

DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▶ VANDEPUTTE
Nom d'usage ▶ ROLLET
Prénom ▶ Mathieu
Adresse ▶ 5 av Médéric
92360 Meudon la Forêt

Titre professionnel visé

Développeur web et web mobile

MODALITE D'ACCES :

- ☒ Parcours de formation
- ☐ Validation des Acquis de l'Expérience (VAE)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.
Ce titre est délivré par le Ministère chargé de l'emploi.

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel (DP)** dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- ▶ pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- ▶ un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- ▶ une déclaration sur l'honneur à compléter et à signer ;
- ▶ des documents illustrant la pratique professionnelle du candidat (facultatif)
- ▶ des annexes, si nécessaire.

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.



Sommaire

Exemples de pratique professionnelle

Intitulé de l'activité-type n° 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité	p.	5
▶ Réaliser une interface utilisateur avec une solution de gestion de contenu	p.	5
▶ Réaliser une interface utilisateur web dynamique	p.	12

Intitulé de l'activité-type n° 2

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité	p.	
▶ Créer une base de données	p.	15
▶ Développer les composants d'accès aux données	p.	18
▶ Développer une partie back-end d'une application web	p.	21

Déclaration sur l'honneur	p.	26
Remerciements	p.	27

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

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

Exemple n°1 ► Réaliser une interface utilisateur avec une solution de gestion de contenu

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre de mon stage au sein de l'entreprise Redecso, l'une de mes principales missions était de donner la possibilité au directeur commercial de créer simplement des devis tout en respectant la logique particulière à l'entreprise.

Ma première difficulté a donc été de tenter de comprendre cette logique en échangeant avec ce dernier, afin de comprendre les étapes et les rouages de la mécanique entraînant la création d'un devis dans une entreprise du bâtiment.

J'ai ainsi pu apprendre qu'un devis n'était autre qu'une accumulation de tâches avec un prix associé, cependant, ce ne pouvait pas être aussi simple. En effet, l'entreprise a son propre système de fonctionnement, ainsi, une tâche peut éventuellement être classifiée afin de faciliter la visibilité dans les gros devis, mais ne le sera pas forcément. De plus, une tâche peut nécessiter un produit (chauffe-eau, menuiseries, etc...) et leur prix doit être référencé.

Il a donc fallu réfléchir au moyen de donner toutes ces possibilités à l'utilisateur tout en conservant une simplicité d'utilisation. La solution s'imposant pour cela aura été de gérer les flux d'information entre le front-end et le back-end à l'aide de requêtes asynchrones (AJAX) afin de rendre l'utilisation de l'application plus dynamique pour l'utilisateur et ainsi lui laisser plus de choix et d'options.

Nous donnions ainsi une part importante au javascript au sein de notre application, car les données devaient être envoyées et affichées sans avoir à recharger la page.

Une page de création de devis devait donc être composée de plusieurs parties.

La première partie serait donc un formulaire permettant de récupérer les informations nécessaires à la création d'un devis.

Quel est le Client / Entreprise ?

MARQUIS Aurélie ▼

Date de création :

jj/mm/aaaa 

Suivant

DOSSIER PROFESSIONNEL (DP)

Ainsi nous renseignons en premier le client ainsi que la date de création du devis, les deux premières informations requises, puis la validation de ce formulaire nous envoie sur un second, identique nous permettant de finaliser la création de notre devis, même si, pour le moment il ne contient aucune tâche ou catégorie.

Objet :

Ex: Intervention Plomberie

Référence :

Ex: Adresse, contacte, tel..

Suivant

Retour

Nous arrivons donc sur une page représentant l'aspect final d'un devis :



REDECSO

Mme Marquis Aurélie

89, boulevard d Alsace

78140 VÉLIZY-VILLACOUBLAY

OBJET : Réfection d'une salle de bains

REFERENCE : 3 rue de la crosse

Categorie	Description	Produit	Quantité	Prix
TOTAL HT :				
BASE TVA 20% :				
MONTANT TOTAL TTC :				

Choisissez l'action que vous souhaitez créer :

Une catégorie

Une Tache

RIB :

Code Banque : 10107 Code Guichet : 00651 Code BIC : BREDFRPPXXX

Numéro de compte : 00029026808 Clé : 55

Domiciliation : BRED PARIS SAINT FARGEAU

Numéro de Compte IBAN : FR76 1010 7006 5100 0290 2680 855

REDECSO 90 routes de Montreuil 93230 Romainville. SARL au Capital de 8000€.

N° Siret : 80107363700011 - Code APE : 4120A - N° TVA intracommunautaire : FR 47801073637

A partir de là, la page est découpée en plusieurs sections :

- L'entête, avec le logo de l'entreprise, le nom, l'adresse du client ainsi que l'objet et la référence du devis renseignés précédemment
- Un tableau ne comportant que l'entête et le pied avec un corps vide dans lequel on insèrera les

DOSSIER PROFESSIONNEL (DP)

taches ou les catégories selon le besoin.

- Un pied de page comportant les informations légales obligatoires pour un devis.

Cette mise en page a pour but de faire en sorte que l'utilisateur puisse observer le rendu au fur et à mesure de l'ajout de ses tâches, car l'affichage correspondra toujours au devis final.

A cette fin, le clic sur un des boutons « catégorie » ou « tâche » nous permet d'afficher un formulaire sur une ligne ayant déjà l'apparence d'une ligne sur le tableau :

Categorie	Description	Produit +	Quantité	Prix
<input type="text" value="Ex: Electricité"/>		<input type="button" value="Save"/>		
TOTAL HT :				
BASE TVA 20% :				
MONTANT TOTAL TTC :				

Categorie	Description	Produit +	Quantité	Prix
Electricité :				
	<input type="text" value="Ex: blabla"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="Prix"/> <input type="button" value="Save"/>
TOTAL HT :				
BASE TVA 20% :				
MONTANT TOTAL TTC :				

Ainsi, une tâche rangée dans une catégorie, après sa création, ressemblera en tout point à une ligne sur un devis que vous seriez susceptible de recevoir.

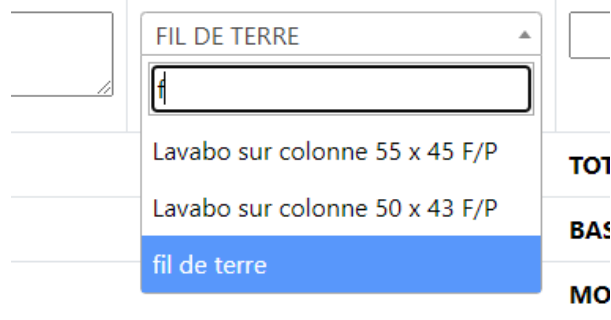
De plus, pour obtenir la totalité des options demandées par le client, la page ne comporte que 3 boutons de base :

- Le bouton « + » situé dans la ligne de titres du tableau à côté du titre « produit », permettant d'afficher la page de création d'un produit, qui apparaîtra dans une modale
- Les boutons catégorie et tâche, permettant de choisir si l'utilisateur veut créer une catégorie ou bien une tâche et faisant apparaître le formulaire correspondant.

L'utilisation et l'apprentissage de cet outil sont donc extrêmement simples, ce qui élargit un point capital des demandes du client.

DOSSIER PROFESSIONNEL (DP)

De plus, toujours dans un souci de simplicité et d'apporter du confort à l'utilisation de l'application, les produits sont renseignés sous le champs de texte et mis à jour selon la recherche effectuée :



Ainsi il suffit d'entrer quelques lettres pour trouver le produit recherché, et le sélectionner afin que le prix soit automatiquement ajouté dans la case correspondante, puis le total du devis calculé à la validation de la tâche. Le client pourra donc définitivement abandonner son bloc note et sa calculatrice.

2. Précisez les moyens utilisés :

Tout ceci a bien évidemment nécessité l'utilisation de nombreux moyens :

- **Html5 :**
Bien évidemment, travaillant sur une technologie web, il nous fallait utiliser le Html afin de gérer nos affichages

```
<table id="dataTable" class="table table-bordered table-striped text-white my-1">
  <thead>
    <tr>
      <th>Devis N° </th>
      <th>Nom</th>
      <th>Objets / mots clés</th>
      <th>Ref</th>
      <th>Date</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody>
    {% for devi in devis %}
      <tr>
        <td class="table-secondary">{{ devi.id }}</td>
        <td class="table-secondary">{{ devi.client.nom }} {{ devi.client.prenom }}</td>
```

- **Bootstrap :**
Afin de pouvoir obtenir un affichage uni et de garder une charte graphique simple, sobre et professionnelle, l'outil le plus adapté semblait bootstrap, nous pouvons y voir quelques classes dans le code cité précédemment.
- **Twig :**
Afin de pouvoir profiter de la puissance des templates et gérer nos affichages plus simplement, nous avons utilisé le langage de templating twig, fourni par Symfony. Notre menu, par exemple, ne sera donc pas écrit sur chaque page, mais dans un seul fichier qui sera lui-même

appelé. Bien sûr c'est une version simpliste de la chose, car nous pouvons ainsi appeler différents morceaux de page selon l'url ou bien encore les droits de la personne connectée.

```
1  {% extends '/devis/devis_layout.html.twig' %}
2
3  {% block tbody %}
4      <tr id="js-form-categorie" style="...">
5          {{ include('categorie/_form.html.twig') }}
6      </tr>
7      <tr id="js-form-tache" style="...">
8          {{ include('tache/_form.html.twig') }}
9      </tr>
10 {% endblock %}
```

Nous pouvions ainsi voir ci-dessus l'intégralité du fichier html appelé pour la page de création d'un devis. En fait, chaque module de cette page est situé dans un fichier séparé afin de faciliter le maintien de cette charte graphique, car la modification d'un fichier entrainera le changement sur l'intégralité de l'application si besoin.

- Select2 :

Un plugin permettant l'auto-complétion des formulaires, notamment pour les produits cités dans cette section. Ce plugin, basé sur la bibliothèque javascript JQuery, permet d'effectuer des requêtes en ajax à l'aide d'un paramétrage relativement simple :

```
$(".choice-search-produit").select2({
  ajax : {
    url : '/produit/search',
    data: function(params) {
      let query = {
        term: params.term,
      }
      return query;
    },
    dataType: 'json',
    processResults: function (data) {
      return {
        results: $.map(data, function (item) {
          console.log(item)
          return {
            text: item.text,
            id: item.id,
            prix: item.prix
          }
        })
      }
    }
  };
});
```

Nous pouvons voir ainsi que nous envoyons les termes tapés dans le champs à l'url '/produit/search' et que l'on récupère l'id, le libelle et le prix du produit sélectionné, le tout en une dizaine de ligne.

- JQuery

Bien que de moins en moins utilisé, le plugin JQuery était nécessaire pour le fonctionnement de select2, donc une fois installé, autant l'utiliser, d'autant qu'il simplifie l'écriture du code, notamment lors des interactions avec le DOM

```
let js_reference_div = $('#js-reference_div');
let js_reference_inline = $(document.createElement( 'div' ));
js_reference_inline.addClass('d-flex pl-5')
    .addClass('col-5');

if(!response.client.entreprise) {
    span_prenom.text(response.client.civilite + ' ' + response.client.nom + ' ' + response.client.prenom);
} else {
    span_prenom.text(response.client.nom.toUpperCase())
}
client_div.append(span_prenom);
span_adresse.text(response.client.adresse);
adresse_div.append(span_adresse);
span_cp.text(response.client.codePostal);
span_ville.text(response.client.ville);
cp_ville_inline.append(span_cp)
    .append(span_ville);
cp_ville_div.append(cp_ville_inline);
js_reference_div.html('<span> <strong> REFERENCE : </strong> ' + response.ref + '</span>');
js_objet_div.html('<span> <strong> OBJET : </strong> ' + response.objet + '</span>');
})
$('#js-form-etape-2').show();
})
```

Sur cet extrait nous pouvons voir que nous créons dynamiquement l'entête de la page de devis en javascript puis que nous insérons chaque chose à sa place grace aux sélecteurs fournis par JQuery.

3. Avec qui avez-vous travaillé ?

Sur ce projet j'ai travaillé seul, encadré cependant par mon tuteur de stage qui m'aiguillait vers les bons outils et les bonnes pratiques à mettre en place, mais l'utilisation de ces outils a nécessité énormément de recherches et de temps sur leur documentation.

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► *Redecso*

Chantier, atelier, service ► *Service Commercial*

Période d'exercice ► Du : *13/09/2021* au : *05/11/2021*

5. Informations complémentaires (facultatif)

Cette partie a été la plus complexe à mettre en place et est loin d'être aboutie, j'en suis conscient mais suis tout de même content d'avoir réussi à obtenir le résultat actuel car cela m'a fait énormément évoluer, tant au niveau technique qu'au niveau de la structuration des événements et de l'affichage des données.

Activité-type 1

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

Exemple n°2 ► Réaliser une interface utilisateur web dynamique

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Une fois la possibilité de créer un devis terminée (bien que l'affichage soit à revoir, chaque fonctionnalité est belle et bien fonctionnelle), il nous fallait maintenant pouvoir les afficher afin de pouvoir les consulter à l'avenir.

La création étant créée de telle manière que les devis créés ressemblaient à leur affichage final, nous avons pu réutiliser la majorité des parties de vues déjà créées pour la création afin d'obtenir un rendu final ressemblant à un devis papier classique :

4 Marquis
89, boulevard d'Alsace
78140

Réfection d'une salle de bains
3 rue de la crosse

Categorie	Description	Produit	Quantité	Prix
Electricité :	Relier les prises à la terre	fil de terre	1.2	
TOTAL HT :				
BASE TVA 20% :				
MONTANT TOTAL TTC :				

RIB :
Code Banque : 10107 Code Guichet : 00651 Code BIC : BREDFRPPXXX
Numéro de compte : 00029026808 Clé : 55
Domiciliation : BRED PARIS SAINT FARGEAU
Numéro de Compte IBAN : FR76 1010 7006 5100 0290 2680 855

REDECSO 90 routes de Montreuil 93230 Romainville, SARL au Capital de 8000€.
N° Siret : 80197363700011 - Code APE : 4120A - N° TVA Intracommunautaire : FR 47801973637
Direction Commercial: Vandeputte Hervé 06.09.22.22.26 / Direction Technique: Haddadi Tahar 06.28.23.92.77
Adresse Mail : redecso@outlook.fr

Ainsi nous retrouvons ici l'entête, le tableau ainsi que le pied de page de la partie création, à l'exception des boutons, cette page n'étant destinée qu'à la consultation d'un devis.

2. Précisez les moyens utilisés :

De la même manière que les vues ont pu être réutilisées car elles étaient semblables à celles de la création, les technologies sont restées les mêmes.

La réutilisation des vues a d'ailleurs été rendue possible grâce à twig, car nous avons pu rappeler les diverses parties nécessaires de la même manière que pour la création :

```
1  {% extends '/devis/devis_layout.html.twig' %}
2
3  {% block javascripts %}
4      {{ parent() }}
5      {{ encore_entry_script_tags('devis_show') }}
6  {% endblock %}
7  {% block tbody %}
8
9      <div id="js-info-client">
10         <div id="scss-info-client">
11             <div id="js-client_div" class="d-flex justify-content-end w-100">{{ devi.client }}</div>
12             <div id="js-adresse_div" class="d-flex justify-content-end w-100">{{ devi.client.adresse }}</div>
13             <div id="js-cp_ville_div" class="d-flex justify-content-end w-100">{{ devi.client.codePostal }}</div>
14         </div>
15         <div id="">
16             <div id="js-objet_div" class="d-flex justify-content-start w-100 ">{{ devi.objet }}</div>
17             <div id="js-reference_div" class="d-flex justify-content-start w-100"> {{ devi.ref }}</div>
18         </div>
19     </div>
20
21
22     {% for categorie in devi.categories %}
23     <tr>
24         <td>{{ categorie.libelle }} :</td>
25         {% for taches in categorie.taches %}
26         <tr>
27             <td></td>
28             <td>{{ taches.libelle }}</td>
29             <td>{{ taches.produit.libelle }}</td>
30             <td></td>
31             <td>{{ taches.produit.prix }}</td>
32         </tr>
33         {% endfor %}
34     </tr>
35     {% endfor %}
36
37
38 {% endblock %}
39
```

Nous pouvons également voir ici que nous redessignons l'entête, car, rappelez vous, lors de la création, elle était créée dynamiquement en js (dans le futur, il semble judicieux de créer une vue à part qui pourra être appelée lors de la création comme lors de l'affichage).

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

De même que pour l'exemple précédent, j'ai travaillé seul sous la supervision de mon tuteur de stage.

4. Contexte

Nom de l'entreprise, organisme ou association ► *Redesco*

Chantier, atelier, service ► *Service Commercial*

Période d'exercice ► Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires (facultatif)

Cette partie m'a permis de me familiariser avec les boucles en twig, car je n'avais pas encore eu l'occasion de les pratiquer, simplifiant ainsi grandement mon code, améliorant sa lisibilité et sa compréhension.

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

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

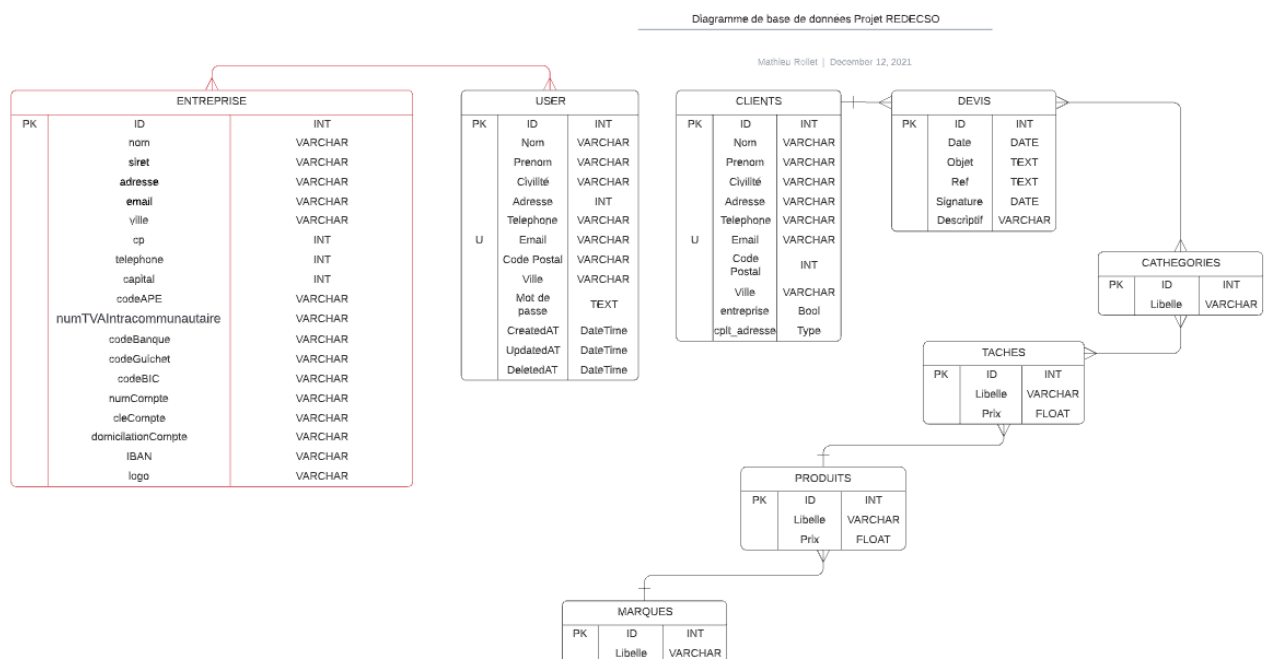
Exemple n° 1 ► Créer une base de données

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Le client a exprimé plusieurs besoin lors de nos conversations.

Tout d'abord celui de pouvoir avoir un fichier client lui permettant de retrouver facilement des devis passés, pouvoir avoir une base de produits également afin de pouvoir les sélectionner lors de la création d'un devis, ainsi que de pouvoir sélectionner les catégories les plus fréquentes.

A partir de ces informations, on a pu en tirer un schéma permettant à la fois à l'application d'être simple tout en pouvant évoluer par la suite.



Nous pouvons donc voir ici que nous avons 9 tables qui composeront notre base de données, plus 4 autres pour servir de tables intermédiaires aux relations de type many to many.

Une fois cette structure modélisée, nous pouvons configurer notre application afin de lui indiquer où sera notre base de données.

Pour cela, nous avons dû éditer le fichier .env se situant à la racine de notre projet et décrivant les

différentes spécificités de notre environnement, y compris la localisation et les informations de connexion à notre base de donnée.

```
DATABASE_URL="mysql://redecso@127.0.0.1:3307/redecso?serverVersion=10.4.13-MariaDB"
```

On peut voir ici que nous nous connectons à l'adresse 127.0.0.1 soit en local, avec l'utilisateur redcso et sans mot de passe (car en local)

Nous voyons la version du serveur mySql appelé, ici MariaDb, version open source de MySql.

Une fois cela fait, nous pouvons créer notre base de données à l'aide de doctrine à l'aide de la commande `symfony console doctrine :database :create`

Cette commande nous créera notre base de données comme indiqué dans le fichier .env, donc ici redcso.

Une fois cela fait, nous générons des entités, chaque entité représentant une table de notre base de donnée.

Une entité est une classe php décrivant la structure qu'auront les objets issus d'une requête. Ils auront donc pour propriétés les différents champs de la base et comme methode les accesseurs permettant d'accéder et/ou de modifier ces champs.

De plus, ces méthodes nous permettront d'accéder plus simplement aux jointures, mais nous aborderons cela plus tard.

2. Précisez les moyens utilisés :

Pour la création de la base de données, nous avons tout d'abord du réfléchir à la structure, pour cela nous avons beaucoup schématisé avant de créer le rendu final via LucidChart, une application de dessein et de diagramme permettant de dessiner simplement ce genre de schéma.

Nous avons ensuite utilisé certains outils fournis par Doctrine et Symfony afin de pouvoir gérer notre base de donnée depuis notre application directement.

- Doctrine :
Un ORM qui est souvent associé à symfony, bien que pas forcément indissociable, et qui permet, au même titre que son homologue Eloquent de chez Laravel de gérer les bases de données à l'aide de lignes de commandes directement depuis son application et nous fournit des méthodes afin de simplifier et sécuriser nos actions les plus courantes, Doctrine centralisera la totalité de la gestion de la base de données dans le code php, rendant l'utilisation d'un logiciel de gestion de base de données (SGBD) tel que phpMyAdmin ou Heidi quasiment obsolète.

DOSSIER PROFESSIONNEL (DP)

- **Symfony :**
Un framework français php basé sur le paradigme MVC très utilisé par les entreprises. Il est souvent associé à plusieurs outils comme twig pour le templating, yaml pour le format de ses fichiers de configuration ou encore Doctrine cité ci-dessus pour la gestion de la base de données.

3. Avec qui avez-vous travaillé ?

Pour cette partie j'ai travaillé en collaboration avec mon tuteur de stage et le directeur commercial, mon expérience ne permettant pas encore d'extraire un schéma complet depuis un cahier des charges ou une conversation, y participer et pouvoir y assister m'ont cependant appris énormément sur ce sujet.

4. Contexte

Nom de l'entreprise, organisme ou association ► *Redecso*

Chantier, atelier, service ► *Service Commercial*

Période d'exercice ► Du : *13/09/2021* au : *05/11/2021*

Activité-type 2

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

Exemple n° 2 ► Développer les composants d'accès aux données

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Concernant l'accès aux données, j'ai à nouveau utilisé les outils qui m'étaient fournis par Symfony et Doctrine, à savoir une liste de méthodes prédéfinies et un Repository, fichier servant à stocker les requêtes en dql afin de pouvoir les appeler via l'entity.

Ainsi, un Repository ressemblera à cela :

```
/**
 * @method Devis|null find($id, $lockMode = null, $lockVersion = null)
 * @method Devis|null findOneBy(array $criteria, array $orderBy = null)
 * @method Devis[]    findAll()
 * @method Devis[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class DevisRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Devis::class);
    }

    public function findAllAsArray(): array
    {
        return $this->createQueryBuilder('d')
            ->orderBy('d.id', 'ASC')
            ->getQuery()
            ->getArrayResult()
        ;
    }
}
```

On peut y voir les 4 méthodes prédéfinies en premier plan ainsi qu'une méthode comportant une requête en dql en second plan.

Cette requête pourrait être écrite de la manière suivante en sql :

```
SELECT *
FROM devis AS d
ORDER BY d.id ASC
```

Pourquoi utiliser le dql dans ce cas ?

Nous avons la possibilité de passer des paramètres à ces requêtes, on appelle cela du data-binding. Ce procédé permet de sécuriser les données avant de les injecter dans le sql en lui-même afin d'éviter les failles de type xhr ou encore sql.

```
public function findOneAsArray($id)
{
    return $this->createQueryBuilder( alias: 'd')

        ->andWhere('d.id = :id')
        ->setParameter( key: 'id', $id)
        ->orderBy( sort: 'd.id', order: 'ASC')
        ->getQuery()
        ->getArrayResult()[0]
    ;
}
```

Ici on voit que l'id est inséré en deux fois. Tout d'abord on renseigne que l'id doit être inséré plus tard en l'indiquant « :id »

Ensuite on utilise la méthode setParameter() pour indiquer que « :id » prendra la valeur de l'id après sécurisation des données.

Cette méthode rend très simple la sécurisation des données ainsi que leur accès à l'aide d'un simple appel à la méthode depuis le controller :

```
$devis_array = $devisRepository->findOneAsArray($devis->getId());
```

Les deux requêtes présentées servent à récupérer un ou tous les devis sous forme de tableau, n'ayant pas trouvé le moyen d'encoder une collection fournie par une requête Doctrine en json, c'est le seul moyen que l'on m'ait indiqué.

2. Précisez les moyens utilisés :

Ici j'ai réutilisé les outils précédents, à savoir Symfony et Doctrine, cette fois en utilisant principalement le Repository, fichier regroupant les requêtes permettant de récupérer ou insérer des données.

3. Avec qui avez-vous travaillé ?

Toujours indépendamment, j'ai eu plusieurs occasions d'interroger mon tuteur lorsque je restais coincé sur un problème, dans tous les cas cependant c'était après avoir passé beaucoup de temps à rechercher.

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► *Redecso*

Chantier, atelier, service ► *Service Commercial*

Période d'exercice ► Du : *13/09/2021* au : *05/11/2021*

5. Informations complémentaires (facultatif)

Conscient que ces automatismes de sécurité sont un atout dans la rapidité de développement, j'aimerais m'intéresser à ce sujet plus en profondeur par la suite, ce point me paraissant important.

Activité-type 2

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

Exemple n° 3 ► Développer une partie back-end d'une application web

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

La première chose à faire en back-end a été de sécuriser les routes.

En effet, l'application étant destinée à une usage interne exclusivement, il fallait sécuriser l'intégralité des routes à l'aides des middlewares fournis par Symfony/Auth, un package installé via composer.

Ce package automatise lui aussi de nombreuses mécaniques telles que le hashage des mots de passes pour les utilisateurs, la création de middleware afin de sécuriser les routes aussi, ainsi que des fichiers de configuration pour les gérer, mais également les vues, les formulaires et les controllers pour exploiter le système d'authentification.

Nous avons commencé par nous intéresser au fichier de configuration, afin d'indiquer que nous voulions sécuriser l'intégralité de l'application par un middleware d'authentification à l'exception de la route permettant de se connecter, bien entendu.

```
# 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 }
```

On peut donc voir que la protection des routes dans notre cas a été simple.

Il a ensuite fallu fournir quelques routes de redirection dans les cas d'une authentification réussie par exemple. Ca se passait dans le fichier Authenticator créé par le package :

```
public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $firewallName)) {
        return new RedirectResponse($targetPath);
    }

    return new RedirectResponse($this->urlGenerator->generate( name: 'home' ));
}
```

Ainsi on décidait qu'en cas de succès, on était automatiquement redirigé vers la page d'accueil.

Il ne nous restait donc plus qu'à créer ces routes et leurs actions.

Comme nous l'avons vu précédemment, l'accès aux données a déjà été géré, ainsi que la partie affichage, il ne nous reste donc plus qu'à joindre les deux afin d'avoir une application web dynamique et sécurisée.

Cette jointure se déroule dans ce que l'on appelle les Controllers. Ces fichiers contiennent la logique métier de l'application, ce sont donc eux qui choisiront quelles données récupérer et comment les formater avant de les envoyer au client.

Dans la majorité des cas, ces controllers possèdent des méthodes semblables, permettant de créer ce que l'on appelle un CRUD (Create, Read, Update, Delete) qui permet les actions de base sur la base de données.

Symfony nous donne encore une fois un outil pour générer automatiquement ce controller avec les méthodes de base pour faire fonctionner les CRUD :

- Index

Méthode accessible par l'url <préfixe>/ via la méthode GET.

Par défaut elle permet de récupérer l'intégralité des entités liées au controller

```
#[Route('/', name: 'user_index', methods: ['GET', 'POST'])]
public function index(UserRepository $userRepository): Response
{
    return $this->render(view: 'user/index.html.twig', [
        'users' => $userRepository->findAll(),
    ]);
}
```

- New

Méthode accessible par l'url <préfixe>/new en méthode GET ou POST

Par défaut, cette méthode, si elle est appelée en méthode GET, renvoie le formulaire de création d'une nouvelle entité liée au controller. En méthode POST, elle traitera le formulaire et insèrera l'entité en base de données après validation des données.

```
#[Route('/new', name: 'user_new', methods: ['GET', 'POST'])]
public function new(Request $request): Response
{
    $user = new User();
    $form = $this->createForm(type: UserType::class, $user);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->persist($user);
        $entityManager->flush();

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

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

DOSSIER PROFESSIONNEL (DP)

- Show

Méthode accessible par l'url <préfixe>/<id> en méthode GET, {id} étant une variable de type int permettant d'aller rechercher automatiquement (via auto-wiring) l'entité correspondante.

```
#[Route('/{id}', name: 'user_show', methods: ['GET'])]
public function show(User $user): Response
{
    dd($user);
    return $this->render(view: 'user/show.html.twig', [
        'user' => $user,
    ]);
}
```

- Edit

La méthode edit() est semblable à la méthode new, elle est cependant accessible en méthode PUT grâce à l'url suivante <préfixe>/<id>/edit, {id} étant l'id de l'entité à éditer

- Delete

La dernière méthode est la méthode delete(), elle est appellable en méthode DELETE via l'url <préfixe>/<i>, ou {id} est l'id de l'entité à supprimer.

2. Précisez les moyens utilisés :

Une fois encore les outils fournis par Symfony nous ont été d'une grande aide. A l'aide de quelques commandes nous avons pu générer une quantité phénoménale de routines, de mécaniques et de fichiers.

μCependant, encore une fois, derrière ce plaisir d'effectuer des actions complexes en peu de temps, il reste cette frustration de ne pas avoir ce qu'il se passe exactement, comme par exemple ce qu'est un middleware, et le temps ne m'a pas permis d'aller chercher ce genre de réponse par exemple, et tant d'autres restent à chercher.

En me renseignant sur les packages de sécurité disponible et leur configuration, j'ai pu également rencontrer de nombreuses manière d'authentifier un utilisateur, sans savoir trop laquelle choisir, j'ai demandé de l'aide à mon tuteur pour faire ce choix qui était primordial.

3. Avec qui avez-vous travaillé ?

J'ai eu la possibilité encore une fois d'agir avec une grande autonomie, me permettant de trouver et de découvrir les outils à ma disposition.

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► *Redecso*

Chantier, atelier, service ► *Service commercial*

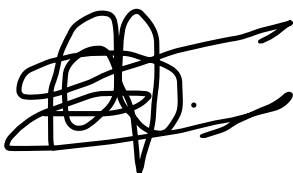
Période d'exercice ► Du : *13/09/2021* au : *05/11/2021*

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] Mathieu Vandepute,
déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis
l'auteur(e) des réalisations jointes.

Fait à Meudon la foret le 12/12/2021
pour faire valoir ce que de droit.

Signature :



REMERCIEMENT

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).

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.
