

ENI ECOLE INFORMATIQUE
Développeur Web et Web Mobile



ROLLET
MATHIEU

RAPPORT DE STAGE

REDECSO

2021

RESUME

Afin de clôturer mon parcours d'étudiant à la formation Développeur Web et Web Mobile, j'ai effectué mon stage en distanciel au sein de l'entreprise REDECSO, petite structure dans la rénovation et décoration d'intérieur.

Le rapport traitera des sujets étudiés et réalisés lors de mon stage de deux mois en entreprise. Il s'agit donc de l'ensemble de mes acquis lors de ma formation à l'ENI et de mon projet développé pour l'entreprise REDECSO.

L'entreprise REDECSO ne possédant pas de service de développement web, j'ai été amené à enrichir mes connaissances par moi-même en tant que développeur web et de mettre à profit les conseils de mes coachs.

Lors de cette période de stage, ma mission était de mettre en œuvre une application web qui consisterait à créer des devis pour des clients. À la demande de l'entreprise, il était nécessaire de recenser des clients, et leurs informations, ainsi que de recenser certains produits "types" qui sont fréquemment utilisés par l'entreprise pour faciliter la création d'un devis.

Cela m'a apporté des compétences supplémentaires dans la conception d'un projet client, tant dans son analyse que dans sa réalisation.

J'ai également eu la possibilité de mener en parallèle une activité de recherche aux problèmes rencontrés et d'affûter l'autonomie et la méthodologie nécessaires à ce métier.

Ce rapport conclut l'ensemble de mes deux mois passés à la création d'une application pour l'entreprise REDECSO, ainsi que de ma formation de 8 mois à l'ENI.

REMERCIEMENTS

Tout d'abord,

Je remercie ENI Ecole de m'avoir laissé une chance d'intégrer et apprendre énormément dans cette formation pour le titre professionnel de développeur web et web mobile.

Conscient que nous ne pouvons pas tout connaître en six mois de formation, cependant, nous pouvons toujours apprendre et nous améliorer, que ce soit dans un milieu professionnel, directement en ligne sur Internet ou même les deux. Notre formation nous a appris à être autonomes, à rechercher continuellement les informations ainsi qu'à les trier. Elle nous a appris aussi à être conscients que les technologies sont perpétuellement renouvelées et mises à jour, pour cela, il nous faut nous tenir au courant et entretenir notre culture générale dans notre domaine (veille technologique).

Nous avons la chance de trouver une multitude de plateforme (OpenClassRoom, Udemy, Stack Overflow...), des sites internet avec des exemples, de la documentation, et même des tutoriels en vidéo.

Pour cela, je tiens donc à remercier mes coachs pour la patience dont ils font preuve. D'une façon générale, je remercie aussi l'ensemble de l'équipe ENI École et mes camarades, avec qui nous avons pu échanger, débattre et apprendre.

PRESENTATION ENTREPRISE REDECSO

REDECSO entreprise fondée en 2014,

Par Tahar Haddadi et son Co-gérant Hervé, c'est une entreprise qui est spécialisée dans la rénovation et décoration d'intérieur dans le bâtiment. L'entreprise travaille essentiellement sur des chantiers de petit œuvre.

Elle effectue des travaux de rénovation principalement dans le domaine de l'Electricité, Peinture, Plomberie, et la Maçonnerie. L'entreprise est amenée à travailler régulièrement pour de grandes entreprises de bureau aussi bien qu'avec des particuliers. Il arrive que des clients travaillent avec des architectes pour mettre en œuvre toutes leurs demandes et s'assurer du bon déroulement d'un chantier en collaboration avec l'entreprise.

L'entreprise REDECSO emploie actuellement 7 employés dans la région Parisienne, et travaille principalement à Paris et la périphérie Parisienne.

Etant une petite structure elle ne justifie pas le besoin d'avoir une équipe informatique et de développeur. Conscient que ses outils informatiques sont limités, elle travaille sur des logiciels informatiques tel que Microsoft Office dans la gestion des devis et facturation.

REDECSO entreprise avec un côté familiale, a toujours mis en œuvre d'aider et d'accompagner des jeunes dans la vie active et leur formation.

CACHIER DES CHARGES

1. Présentation du contexte	5
1.	
2. Objectif de l'application	6
3. L'Analyse du projet	6
2.	
4. Structure de la base de données	6
a. Remarque MCD	7
5. Architecture et sécurité web	8
6. Les fonctionnalités	9
a. Création client	10
7. L'avenir et conclusion du projet	11

CACHIER DES CHARGES – 1. Présentation du contexte

L'entreprise REDECSO n'a actuellement aucun site vitrine malgré son désir d'espérer un jour en posséder un et n'utilise aucun outil professionnel car aucun outil ne correspond ou bien n'est réellement adapté à sa demande. Lors de la création de devis elle utilise Microsoft Word comme simple éditeur de texte, un bloc note comme référencement de produit, et une calculatrice pour calculer les montants.

Dans ce contexte, j'ai longuement échangé avec le dirigeant et le directeur commercial pour connaître son travail et surtout comprendre la vision et l'utilité d'une application de gestion des devis. Il était important de proposer une application simple d'utilisation et intuitive.

Lors de nos échanges concernant le projet d'une application qui aurait pour but la création d'un devis et d'une facture, l'entreprise REDECSO a exprimé une volonté à la sécurisation de ses données, il était donc important de dissocier un site vitrine et un outil professionnel pour la gestion des devis et des factures.

Ce que souhaitait aussi le directeur commercial c'était mettre en lien un référencement des clients, et de certains produits pour pouvoir générer plus facilement des devis lorsqu'il s'occupe de chantier de rénovation dits « classiques »

Avec tous ces éléments j'ai donc proposé un projet plus en détails, et lui ai expliqué qu'il me faudra mettre en place une base de données qui, globalement contiendrait plusieurs tables telles qu'USER pour stocker les utilisateurs, ou encore la table PRODUIT, permettant d'enregistrer les caractéristiques d'un produit (prix, marque, etc...) afin de pouvoir les sélectionner ultérieurement lors de la création d'un devis.

L'application doit permettre, une fois la création d'un devis ou d'une facture effectuée, la possibilité de générer un PDF avec tous les détails créés sur l'application.

CACHIER DES CHARGES – 2. Objectif de l'application :

L'application en elle-même devait permettre au commercial de l'entreprise d'avoir la possibilité d'utiliser l'application de n'importe où pour lui permettre la création d'un devis et de générer des factures le plus simplement possible dans une logique propre à l'entreprise.

CACHIER DES CHARGES – 3 L'Analyse du projet :

L'entreprise REDECSO est une petite structure de ce fait elle a sa propre façon de fonctionner, c'est important de le comprendre dès le début pour analyser au mieux le projet. C'est comme cela que j'ai découvert comment se construisait étape par étape un devis lorsque le commercial était en pleine rédaction d'un devis.

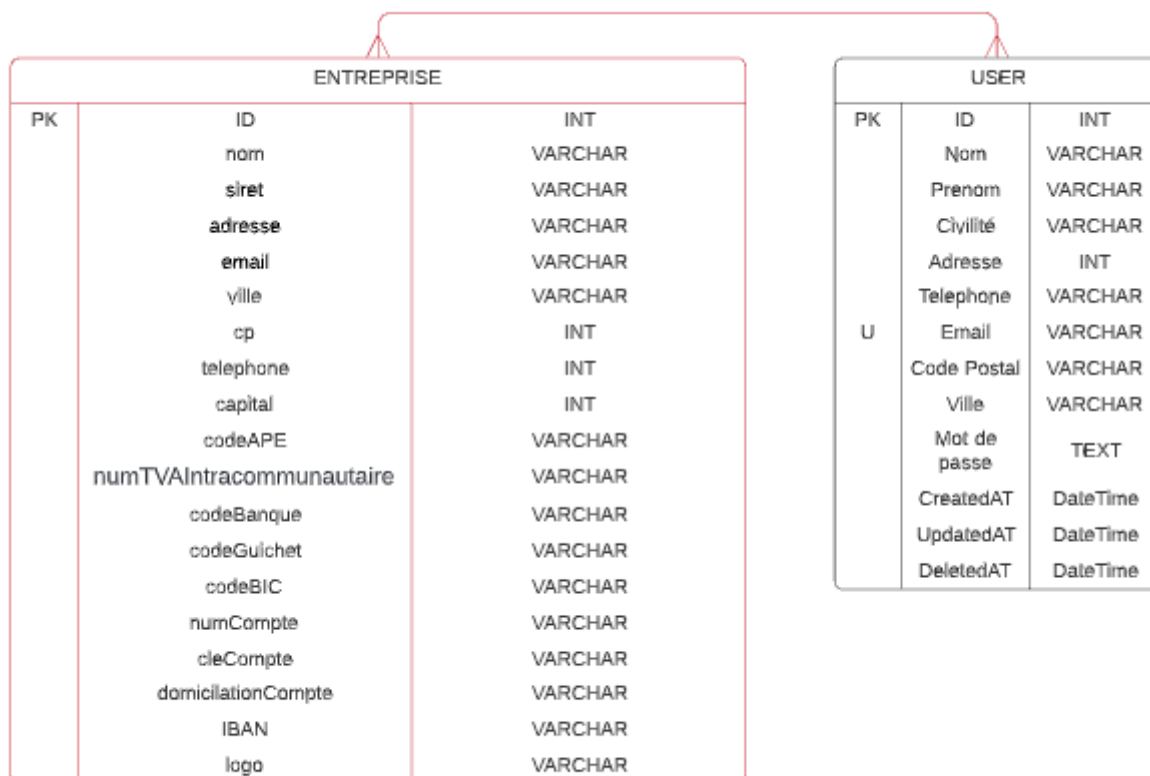
Il n'était pas question de faire des devis classiques, avec une seule méthode mais bien de faire des devis personnalisés, et personnalisable pour chaque client dont l'entreprise s'occupe.

CACHIER DES CHARGES – 4 Structure de la base de données :

- **ENTREPRISE**, pour le référencement des entreprises.
- **USER**, pour l'authentification sur l'application
- **CLIENTS**, pour le référencement des clients.
- **DEVIS**, pour la création d'un devis.
- **CATEGORIES**, car l'entreprise regroupe plusieurs catégories de métier.
- **TACHES**, pour donner un prix de base sur des tâches récurrentes, ou bien regrouper des produits dans la tâche en elle-même et donc calculer le prix de la tâche plus les produits.
- **PRODUITS**, pour référencer les noms et prix liés à une marque.
- **MARQUES**, pour le référencement des produits et les lier à une marque.

CACHIER DES CHARGES – REMARQUE MCD

- ⇒ **ENTREPRISE** : Concernant la table ENTREPRISE j'ai dû me pencher sur un problème en particulier. Quelle sera la relation avec la table User. J'ai pris la décision de mettre en ManyToMany car la logique sur une petite application web voudrait qu'un utilisateur ne puisse pas faire partie de plusieurs entreprises. Seulement, dans un souci de développement, il faut aussi penser à l'avenir. Si l'application devient un outil que plusieurs entreprises utilisent, il se peut qu'un comptable travaille pour plusieurs entreprises, ou bien un service juridique.
- ⇒ **USER** : La table user est indispensable pour utiliser l'application web. On peut remarquer que j'ai rajouté trois champs non indispensables (*createdAT*, *updatedAT*, *DeletedAT*) mais toujours dans un souci d'information. Il est nécessaire selon moi d'essayer de prévoir un bon MCD pour l'avenir.



CACHIER DES CHARGES – Architecture et sécurité web

La section Access control permet de gérer la navigation d'un utilisateur en prenant en compte son rôle. Ainsi, seulement un utilisateur ayant le rôle **'ROLE_ADMIN'** a le droit d'accéder aux url / ...

```
# Easy way to control access for large sections of your site
# Note: Only the *first* access control that matches will be used
access_control:
  - { path: ^/login, roles: [] }
  - { path: ^/, roles: [ROLE_ADMIN] }
  # - { path: ^/profile, roles: ROLE_USER }
```

La conception et l'utilisation pour le début de l'application étaient claires, cela serait un outil interne à l'entreprise.

En revanche elle devra pouvoir être accessible dans un environnement moins sécurisé que sur le réseau privé de l'entreprise mais pouvant évoluer aussi sur des réseaux publics.

Pour verrouiller l'accès à l'application web, il a fallu tout simplement obliger que pour qu'un utilisateur utilise l'application, il devra être enregistré, et connecté.

Grace au **security.yaml** dans Symfony5 cela reste très simple. Toutes nos routes qui commenceraient par « / » sont interdites si vous ne possédez pas le rôle **ROLE_ADMIN**.

En revanche, on peut aussi remarquer que la seule page accessible sans rôle, est l'url « /login ». Ce qui se produit lorsque vous n'avez pas d'accès à une certaine route, automatiquement vous serez redirigé sur une page de login.

```
class SecurityController extends AbstractController
{
    /**
     * @Route("/login", name="app_login")
     */
    public function login(AuthenticationUtils $authenticationUtils): Response
    {
        $error = $authenticationUtils->getLastAuthenticationError();
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('view: security/login.html.twig', ['last_username' => $lastUsername, 'error' => $error]);
    }
}
```

CACHIER DES CHARGES – 7 Fonctionnalité : Création client

Rechercher:

Nom	Prenom	Entreprise
Davignon	Sidney	
Gradasso	Lambert	
Lehazif	Anthony	
Marquis	Aurélie	
maxime	nomdefamille	
Roberto	Robert	
rollet		✓
Trechebat		✓

Cocher si c'est une entreprise ☐

Nom

Prenom

Adresse

Complément d adresse

Numéro

Email

CP

Ville

Pour enregistrer un client nous avons deux particularités, soit c'est un client simple ou bien une entreprise. Car nous le savons, l'entreprise REDECSO peut être amenée à travailler pour des entreprises comme pour des particuliers.

Si c'est une entreprise nous pouvons enlever les champs non requis pour enregistrer l'entreprise ou le client.

Rechercher:

Nom	Prenom	Entreprise
Davignon	Sidney	
Gradasso	Lambert	
Lehazif	Anthony	
Marquis	Aurélie	
maxime	nomdefamille	
Roberto	Robert	

Cocher si c'est une entreprise ☒

Nom

Adresse

Complément d adresse

Numéro

Email

CP

Ville

CACHIER DES CHARGES – Fonctionnalité : Création client

L'utilisation de JavaScript était nécessaire pour générer le html pour rendre notre modale dynamique :

Ouvrir la modale :

```

////////////////// ADD CLIENT ////////////////////
$(html: '.add_client').on( events: 'click', handler: function (){
  // console.log($(this).data('target'));
  $.get($(this).data( key: 'target'), success_data: function (html){
    $(html: '#modal-body').html(html);
    let modal = new bootstrap.Modal(document.getElementById( elementId: 'modale'), {});
    modal.show();
  })
})

```

Afficher ou cacher des champs en fonction d'une action sur la page web avec le .toggle() :

```

////////////////// CASE MODALE ENTREPRISE ////////////////////
$(document).on( events: 'change', selector: '#client_entreprise', handler: function (){
  $(html: '#client_civilite').toggle();
  $(html: '#client_prenom').toggle();
})

```

Création et traitement du formulaire :

```

#[Route('/new', name: 'client_new', methods: ['GET', 'POST'])]
public function new(Request $request): Response
{
    $client = new Client();

    $form = $this->createForm( type: ClientType::class, $client,[
        'action'=>$this->generateUrl( route: '/new'),
    ]);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        // dd($form->getData());
        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->persist($client);
        $entityManager->flush();

        return $this->redirectToRoute( route: 'client_index',[], status: Response::HTTP_SEE_OTHER);
    }

    return $this->renderForm( view: 'client/new.html.twig', [
        'client' => $client,
        'form' => $form,
    ]);
}

```

CACHIER DES CHARGES – L’avenir et conclusion du projet

Concernant l’avenir de l’application pour l’entreprise REDECSO, elle n’est pas fonctionnelle et pour cela elle ne pourra donc être utilisée. Cependant j’ai exprimé mon souhait de pouvoir continuer à la développer en indépendant pour enrichir mes connaissances personnelles.

En effet, conclure sur une application non fonctionnelle me laisse sur ma faim, c’est pourquoi je souhaite aller au bout de ce projet.

Nous aurions pu pousser le projet plus loin, en intégrant de multiples fonctionnalités telle que la possibilité d’envoyer un mail directement avec le pdf.

Cependant, conscient de mes capacités, et étant complètement autonome il est important de rester conscient des difficultés et des contraintes du temps imparti.

LE PROJET – SOMMAIRE

1. Présentation Front-end	13-15
2. Présentation Back-end	16 - 17
a. Les migrations	18
b. Le Repository	19 – 20
c. Le Controller	21
3. Liste des compétences	22
4. Environnement de développement (IDE)	23
5. Framework	24
6. Langages utilisés	25
7. Langage et Template	26 – 24
8. Environnement de développement	28
9. Conclusion	29

LE PROJET – Front-end

Pour mener à bien notre Front, **Bootstrap**, **Form**, **Twig**, **DataTables**, et **select2** ont été utilisés lors de ce projet, pour cela nous avons utilisé « **Composer** » et « **Yarn** » qui sont les gestionnaires de packages utilisés par les applications PHP modernes. **Composer** sert à gérer les dépendances dans les applications Symfony. Ainsi la commande pour installer nos packages **Twig** et **Form** est : `COMPOSER REQUIRE DOCTRINE TWIG FORM` qui nous nous aideront grandement dans la gestion et la récupération des données en back. Nous avons eu besoin aussi de Yarn qui est le gestionnaire de packages front. Nous avons aussi utilisé la `commande yarn add bootstrap @popperjs/core` qui est fournie dans la documentation de Bootstrap, suivie de la commande `yarn install`, qui nous permet d'installer les dépendances listées dans le fichier `package.json`.

```
"dependencies": {
  "@fortawesome/fontawesome-free": "^5.15.4",
  "@popperjs/core": "^2.10.1",
  "bootstrap": "^5.1.1",
  "datatables.net": "^1.11.2",
  "datatables.net-bs5": "^1.11.2",
  "datatables.net-buttons": "^2.0.0",
  "datatables.net-buttons-bs5": "^2.0.0",
  "datatables.net-datetime": "^1.1.1",
  "datatables.net-dt": "^1.11.2",
  "datatables.net-editor-bs5": "^2.0.3",
  "datatables.net-responsive-bs5": "^2.2.9",
  "datatables.net-select-bs5": "^1.3.3",
  "jquery": "^3.6.0",
  "select2": "~4.0"
```

A la suite de cela ce qui nous permettra d'installer le bundle de Bootstrap sur notre projet. Pour importer Bootstrap sur tout le projet, il nous suffira de configurer notre `app.js`.

```
app.js x
10
11 // start the Stimulus application
12 import './bootstrap';
13
```

LE PROJET – Front-end

Pour DataTables le principe est le même, installation par yarn et utilisé la commande : `yarn add datatables.net` puis l'importation dans le `app.js` et l'initialisation de la ou les variable(s) a utilisé dans le projet.

```

app.js x
24 window.datatablefr = require('./datatable_fr.json');
25 var $ = require( 'jquery' );
26
27 require( 'datatables.net-bs5' )();
28 require( 'datatables.net-editor-bs5' )();
29 require( 'datatables.net-buttons-bs5' )();
30 require( 'datatables.net-buttons/js/buttons.html5.js' )();
31 require( 'datatables.net-datetime' )();
32 // require( 'datatables.net-responsive-bs5' )();
33 require( 'datatables.net-select-bs5' )();

```

Une fois notre `app.js` configuré, il nous faudra configurer notre `webpack.config.js` Pour cela nous devons utiliser des alias correspondants à nos **fichiers JS** pour les compiler, les importer et les copier dans le dossier `public/` afin d'être accessibles par notre `base.html.twig`. Il faut ensuite utiliser la commande `yarn dev` (*yarn prod, en environnement de prod*) dans la console ce qui effectuera la compilation pour nos assets.

```

webpack.config.js x
28 */
29 .addEntry( name: 'app', src: './assets/app.js')
30 .addEntry( name: 'login', src: './assets/js/login.js')
31 .addEntry( name: 'home', src: './assets/js/home.js')
32 .addEntry( name: 'client', src: './assets/js/client.js' )
33 .addEntry( name: 'js/app', src: './assets/app.js')
34 .addStyleEntry( name: 'css/app', src: './assets/styles/app.scss')
35 .addEntry( name: 'produit', src: './assets/js/produit.js' )
36 .addEntry( name: 'produit_new', src: './assets/js/produit/new.js')
37 .addEntry( name: 'devis', src: './assets/js/devis.js' )
38 .addEntry( name: 'devis_show', src: './assets/js/devis/devis_show.js' )
39 .addEntry( name: 'devis_client', src: './assets/js/devis/form_devis_client.js')
40 .addEntry( name: 'navtool_devis', src: './assets/js/partials/navtool_devis.js')
41 .addEntry( name: 'marque', src: './assets/js/marque.js' )
42 .addEntry( name: 'navbar', src: './assets/js/partials/navbar.js')
43 .addEntry( name: 'tache', src: './assets/js/tache.js' )
44

```

LE PROJET – Front-end

Voici un exemple où **Twig**, **DataTables**, et **Bootstrap** sont présents.

Ce sont des outils extrêmement pratiques et fonctionnels pour avoir un rendu client très rapidement.

Rechercher:

Nom	Prenom	Entreprise	Telephone	Email	actions
Davignon	Sidney		05.28.13.87.64	DavignonSyd@hotmail.fr	
Gradasso	Lambert		04.39.33.22.42	GradassoLambert@jourrapide.com	
Lehazif	Anthony		0614327992	anthony@hotmail.fr	
Marquis	Aurélié		01.18.58.60.93	AuréliéMarquis@hotmail.fr	
maxime	nomdefamille		0614327993	maxime@laposte.net	
Roberto	Robert		0625321545	roberto@hotmail.fr	
rollet		✓	0614327993	mathieu@laposte.net	
Trecobat		✓	0132487562	trecobat@hotmail.fr	

Affichage de 1 à 8 sur 8 éléments

Précédent 1 Suiv

Ajouter un nouveau client

```
<table id="dataTable" class="table table-bordered table-striped text-white my-5">
  <thead>
    <tr>
      <th>Nom</th>
      <th>Prenom</th>
      <th>Entreprise</th>
      <th>Telephone</th>
      <th>Email</th>
      <th>actions</th>
    </tr>
  </thead>
  <tbody>
    {% for client in clients %}
      <tr class="table-dark">
        <td class="table-secondary text-center">{{ client.Nom }}</td>
        <td class="table-secondary text-center">{{ client.prenom }}</td>
        <td class="table-secondary text-success text-center">{{ client.entreprise ? '<i class="fas fa-check-circle"></i>' : ' ' }}</td>
        <td class="table-secondary text-center">{{ client.telephone }}</td>
        <td class="table-secondary text-center">{{ client.email }}</td>
        <td class="d-flex">
          <button type="button" data-target="{{ path('/{id}', {'id': client.id}) }}" class="client_show btn btn-outline-success mx-1 ">
            <i class="fas fa-eye"></i>
          </button>

          <button type="button" data-target="{{ path('/{id}/edit', {'id': client.id}) }}" class="client_edit btn btn-outline-warning mx-1 ">
            <i class="fas fa-pen fa-xs"></i>
          </button>

          <form method="post" action="{{ path('/{id}', {'id': client.id}) }}" onsubmit="return confirm('Êtes-vous sûr de vouloir supprimer l'
informations du client ?');">
            <input type="hidden" name="_token" value="{{ csrf_token('delete' ~ client.id) }}">
          </form>
        </td>
      </tr>
    {% endfor %}
  </tbody>
</table>
```

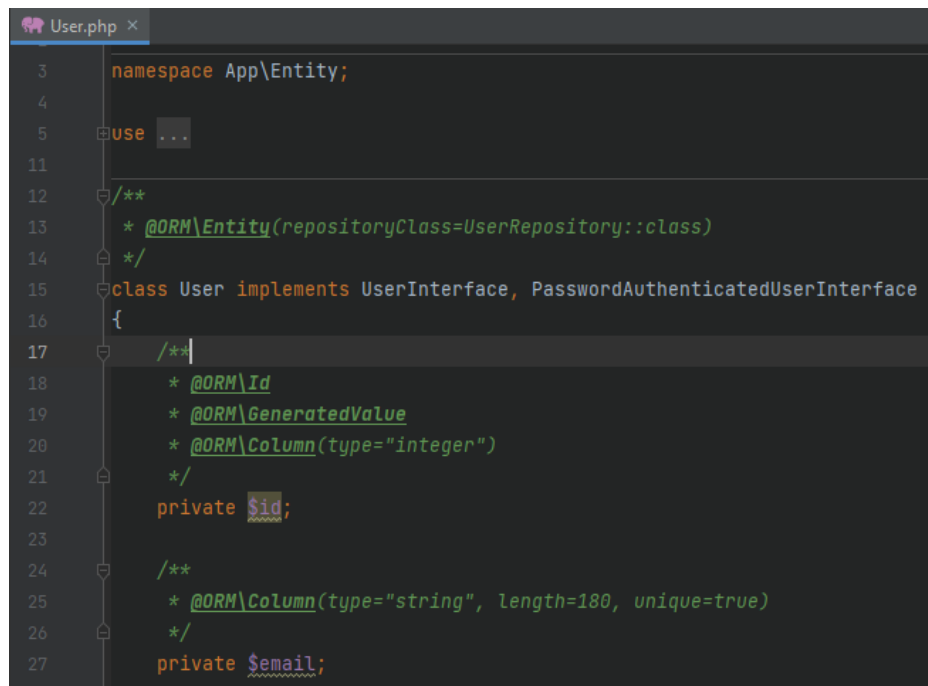

LE PROJET – Back-end

Le back end a pour rôle de récupérer les données et les traiter afin de les renvoyer à la vue. Il comporte les modèles et les contrôleurs du paradigme MVC

Symfony nous fournit également de nombreux outils afin de gérer ces fonctionnalités. Nous commencerons par étudier le côté base de données avec l'ORM Doctrine, suite à quoi nous aborderons les contrôleurs avec les traitements des données.

Doctrine est un outil de gestion de la base de données qui, associé à Symfony qui nous permet de gérer la base de données directement depuis notre application.

Afin de pouvoir décrire la structure de notre base de données nous utiliserons les entités, appelées Entity par Symfony. Chaque Entity représentera une table et possèdera comme propriétés les différents champs de cette table. Ainsi la commande `symfony console make:entity` permet de créer une entité vierge à laquelle on donnera le nom de la table au singulier.



```

3 namespace App\Entity;
4
5 use ...
6
7
8
9
10
11
12 /**
13  * @ORM\Entity(repositoryClass=UserRepository::class)
14  */
15 class User implements UserInterface, PasswordAuthenticatedUserInterface
16 {
17     /**
18      * @ORM\Id
19      * @ORM\GeneratedValue
20      * @ORM\Column(type="integer")
21      */
22     private $id;
23
24     /**
25      * @ORM\Column(type="string", length=180, unique=true)
26      */
27     private $email;
28 }

```

Comme on peut le voir, on indique en annotation les différentes informations nécessaires à la création des différents champs (type, longueur, contraintes...)

LE PROJET – Back-end

De plus, afin de sécuriser les données à l'aide de l'encapsulation, ces propriétés sont mises privées. Il nous faudra donc les Accesseurs afin de pouvoir récupérer et/ou éditer ces données.

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

public function getEmail(): ?string
{
    return $this->email;
}

public function setEmail(string $email): self
{
    $this->email = $email;

    return $this;
}
```

Ainsi sur l'image ci-dessus nous pouvons voir que nous pouvons accéder à l'ID sans pouvoir le modifier contrairement à l'email que nous pouvons consulter et éditer.

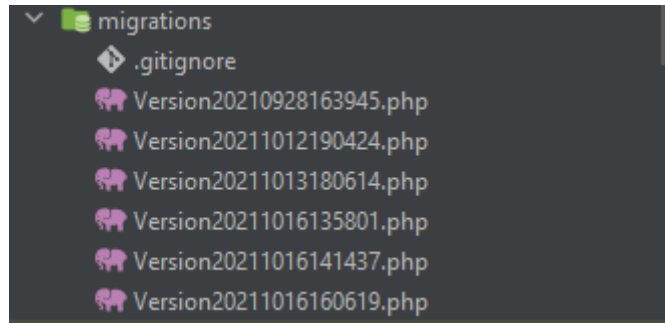
De plus les entités nous permettent de définir les relations entre les tables.

```
/**
 * @ORM\ManyToMany(targetEntity=Entreprise::class, inversedBy="users")
 */
private $entreprises;
```

On peut ainsi voir, toujours en annotation, qu'on indique les différentes informations permettant de créer le champs (La table liée via son entité ainsi que la propriété permettant d'effectuer la relation inverse).

LE PROJET –Migration

Doctrine-nous permet ce que l'on appelle les migrations. Celles-ci nous permettent de créer ou modifier la structure de notre base de données.



On peut créer un nouveau fichier de migration à l'aide de la commande : **Symfony console make:migration**

Lorsque nous avons créé nos Entités, il nous suffit de migrer notre entité en base de données, ce qui créera notre table en base de données avec le fichier généré lors de la création de notre entité. Dans ce fichier il y a tout le code SQL généré par Symfony avec les champs que nous avons préremplis.

```

Version20211016160619.php x
10  /**
11   * Auto-generated Migration: Please modify to your needs!
12   */
13  final class Version20211016160619 extends AbstractMigration
14  {
15      public function getDescription(): string
16      {
17          return '';
18      }
19
20      public function up(Schema $schema): void
21      {
22          // this up() migration is auto-generated, please modify it to your needs
23          $this->addSql('CREATE TABLE user_entreprise (user_id INT NOT NULL, entreprise_id INT NOT NULL, INDEX IDX_AA7E3C8CA76ED395 (user_id), INDEX IDX_AA7E3C8C8C4AEAFEA (entreprise_id), PRIMARY KEY(user_id, entreprise_id)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
24          $this->addSql('ALTER TABLE user_entreprise ADD CONSTRAINT FK_AA7E3C8CA76ED395 FOREIGN KEY (user_id) REFERENCES user (id) ON DELETE CASCADE');
25          $this->addSql('ALTER TABLE user_entreprise ADD CONSTRAINT FK_AA7E3C8CA4AEAFEA FOREIGN KEY (entreprise_id) REFERENCES entreprise (id) ON DELETE CASCADE');
26      }
27
28      public function down(Schema $schema): void
29      {
30          // this down() migration is auto-generated, please modify it to your needs
31          $this->addSql('DROP TABLE user_entreprise');
32      }
33  }

```

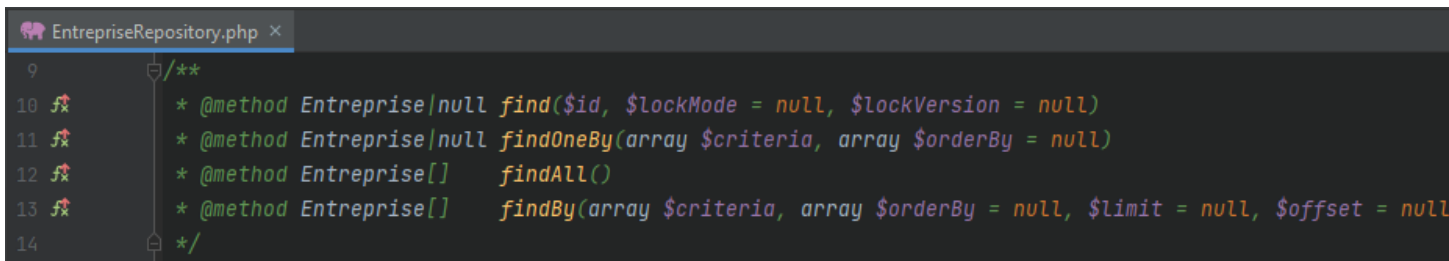
Pour migrer notre fichier il suffira d'exécuter la commande : **symfony console doctrine:migration:migrate**

En cas d'erreur il suffira de modifier directement le fichier de l'entité et recréer un fichier de migration puis l'exécuter.

LE PROJET – Le repository

A partir du moment où nous avons généré nos Entités nous avons créé notre repository qui nous servira à récupérer nos données dans notre base de données avec des requêtes pour ensuite les traiter dans le Controller.

Une fois que nous avons effectué nos migrations, celles-ci font la jonction avec la base de données. Symfony est très utile car cela va nous générer automatiquement des « méthodes » de base (find, findOneBy, findAll et findBy) qui seront utilisées automatiquement lors de la génération de notre contrôleur, nous en parlerons dans le chapitre des « controller »

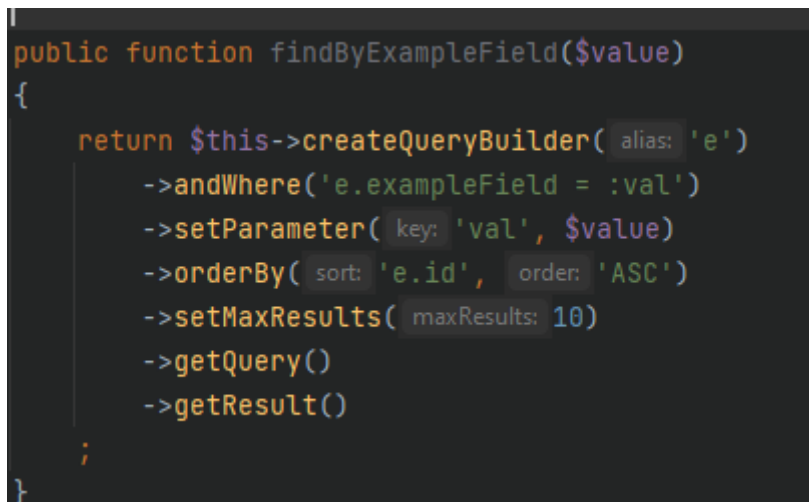


```

9      /**
10     * @method Entreprise|null find($id, $lockMode = null, $lockVersion = null)
11     * @method Entreprise|null findOneBy(array $criteria, array $orderBy = null)
12     * @method Entreprise[]   findAll()
13     * @method Entreprise[]   findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
14     */
  
```

Grace a ce repository nous allons pouvoir faire des requêtes en DQL (DoctrineQueryLanguage) qui s'apparente au SQL, à la différence près qu'il s'applique sur les entités et non sur la base de données. Nous pouvons retrouver toutes les syntaxes du DQL sur ce lien : <https://www.doctrine-project.org/projects/doctrine-orm/en/latest/reference/dql-doctrine-query-language.html>

Voici un exemple d'une requête en DQL généré automatiquement par Symfony lors d'une migration d'une entité.



```

public function findByExampleField($value)
{
    return $this->createQueryBuilder( alias: 'e')
        ->andWhere('e.exampleField = :val')
        ->setParameter( key: 'val', $value)
        ->orderBy( sort: 'e.id', order: 'ASC')
        ->setMaxResults( maxResults: 10)
        ->getQuery()
        ->getResult()
    ;
}
  
```

LE PROJET –Le repository

Voici un exemple d'une requête personnalisée en DQL qui concernera nos produits :

```

35
36 public function findByLibelleAsArray($term)
37 {
38     return $this->createQueryBuilder( alias: 'p')
39         ->select( select: 'p.id', 'p.libelle AS text', 'p.prix')
40         ->andWhere("p.libelle LIKE :term")
41         ->setParameter( key: 'term', value: "%".$term."%")
42         ->getQuery()
43         ->getResult()
44     ;
45 }
```

En SQL nous pourrions l'écrire de cette façon (ici « term » prendra la valeur par):

```

SELECT p.id, p.libelle, p.prix
FROM produits AS p
WHERE p.libelle LIKE '%par%'
```

```
var_dump($term);
```

Si nous effectuons un « **var dump** » avec notre variable « **\$term** » que nous avons dans notre requête, il nous suffit de le retourner en format JSON (*JavaScript Objet Notation*) en utilisant notre requête lors de la création d'un produit dans notre Controller cela nous retournerait dans notre vue **un id, un libelle, et un prix.**

Exemple ci-dessous :

```
{"id":130,"libelle":"parquet","prix":500}
```

LE PROJET – Les Controller

Les contrôleur sont ce que moi personnellement j'appelle le réceptacle des données, il est comme un intermédiaire entre le modèle et la vue. Grace au contrôleur nous allons demander au modèle les données, de les analyser et décider de comment nous souhaitons les renvoyer à la vue. *(Nous traiterons que des données en PHP bien entendu là-dedans.)*

Pour générer nos contrôleurs j'ai fait appel à Symfony, qui me simplifie la vie en général.

Commande utilisée :

```
symfony console make:crud
```

Symfony nous propose de générer entièrement de bout en bout un **CRUD** (*Create, Read, Update, Delete*), cela consiste à nous générer des fonctions dans notre contrôleur avec les routes associées puis de générer aussi des vues HTML pour accéder immédiatement à notre base de données et nous donner la possibilité des Créer, lire, mettre à jour, et donc aussi de supprimer des données dans notre base de données grâce aux fonctions créées dans notre contrôleur.

Fonction avec les routes, et les vues associée à nos contrôleur :

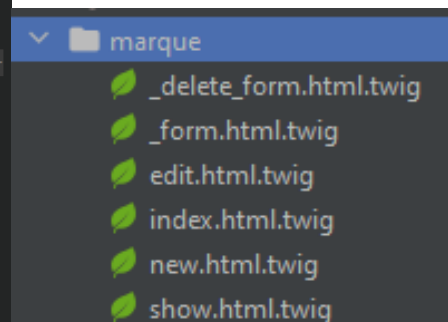
```
#[Route('/marque')]
class MarqueController extends AbstractController
{
    #[Route('/', name: 'marque_index', methods: ['GET'])]
    public function index(MarqueRepository $marqueRepository): Response{...}

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

    #[Route('/{id}', name: 'marque_show', methods: ['GET'])]
    public function show(Marque $marque): Response{...}

    #[Route('/{id}/edit', name: 'marque_edit', methods: ['GET', 'POST'])]
    public function edit(Request $request, Marque $marque): Response{...}

    #[Route('/{id}', name: 'marque_delete', methods: ['POST'])]
    public function delete(Request $request, Marque $marque): Response{...}
}
```



marque

- _delete_form.html.twig
- _form.html.twig
- edit.html.twig
- index.html.twig
- new.html.twig
- show.html.twig

LE PROJET - liste des compétences

Installation et configuration de Symfony 5

Installation et configuration Wamp

Installation et utilisation de dépendances sur Symfony :

Back-end :

- Doctrine
- Form
- Orm-Fixtures
- Maker

Front-End :

- Twig
- SCSS
- Bootstrap
- Fontawesome

Installation et configuration de librairies JavaScript

- Select2
- DataTables

PROJET – Environnement de développement (IDE)

Figure représentant le logo de PhpStorm



Pour le développement de l'application j'ai travaillé sur PhpStorm, logiciel de développement. De plus, j'avais déjà eu l'opportunité de travailler avec cet IDE lors de ma formation à l'ENI.

JetBrains PhpStorm est un IDE commercial et multiplateforme pour PHP construit sur la plate-forme IntelliJ IDEA de JetBrains. PhpStorm fournit un éditeur pour PHP, HTML et JavaScript avec analyse de code à la volée, prévention des erreurs et refactorisations automatisées pour le code PHP et JavaScript.

LE PROJET – Framework

Figure représentant le logo de Symfony

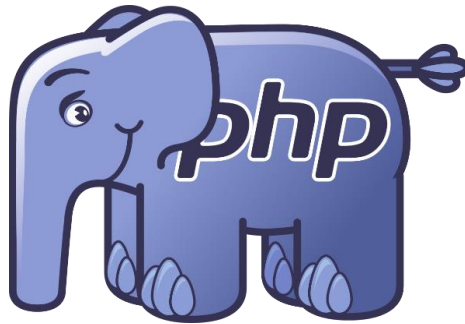


SYMFONY a été choisi pour supporter le projet, par le fait de sa modularité et de son développement en couche, il était idéal de faire ce choix. Lancé en 2005 par SensioLabs une agence web française, Symfony était à l'origine appelé Sensio Framework. Lorsque Sensio a souhaité partager son code avec la communauté, elle l'a renommé Symfony Framework, pour garder les initiales SF. Avec le passage à la version 2.0, l'outil est devenu simplement Symfony.

Pour ce qui concerne la modularité, Symfony 5 a été jusqu'au bout de la logique : chaque projet est découpé en modules (bundles), les plus précis possibles et le framework lui-même est un groupe de modules que chacun est libre d'utiliser ou non. Symfony est un kit de composants destinés à faciliter le développement de sites internet riches ou d'applications web. Pour cela, le code est séparé en trois couches selon le modèle MVC qui sépare le modèle de données (M), l'interface utilisateur ou vue (V) et le contrôleur (C) qui gère les événements, la synchronisation, etc...

LE PROJET – Langages

Figure représentant le logo de PHP



PHP8 : Hypertext Preprocessor, plus connu sous son sigle PHP, est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.

Figure représentant le logo de Javascript



JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les technologies HTML et CSS, JavaScript est parfois considéré comme l'une des technologies cœur du World Wide Web³. Une grande majorité des sites web l'utilisent⁴, et la majorité des navigateurs web disposent d'un moteur JavaScript⁵ dédié pour l'interpréter, indépendamment des considérations de sécurité qui peuvent se poser le cas échéant. C'est un langage orienté objet à prototype : les bases du langage et ses principales interfaces sont fournies par des objets.

LE PROJET – Langages et Template

Figure représentant le logo de HTML5



L'HyperText Markup Language, HTML, désigne un type de langage informatique descriptif. Il s'agit plus précisément d'un format de données utilisé dans l'univers d'Internet pour la mise en forme des pages Web.

Figure représentant le logo de TWIG



Twig est un moteur de gabarit développé en PHP inclus par défaut avec le framework Symfony 5. L'intérêt principal d'un moteur de gabarit est de séparer la logique de sa représentation. Il est capable de rendre notre code dynamique pour intégrer des pages web et nous permet de faire boucler des listes d'éléments, afficher une portion de code selon une condition ou encore formater une date en fonction de la date locale utilisée par un visiteur du site. Idéal si vous travaillez en équipe avec des intégrateurs, qui n'auront qu'à modifier les Template dans le répertoire views/ des bundles présents dans Symfony.

Figure représentant le logo de Bootstrap



Bootstrap est une collection d'outils utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web.

LE PROJET – Langages et Template

Figure représentant le logo de CSS



Les feuilles de styles, ou **Cascading Style Sheets** en anglais, sont connues sous l'abréviation de CSS. Ce langage permet de gérer la présentation des documents HTML, il est recommandé par le fameux W3C (World Wide Web Consortium), au même titre que le HTML ou le XML.

Le but de la CSS est de séparer la structure d'un document HTML de sa présentation : HTML sert à structurer le contenu, CSS sert à mettre en forme plus facilement un contenu structuré. La mise en forme s'applique grâce à des styles sur de nombreux éléments : le positionnement, la police, les couleurs, les bordures, les arrière-plans, les animations...

Figure représentant le logo de SASS



SASS (qui signifie Syntactically Awesome Style Sheets) est un pré-processeur pour le langage CSS. Il a été créé il y a quelques années par Hampton Catlin et Nathan Weizenbaum. Tout comme le langage LESS (qu'on a vu dans un cours précédent) il permet de générer dynamiquement du code CSS tout en offrant une syntaxe simple et un code facilement réutilisable et maintenable.

Nous utiliserons principalement le SCSS pour permettre d'utiliser des variables et d'avoir une meilleure structure du code, grâce à l'imbrication.

LE PROJET - Environnement de développement

Figure représentant le logo de CSS



WampServer est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant trois serveurs, un interpréteur de script, ainsi que phpMyAdmin pour l'administration Web des bases MySQL

Figure représentant le logo de CSS



MySQL est un serveur de bases de données relationnelles développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multi-thread et multi-utilisateur. Donc idéal pour l'application. Je m'appuierais sur l'interface PhpMyAdmin pour vérifier la bonne création de mes tables

LE PROJET - EN CONCLUSION

Ce stage a été une expérience professionnelle significative, car il m'a permis de déterminer la voie dans laquelle je voulais me spécialiser et m'a conforté dans mon choix de devenir développeur informatique.

Le seul regret que j'ai, c'est de ne pas avoir fait partie d'un service informatique et donc de ne pas avoir pu être immergé dans le métier. Cela aurait été un atout pour ma carrière.

Néanmoins, j'ai trouvé très enrichissant de pouvoir gérer un projet client dans sa totalité de l'analyse du besoin au développement de l'application.

J'ai apprécié la liberté que m'a laissé mon tuteur de stage dans la réalisation du projet. Cela m'a permis de mettre à profit les enseignements acquis à l'ENI mais également de développer mes connaissances par de la recherche personnelle et de les enrichir grâce aux échanges avec mon tuteur de stage.

À travers les missions qui m'ont été confiées, j'ai pu développer mon autonomie, mes connaissances, de nouvelles compétences et découvrir de nouvelles technologies réputées dans le domaine du web. Grâce à ces nouvelles compétences et cette expérience professionnelle en développement web, l'envie de découvrir davantage ce domaine et les multiples technologies a été confortée.

La formation BAC+2 développeur web et web mobile de l'ENI, qui a pour but initial de former des développeurs informatiques, a su m'apporter une base théorique et technique solide. Ce stage a permis de conforter et d'affiner mon orientation professionnelle vers le développement informatique.