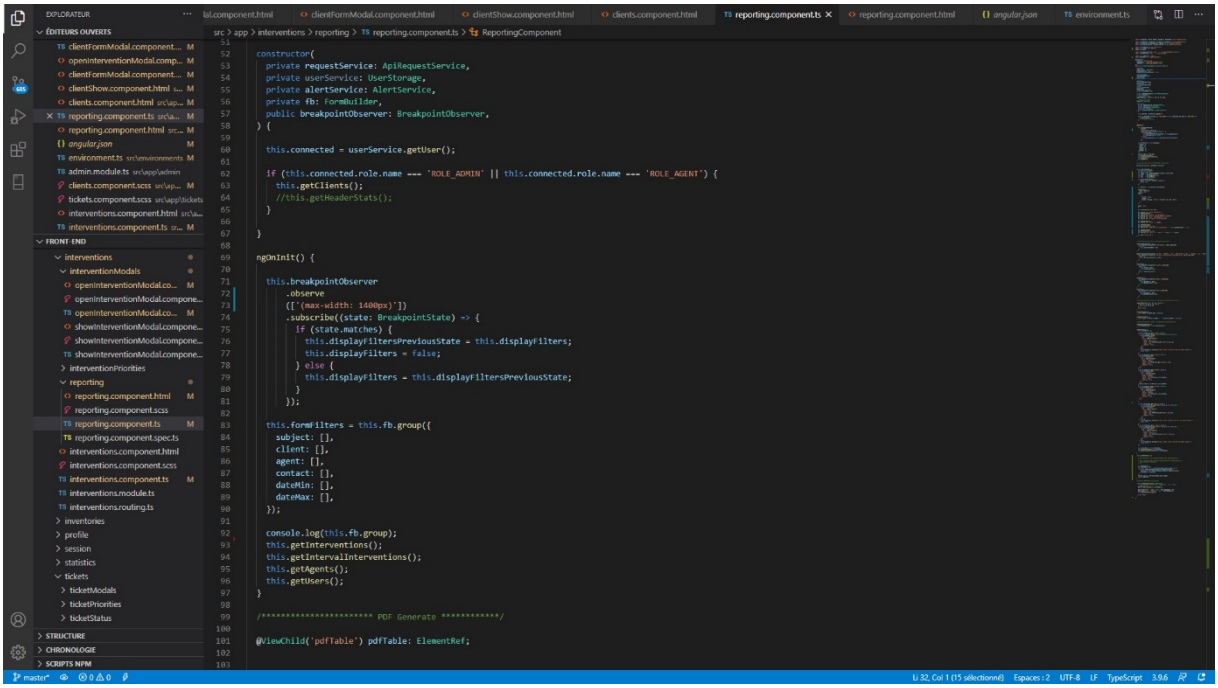
Appels aux requêtes API



```
/*************** Api Calls ********************/

getInterventions(data = []) {
  this.requestService.getList('Interventions', data).subscribe(
    ret => {
      this.interventionData = ret;
    }
  );
}

getIntervallInterventions(data = [{'key' : 'dateMin', 'val' : '2017-01-01'}, {'key' : 'dateMax', 'val' : '2018-10-01'}]) {
  this.requestService.getList('statistics/interventions/reportByInterval', data).subscribe(
    data => {
      this.interventionInterval = data;
    }
  );
}

getClients() {
  this.requestService.getList('Clients').subscribe(
    data => {
      this.clients = data;
      this.clientsTemp = data;
    },
    err => console.error(err),
  );
}


getAgents() {
  this.requestService.getList('Agents').subscribe(
    data => {
      this.agentUsers = data;
      this.agentUsersTemp = data;
    }
  );
}

getUsers() {
  this.requestService.getList('Users').subscribe(
    data => {
      this.queryUsers = data;
      this.queryUsersTemp = data;
    }
  );
}

/*************************** Form helpers *********************/

compareObjects(o1: any, o2: any): boolean {
  if (o1 !== null && o2 !== null) {
    return o1.id === o2.id;
```

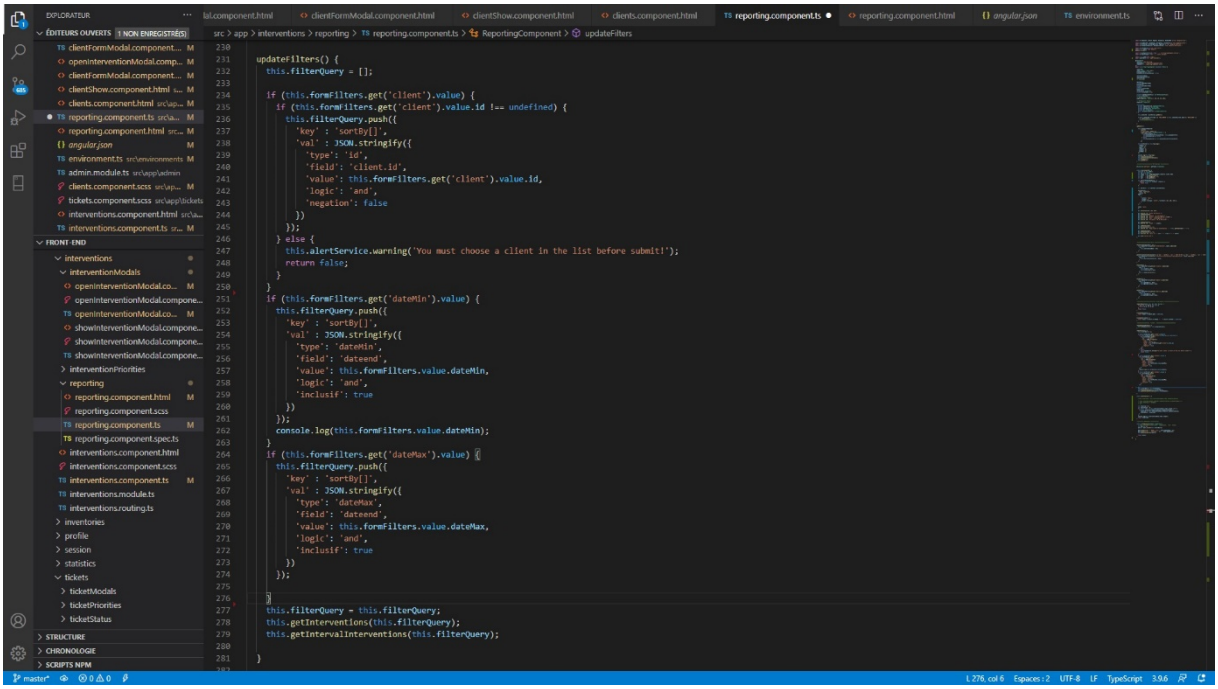
Et appliquer un filtre mis à disposition par Angular et Material :

- Initialiser le filtre dans le fichier « reporting.component.ts »:



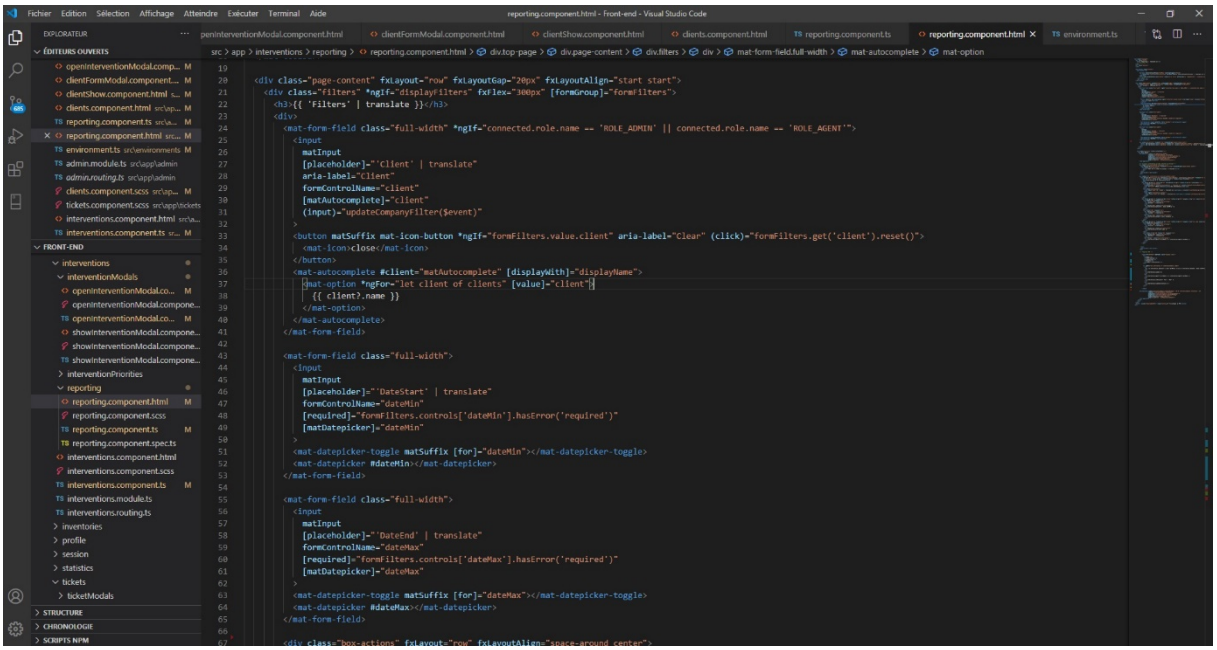
```
constructor(
  private requestService: ApiRequestService,
  private userService: UserStorage,
  private alertService: AlertService,
  private fb: FormBuilder,
  public breakpointObserver: BreakpointObserver,
) {

  this.connected = userService.getUser();

  if (this.connected.role.name === 'ROLE_ADMIN' || this.connected.role.name === 'ROLE_AGENT') {
    this.getClients();
    //this.getHeaderStats();
  }

}

ngOnInit() {

  this.breakpointObserver
      .observe
      (['(max-width: 1400px)'])
      .subscribe((state: BreakpointState) => {
        if (state.matches) {
          this.displayFiltersPreviousState = this.displayFilters;
          this.displayFilters = false;
        } else {
          this.displayFilters = this.displayFiltersPreviousState;
        }
      });

  this.formFilters = this.fb.group({
    subject: [],
    client: [],
    agent: [],
    contact: [],
    dateMin: [],
    dateMax: [],
  });

  console.log(this.fb.group);
  this.getInterventions();
  this.getIntervalInterventions();
  this.getAgents();
  this.getUsers();
}

/********** PDF Generate ***********/

@ViewChild('pdfTable') pdfTable: ElementRef;
```

- Ensuite, créer une méthode gérant la logique du filtre :



```
updateFilters() {
  this.filterQuery = [];

  if (this.formFilters.get('client').value) {
    if (this.formFilters.get('client').value.id !== undefined) {
      this.filterQuery.push({
        'key' : 'sortBy[]',
        'val' : JSON.stringify({
          'type': 'id',
          'field': 'client.id',
          'value': this.formFilters.get('client').value.id,
          'logic': 'and',
          'negation': false
        })
      });
    } else {
      this.alertService.warning('You must choose a client in the list before submit!');
      return false;
    }
  }
  if (this.formFilters.get('dateMin').value) {
    this.filterQuery.push({
      'key' : 'sortBy[]',
      'val' : JSON.stringify({
        'type': 'dateMin',
        'field': 'dateend',
        'value': this.formFilters.value.dateMin,
        'logic': 'and',
        'inclusif': true
      })
    });
    console.log(this.formFilters.value.dateMin);
  }
  if (this.formFilters.get('dateMax').value) {
    this.filterQuery.push({
      'key' : 'sortBy[]',
      'val' : JSON.stringify({
        'type': 'dateMax',
        'field': 'dateend',
        'value': this.formFilters.value.dateMax,
        'logic': 'and',
        'inclusif': true
      })
    });
  }
  this.filterQuery = this.filterQuery;
  this.getInterventions(this.filterQuery);
  this.getIntervalInterventions(this.filterQuery);

}
```

- L’étape suivante sera l’affichage du formulaire et de sa configuration avec Material dans le fichier reporting.component.html :



Création du PDF

Installation et configuration des Modules « jspdf » et « jspdf-autotable » :

- Avec Angular CLI et en ligne de commande : « npm install jspdf jspdf-autable »

- Configuration dans « angular.json » :

```
"../node_modules/jspdf/dist/jspdf.min.js",
"../node_modules/jspdf-autotable/dist/jspdf.plugin.autotable.js"
```

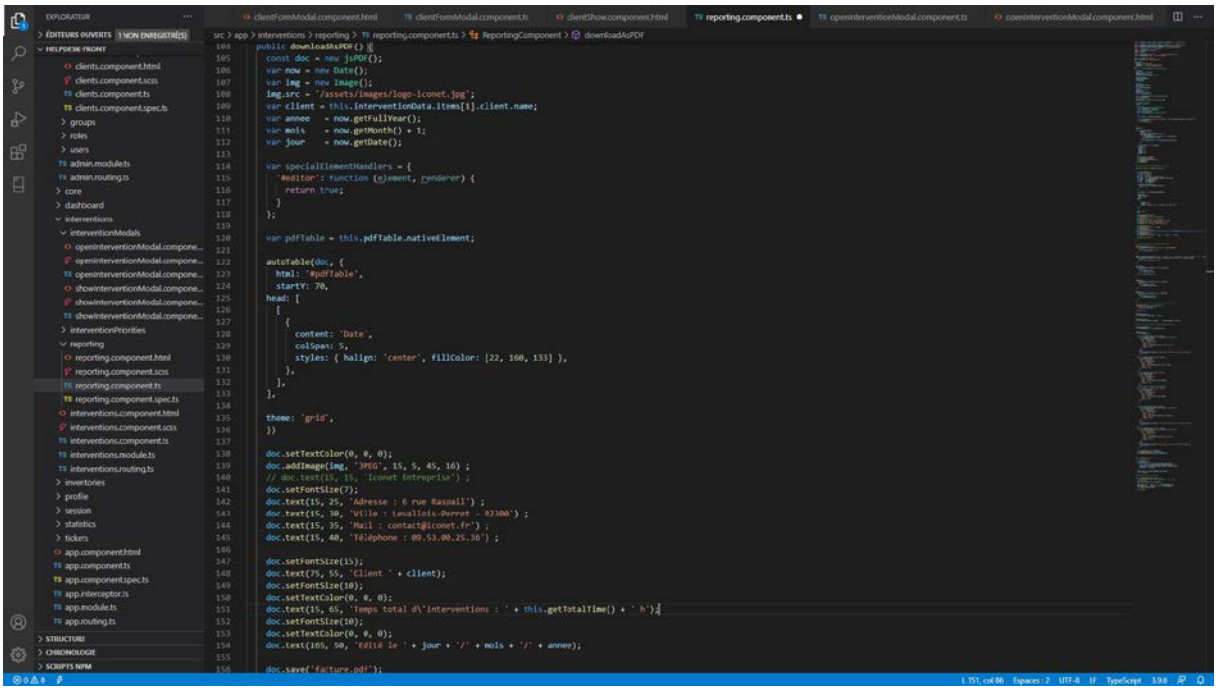
- Utilisation du module « jspdf »

- Implémentation de la logique dans le fichier « reporting.component.ts » et initialisation :

Import des modules dans le fichier .ts :

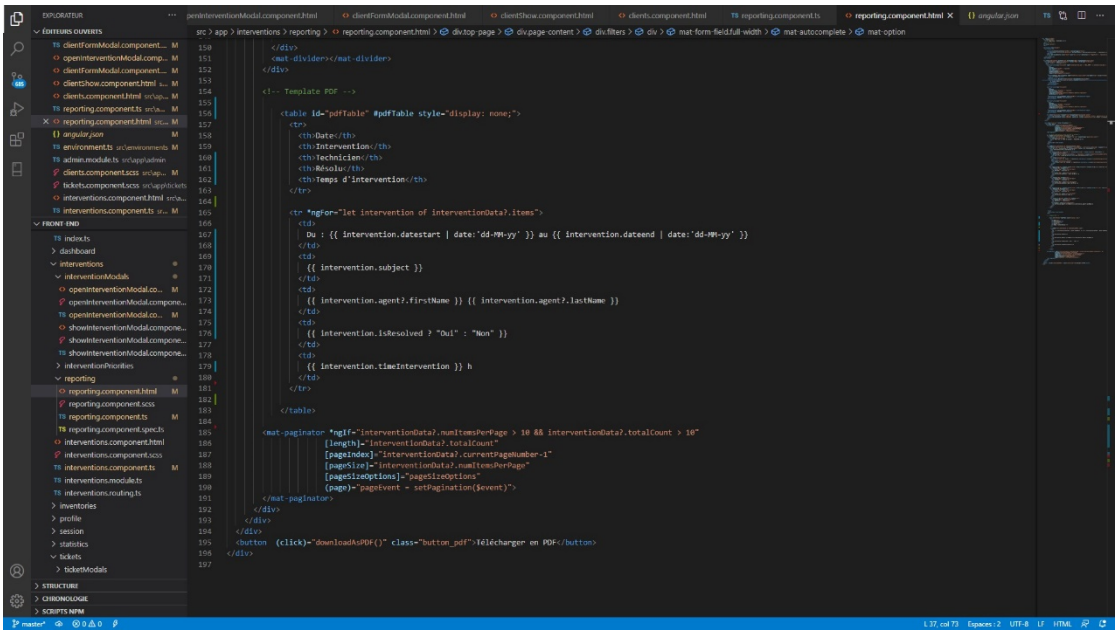
```
import * as jsPDF from 'jspdf';
import autoTable from 'jspdf-autotable';
```

- Création de la logique dans une méthode :



```
public downloadAsPDF() {
  const doc = new jsPDF();
  var now = new Date();
  var img = new Image();
  img.src = '/assets/images/logo-iconet.jpg';
  var client = this.interventionData.items[1].client.name;
  var annee  = now.getFullYear();
  var mois   = now.getMonth() + 1;
  var jour   = now.getDate();

  var specialElementHandlers = {
    '#editor': function (element, renderer) {
      return true;
    }
  };

  var pdfTable = this.pdfTable.nativeElement;

  autoTable(doc, {
    html: '#pdfTable',
    startY: 70,
  head: [
    [
      {
        content: 'Date',
        colSpan: 5,
        styles: { halign: 'center', fillColor: [22, 160, 133] },
      },
    ],
  ],

  theme: 'grid',
  })

  doc.setTextColor(0, 0, 0);
  doc.addImage(img, 'JPEG', 15, 5, 45, 16) ;
  // doc.text(15, 15, 'Iconet Entreprise') ;
  doc.setFontSize(7);
  doc.text(15, 25, 'Adresse : 6 rue Raspail') ;
  doc.text(15, 30, 'Ville : Levallois-Perret - 92300') ;
  doc.text(15, 35, 'Mail : contact@iconet.fr') ;
  doc.text(15, 40, 'Téléphone : 09.53.00.25.36') ;

  doc.setFontSize(15);
  doc.text(75, 55, 'Client ' + client);
  doc.setFontSize(10);
  doc.setTextColor(0, 0, 0);
  doc.text(15, 65, 'Temps total d\'interventions : ' + this.getTotalTime() + ' h');
  doc.setFontSize(10);
  doc.setTextColor(0, 0, 0);
  doc.text(165, 50, 'Edité le ' + jour + '/' + mois + '/' + annee);

  doc.save('facture.pdf');
```

- Création du template PDF dans le fichier reporting.component.html



```
      </div>
      <mat-divider></mat-divider>
    </div>

    <!-- Template PDF -->

        <table id="pdfTable" #pdfTable style="display: none;">
          <tr>
            <th>Date</th>
            <th>Intervention</th>
            <th>Technicien</th>
            <th>Résolu</th>
            <th>Temps d'intervention</th>
          </tr>

          <tr *ngFor="let intervention of interventionData?.items">
            <td>
              Du : {{ intervention.datestart | date:'dd-MM-yy' }} au {{ intervention.dateend | date:'dd-MM-yy' }}
            </td>
            <td>
              {{ intervention.subject }}
            </td>
            <td>
              {{ intervention.agent?.firstName }} {{ intervention.agent?.lastName }}
            </td>
            <td>
              {{ intervention.isResolved ? "Oui" : "Non" }}
            </td>
            <td>
              {{ intervention.timeIntervention }} h
            </td>
          </tr>

        </table>

    <mat-paginator *ngIf="interventionData?.numItemsPerPage > 10 && interventionData?.totalCount > 10"
                   [length]="interventionData?.totalCount"
                   [pageIndex]="interventionData?.currentPageNumber-1"
                   [pageSize]="interventionData?.numItemsPerPage"
                   [pageSizeOptions]="pageSizeOptions"
                   (page)="pageEvent = setPagination($event)">
    </mat-paginator>
  </div>
 </div>
</div>
<button  (click)="downloadAsPDF()" class="button_pdf">Télécharger en PDF</button>
</div>
```

IV. Modification du cahier des charges

À cette demande initiale de l'entreprise, se sont ajoutés en cours de stage des besoins supplémentaires :

- Ajout d'une information supplémentaire dans le PDF concernant le temps d'intervention pour chacune d'elle. Cela a été fait en :
    - Ajoutant un champ « time_intervention » dans la table « interventions » via Symfony avec Doctrine et la modification d'une Entity en ligne de commande ;
    - Modifiant le fichier « InterventionType » dans « Form », Symfony, en ajoutant la nouvelle entrée « ->add('timeIntervention') »
    - Modifiant dans Angular la création, modification et vue d'une intervention en l'incrémentant d'un nouveau champ dans le formulaire concernant les heures d'intervention ;
    - Récupérant dans la partie Reporting l'information « timeIntervention » afin de l'afficher dans le PDF.

- Ajout des informations de contrat pour une compagnie (information qui était inexistante). Cela a été possible en :
    - Ajoutant les champs « contract (booléen), hours_contract (integer), datestart_contract (datetime), contract_time(integer) » dans la table « compagnies » via Symfony avec Doctrine et la modification d'une Entity en ligne de commande ;
    - Modifiant le fichier « ClientsType » dans « Form », Symfony, en ajoutant les nouvelles entrées avec la particularité de configurer le « datetime » ;
    - Modifiant le formulaire de création, mise à jour et l'affichage d'une compagnie en l'implémentant des nouveaux champs de formulaire ;
    - Ajouter un style pour signifier si une entreprise à un contrat ou non en mettant une icône verte si oui ou rouge s'il n'y a pas de contrat avec « [ngClass]="{'contractTrue': client?.contract, 'contractFalse' : !client?.contract}" » et la mise en place du style dans le fichier .scss.

V. Déploiement de l'application web

Les besoins :

- Un accès FTP (possibilité de le faire via le SSH avec les identifiants FTP et l'adresse du serveur)
- Un accès à PHPmyadmin
- Le dossier backend devant s'appeler « helpdesk-api »
- Le dossier frontend devant s'appeler « helpdesk-front »
- Modifier la base de donnée actuelle via Doctrine (Symfony) pour ajouter les nouvelles tables et champs avec : **php bin/console doctrine:migrations:migrate**
- Configurer le fichier .env avec les identifiants à la base de données. Mettre l'environnement en « prod »

L'installation :

- Avant toute modification, il faut effectuer une sauvegarde (backup) via PLESK
- Via le FTP aller dans le dossier : **/var/www/vhosts/helpdesk.iconet.fr**
- Renommer les dossiers actuels « **helpdesk-api** » et « **helpdesk-front** » en ajoutant « **.back** » afin de les garder en sauvegarde
- Côté **Symfony** envoyer via le FTP le dossier en supprimant les dossiers « var/cache « et « var/log »
- Côté **Angular**, aller dans le fichier **environnement.prod.ts** et ligne 43 (si le code n'a pas été changé) configurer la requête vers l'API comme suit : **apiUrl: 'https://helpdesk-api.iconet.fr/api',**
- Ensuite lancer via une ligne de commande : **ng build –prod**. Ce qui va créer un dossier « **dist** » qu'il faudra nommer « helpdesk-front » et l'envoyer via le FTP sur le serveur.

Problèmes rencontrés :

- Problème de requête CORS : aller dans le fichier **.env** coté **Symfony** et ligne 26 configurer : **CORS_ALLOW_ORIGIN=^https?://helpdesk.iconet.fr$**
- Si problème de connexion à la base de données (**error 401 Bad credentials**) : faire attention à la casse des noms de tables dans la base de données. Certaines tables doivent commencer par une lettre majuscule. Se référer à la base de données actuelle pour savoir quel nom de table changer.

VI. Résumé de mon action

- Installer et configurer l'environnement du Backend (Symfony) et du Frontend (Angular) en local par le biais de WAMP ;
- Configurer l'API REST pour avoir accès à certaines ressources (interventions et plage de temps) ;
- Tester les différentes requêtes via le logiciel POSTMAN ;
- Configurer les requêtes vers l'API REST dans l'environnement d'Angular ;
- Ajouter une nouvelle section Reporting dans le menu latéral;
- Créer un Component « reporting » ;
- Effectuer les requêtes, la gestion de la logique gérant le formulaire et l'utilisation du Module « jspdf » dans le fichier reporting.component.ts ;
- Créer l'affichage de la nouvelle section « reporting » qui comprend la liste d'intervention, un formulaire de tri selon les besoins et un bouton permettant l'export PDF via le fichier reporting.component.html ;
- Modifier la table « interventions » afin d'ajouter un champ « hours_intervention » via Symfony. Ajout d'un champ de formulaire dans la création et modification d'une intervention ;
- Modifier la table « compagnies » afin d'ajouter les informations de contrat. Ajout également dans le Front des champs de formulaire correspondant ;
- Déploiement de l'application web sur le serveur de l'entreprise.

VII. Bilan du stage

En partant des objectifs personnels qui étaient les miens en amont du stage, continuez la progression en termes de compétences, connaissances et apprentissage de frameworks, je peux dire que ce stage a été riche d'enseignements.

J'ai appris bien plus que je ne l'espérais. Habitué pendant la formation à partir d'une page blanche afin d'établir l'architecture, le style et les fonctionnalités demandés, ce stage m'a permis une nouvelle entrée (et un challenge de taille) : reprendre un code existant, utilisant deux Frameworks, dans une entreprise n'ayant aucun développeur pour l'expliquer afin d'ajouter de nouvelles fonctionnalités.

Ce contexte relativement particulier m'a demandé d'assimiler plusieurs notions tout en respectant le délai de stage qui est de deux mois. Durant cette période, j'ai appris:

- Les notions de « reverse engineering » afin de comprendre la construction d'un code existant ;
- Les principes d'utilisation des frameworks Symfony et Angular ;
- Utiliser les lignes de commandes avec Composer et NPM ;
- Le fonctionnement et la configuration d'un API REST en Backend (Symfony);
- À tester un API REST en envoyant des requêtes avec un logiciel type POSTMAN ;
- À faire communiquer backend et frontend utilisant des frameworks différents;
- La construction d'une Single Page Application via le système de routing proposé par Angular ;
- La création de formulaires avec Angular et Material et le système de filtre ;
- Me servir des Bundles (Symfony) et/ou Modules (Angular) pour profiter facilement de nouvelles fonctionnalités.

En parallèle de l'aspect technique, une immersion en entreprise avec ce que cela comporte en termes de travail en équipe, demande, contrainte de temps, changement de cahier des charges a été riche d'enseignement.

Ce stage a été dense en informations à assimiler et assez complexe, j'ai parfois ressenti le manque d'un développeur web dans l'équipe. Néanmoins, le challenge que cela m'a imposé a été très stimulant et m'a donné l'envie d'approfondir l'apprentissage de frameworks. L'installation d'une fonctionnalité « Reporting », simple d'apparence, m'a demandé de faire travailler le backend et le frontend en passant la compréhension et l'utilisation d'un API REST.

Cet ajout du « Reporting » a permis à l'entreprise d'utiliser un outil en moins « excel » qui était chronophage. Cela a permis aussi d'avoir une réflexion sur l'ajout d'autres fonctionnalités afin de continuer la réduction des outils et la centralisation des informations dans l'application web interne. La volonté de l'entreprise, au-delà de l'utiliser en interne, est de vendre cette application à d'autres entreprises.

Cette expérience a conforté mon envie et la motivation de faire du développement web mon métier. J'ai trouvé une grande stimulation intellectuelle à apprendre tant de notions tout en respectant un délai. J'apporterai une nuance dans le fait d'être l'unique développeur web (qui plus est stagiaire). Si le choix s'offre à moi, je privilégierai le travail au sein d'une équipe de développeurs web.

Je conclurais ce rapport de stage en mentionnant le très grand plaisir que j'ai eu au cours de la formation de développeur web junior qui a été pour moi une révélation.

Je remercie le lieu de stage de m'avoir accueilli si agréablement (et de m'avoir fait sué intellectuellement ^^).