



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

<i>Nom de naissance</i>	▸ ALOIS
<i>Nom d'usage</i>	▸ ALOIS
<i>Prénom</i>	▸ Kévin
<i>Adresse</i>	▸ 6 rue des Verchères, 69360 Sérézin-du-Rhône

## Titre professionnel visé

**Titre Professionnel**  
**Développeur Web et Web mobile**

### MODALITÉ D'ACCÈS :

- ☒ 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.*

► <http://travail-emploi.gouv.fr/titres-professionnels>

# Sommaire

---

## Exemples de pratique professionnelle

<b>Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité</b>	<b>p.</b>	<b>5</b>
▸ Maquetter d'une application	p.	5
▸ Réaliser une interface utilisateur web statique et adaptable	p.	7
▸ Développer une interface utilisateur dynamique	p.	16
<b>Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité</b>	<b>p.</b>	<b>22</b>
▸ Créer une base de données	p.	22
▸ Développer les composants d'accès aux données	p.	28
▸ Développer la partie back-end d'une application web ou web mobile	p.	38
<b>Titres, diplômes, CQP, attestations de formation</b> <i>(facultatif)</i>	<b>p.</b>	<b>43</b>
<b>Déclaration sur l'honneur</b>	<b>p.</b>	<b>44</b>
<b>Documents illustrant la pratique professionnelle</b> <i>(facultatif)</i>	<b>p.</b>	<b>45</b>
<b>Annexes</b> <i>(Si le RC le prévoit)</i>	<b>p.</b>	<b>46</b>

# **EXEMPLES DE PRATIQUE PROFESSIONNELLE**

## Activité-type 1 en intégrant les recommandations de sécurité

Exemple n°1 • Maquetter une application

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

Une des premières étapes d'un projet est la conception de prototypes (wireframe) pour se rendre compte de plusieurs aspects comme, l'utilisation du site ou de l'application, des contraintes esthétiques et ergonomiques, de l'expérience utilisateur.

A ce niveau, les prototypes sont très importants, pour nous mais aussi pour le client.

Les prototypes (ou wireframe) que je compte vous présenter concerne la conception de mon site personnel afin de me présenter plus facilement à de futurs employeurs et accroître ma visibilité sur internet.

J'ai pensé ce site sur une seule et unique page avec la possibilité d'aller dans n'importe quelle rubrique via le menu qui reste affiché en haut de page pour nous accompagner durant la navigation.

Voir fiche Annexe 1, page 45.

Accueil :

- Un header avec une barre de navigation ainsi que le titre du site.
- Photo.
- Quelques liens vers les réseaux sociaux.

Dev :

- Explication de ma reconversion.
- L'école.
- Mes nouveaux acquis.

Portfolio :

- Ensemble des projets effectués.
- Cliquable pour avoir plus de détails.

Expériences :

- Expériences de ma vie professionnelle.

Formation:

- Formation antérieure à ma reconversion.
- Diplômes.

Loisirs :

- centre d'intérêt

Contact :

- Formulaire de contact

## 2. Précisez les moyens utilisés :

D'une part, j'utilise le papier et le crayon pour les grandes lignes, d'autre part, j'utilise un logiciel de maquette pour réaliser les différents wireframes pour chaque page du site pour le format site web mais aussi mobile ou tablette.

## 3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé seul durant l'évaluation et à la fin du premier mois de spécialisation.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ O'Clock

Chantier, atelier, service ▶ FAQ-O-Clock

Période d'exercice ▶ Du : 20/09/2019 au : 22/09/2019

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

**Activité-type 1**

*Exemple n° 2* • Réaliser une interface utilisateur web statique et adaptable

---

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

Aujourd'hui, l'information est utilisée sur plusieurs tailles d'écran mais principalement sur des écrans d'ordinateurs (au-dessus de 800px), des smartphones (en dessous de 500px), des tablettes (entre 500px et 800px) et d'autres types d'écran arrivent sur le marché comme les montres connectées, les smartphones pliables, les écrans sur les meubles de nos lieux de vie.

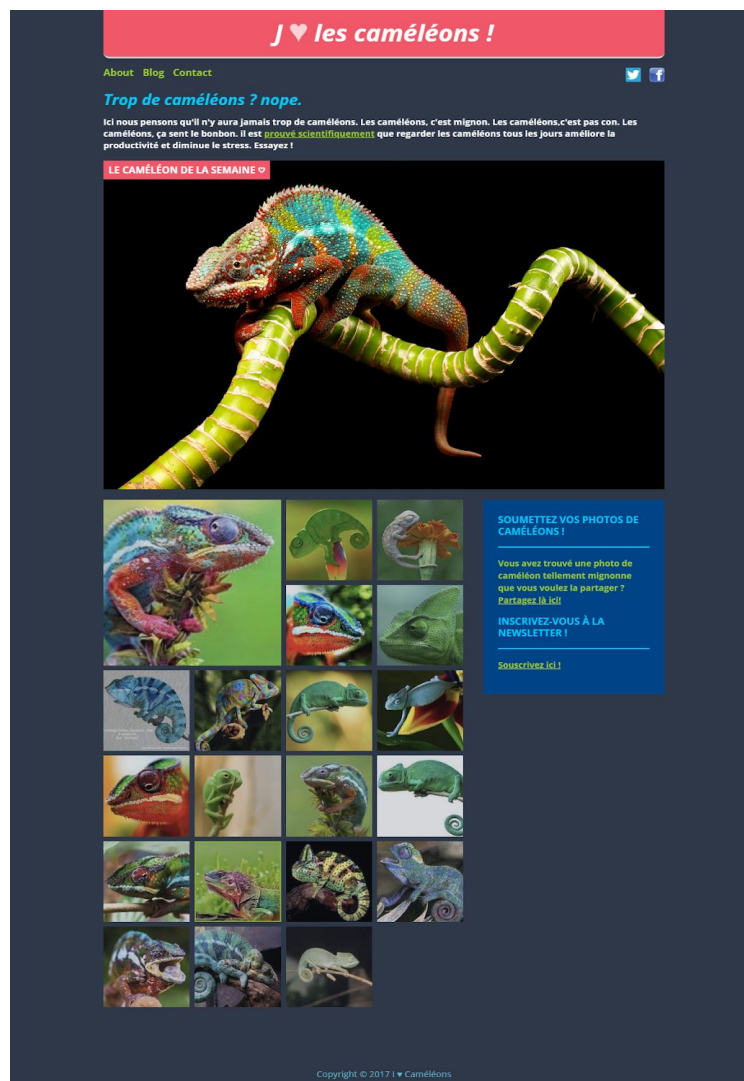
C'est pour cela, qu'aujourd'hui nous sommes pratiquement dans l'obligation de créer des sites adaptables à tous les supports.

Une solution : "Responsive".

Le responsive permet la modification des paramètres visuels suivant la taille de l'écran que l'on utilise. C'est au développeur de mettre en place ces fonctionnalités dans le css.

Pour ces projets, j'ai utilisé différentes solutions sachant que les 2 sites sont pensés principalement pour l'utilisation sur ordinateur et donc grand écran.

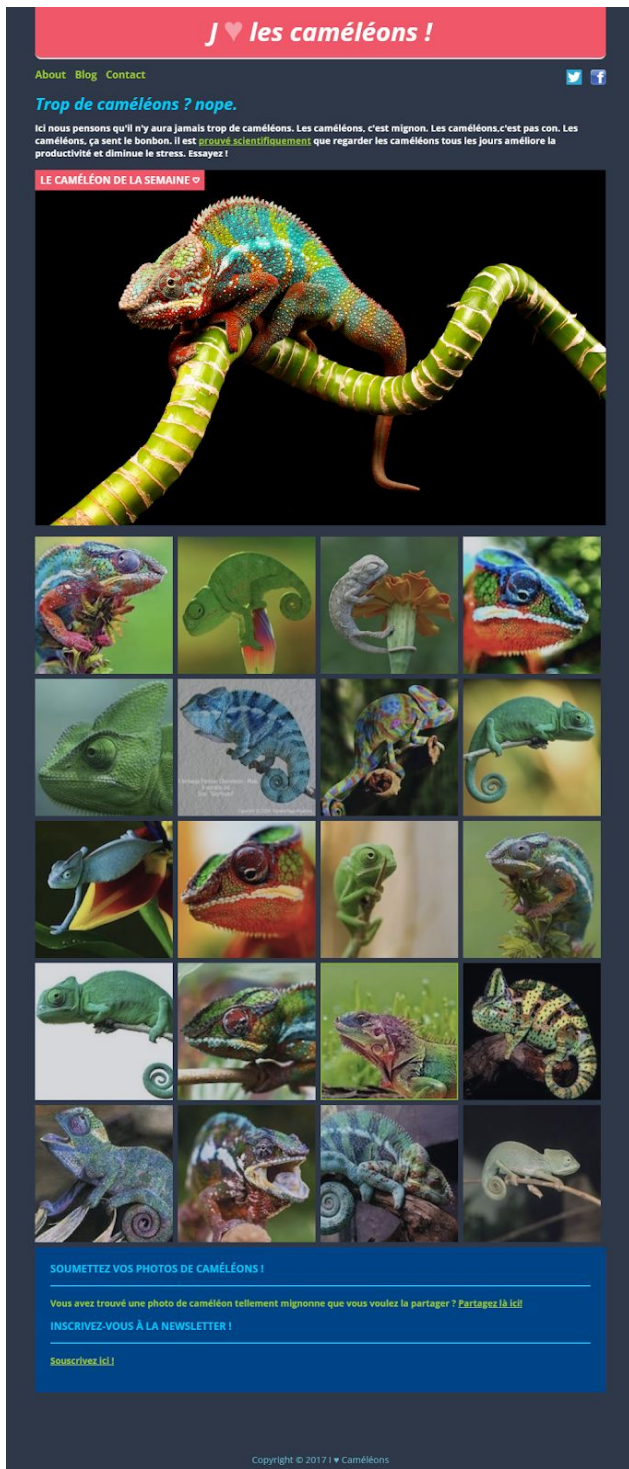
Exercice css avec des caméléons :



Voici la page principale correspondant à une taille d'écran de 1200 pixels. On peut remarquer que plusieurs photos de caméléons sont côte à côte, ainsi qu'un encadré d'informations.



Passons à une taille d'écran de 992px (taille d'un iPad pro) :



Pour cette page de taille 992 px on peut remarquer que les caméléons sont comme pour les 1200 px, 4 sur chaque ligne mais que l'encadré est passé en dessous. Dans le code ci-dessous on lui donne une valeur max (de 992 px pour notre exemple) et ensuite on lui donne le comportement que l'on attend à cette taille. Dans notre cas on utilise principalement des pourcentages (ou des «en») qui gardera la proportion que l'on souhaite avoir pour cette taille.

Dans notre cas 25% (1/4 de la page) moins 8 px, pour la marge, donc on aura au total 4 caméléons par ligne.

Pour le bloc info on lui demande 100% de la largeur donc la «div» se met automatiquement sur la base de la page.

/\* Si la taille de l'écran est inférieur ou égale à 992px de large, on applique les règles entre les accolades \*/

```
@media (max-width: 992px) {  
  .gallery {  
    width: 100%;  
  }  
  .gallery_thumbnail,  
  .gallery_thumbnail:first-child {  
    width: calc(25% - 8px);  
  }  
  .community {  
    width: 100%;  
  }  
}
```

Une taille d'écran de 768px (taille d'une tablette) :



Pour cette page de taille 768 px on peut remarquer que les caméléons changent de dispositions, 3 sur chaque ligne mais que l'encadré a disparu. Dans le code ci-dessous on lui donne une valeur max (de 768 px pour notre exemple) et ensuite on lui donne le comportement que l'on attend à cette taille.

Dans notre cas 33.3% (1/3 de la page) moins 8 px (pour la marge), donc on aura au total 3 caméléons par ligne.

Pour le bloc info, on demande à le faire disparaître avec la commande `< display : none >`.

```
/* On définit un nouveau breakpoint pour des règles qui
s'appliquent quand la largeur du viewport est inférieure
ou égale à 768px */
```

```
@media (max-width: 768px) {
  /* Les photos sont en trois colonnes et non plus en 4 */
  .gallery_thumbnail,
  .gallery_thumbnail:first-child {
    width: calc(33.3% - 8px);
  }
  /* On cache le aside en deçà de 768px de Large */
  .community {
    display: none;
  }
}
```

Une taille d'écran de 420px (taille d'un iPhone 8 plus) :



Pour cet exemple, nous avons utilisé un écran de 420px (un gros smartphone). Pour profiter pleinement de l'écran du téléphone et de la totalité des belles images de caméléons sur la largeur de l'écran, nous utilisons donc une largeur de 100% (1/1) moins les 8px de marge. Ce qui nous donne un caméléon par ligne.

Pour notre bloc informations comme pour l'exemple précédent nous l'envoyons par choix..

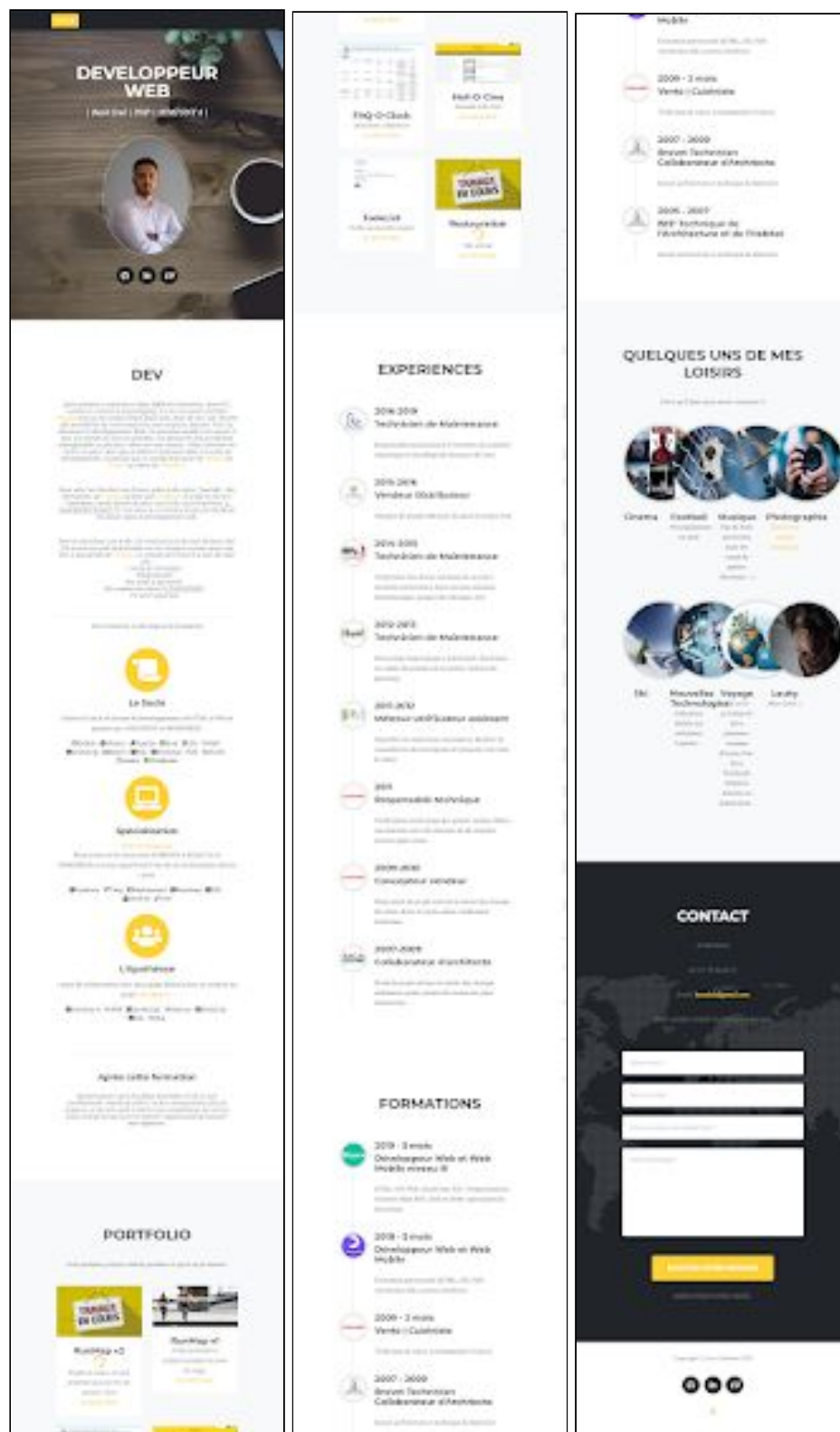
```
/* On défini un nouveau breakpoint pour des règles qui
s'appliquent quand la largeur du viewport est inférieure
ou égale à 420px */

@media (max-width: 420px) {
  /* Les photos sont en trois colonnes et non plus en 4 */
  .gallery_thumbnail,
  .gallery_thumbnail:first-child {
    width: calc(100% - 8px);
  }
  /* On cache le aside en deça de 768px de Large */
  .community {
    display: none;
  }
}
```





Taille pour tablette 767px :



## 2. Précisez les moyens utilisés :

J'ai eu l'occasion d'utiliser plusieurs solutions comme la méthode personnalisée avec un css réalisé par mes soins, mais aussi avec Bootstrap qui est la méthode que je préfère car plus rapide pour exécuter un projet.

## 3. Avec qui avez-vous travaillé ?

Travail effectué seul lors d'un exercice le soir pour les caméléons ainsi que pour mon CV.  
Après plusieurs entretiens, je me suis rendu compte que le CV personnel, site vitrine, était très important. Ce site a été mon premier projet après la formation.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶	O'Clock / CV
Chantier, atelier, service ▶	Formation / Projet personnel
Période d'exercice ▶	11/06/2019   11/11/2019 au 19/11/2019

## Activité-type 1 en intégrant les recommandations de sécurité

Exemple n° 3 • Réaliser une interface utilisateur web dynamique

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

Durant la formation nous avons appris à rendre notre code et notre interface utilisateur dynamiques afin d'avoir une expérience de navigation plus intéressante et plus ergonomique.

Par exemple, pour un formulaire de connexion, si la personne ne renseigne pas tous les champs correctement, nous pouvons lui signaler par un message d'erreur et donc modifier les informations.

Dans un premier temps, lors du remplissage des champs, s'il y a le bon nombre de caractères, les "input" seront verts sinon ils seront rouges et ne seront pas validés.

->

Formulaire ->

```

106 <div id="login">
107   <form action="login.php" id="login-form" method="post" autocomplete="off">
108     <div id="form-title">
109       Connexion
110     </div>
111     <div class="field">
112       <label class="field-label" for="field-username">Identifiant</label>
113       <input class="field-input" id="field-username" name="username" type="text" placeholder="Identifiant">
114       <p class="field-info">Obligatoire - doit contenir au minimum 3 caractères</p>
115     </div>
116     <div class="field">
117       <label class="field-label" for="field-password">Mot de passe</label>
118       <input class="field-input" id="field-password" name="password" type="password" placeholder="Mot de passe">
119       <p class="field-info">Obligatoire - doit contenir au minimum 3 caractères</p>
120     </div>
121     <div id="errors"></div>
122     <button id="login-submit">Connexion</button>
123   </form>
124 </div>

```



Le code pour arriver à ce résultat :  
Démarrage de l'écouteur d'événement

->

```
1  var app = {  
2    init: function() {  
3      console.log('app.init');  
4      app.inputs = document.querySelectorAll('.field-input');  
5      console.log(app.inputs);  
6  
7      var form = document.querySelector('#login-form');  
8      form.addEventListener('submit', app.validateForm);  
9  
10     app.errorsArea = document.querySelector('#errors');  
11  }
```

'blur' est le fait de sortir des "input" par un click à l'extérieur de "input" rempli.

->

```
for(var index = 0; index < app.inputs.length; index++) {  
  var input = app.inputs[index];  
  input.addEventListener(  
    // typeEvt,  
    'blur',  
    // handler  
    function(evt) {  
      var input = evt.target;  
      app.validateInput(input);  
    }  
  );  
}
```

Une fois sorti à l'extérieur de "input" rempli, la fonction "validateInput" réinitialise la classe 'field-input' que l'on retrouve dans le formulaire en html et elle vérifie que le nombre de caractères ne soit pas inférieur à 3 (pour cet exemple)

->

```
validateInput: function (input) {  
  // Reset des classes CSS pour repartir sur une base saine.  
  input.className = 'field-input';  
  
  // Validation : si valeur du champ >= 3 caractères, alors  
  // bordure verte, sinon rouge.  
  if (input.value.length < 3) {  
    // Manipulation directe des styles en JS : pas une bonne pratique  
    // (principe SRP). Éviter de le faire.  
    // evt.target.style.borderColor = 'red';  
  
    // Mieux : contrôler en JS la présence ou l'absence d'une classe CSS,  
    // et implémenter les styles dans la feuille de style.  
    // evt.target.className = evt.target.className + ' ' + 'field-input-red';  
  
    // Solution finale : utiliser les utilitaires de gestion des classes.  
    input.classList.add('field-input-red');  
    return input.placeholder + ' doit contenir au moins 3 caractères.'  
  }  
  else {  
    // evt.target.style.borderColor = 'green';  
    input.className = input.className + ' ' + 'field-input-green';  
  }  
},
```

Si le nombre de caractère est inférieur à 3 alors la nouvelle classe sera 'field-input-red' si le nombre de caractère est correct on a donc un 'field-input-green'.

Ces deux nouvelles classes sont créées dans le css et donc utilisées seulement en cas de besoin dynamiquement.

->

```
.field-input-green {  
  border-color: #00d1b2;  
}  
.field-input-red {  
  border-color: #ff3860;  
}
```

Si nous ne faisons pas attention aux champs de couleur en tant qu'utilisateur et que nous appuyons sur le bouton "connexion". Une autre "alerte" nous est donnée après la soumission du formulaire  
->

**Connexion**

**Identifiant**

ke

Obligatoire - doit contenir au minimum 3 caractères

**Mot de passe**

•

Obligatoire - doit contenir au minimum 3 caractères

Identifiant doit contenir au moins 3 caractères.

Mot de passe doit contenir au moins 3 caractères.

CONNEXION

Dans notre formulaire html, nous avons une div, id "errors", prête à l'emploi en cas d'erreur après la soumission du formulaire.

La fonction utilisera donc 'displayErrors' :

->

```
validateForm: function(evt) {
    var errors = app.checkFormErrors();
    if (errors.length) evt.preventDefault();
    app.displayErrors(errors);
},

checkFormErrors: function() {
    var errors = [];
    for (var index = 0; index < app.inputs.length; index++) {
        var input = app.inputs[index];
        var error = app.validateInput(input);
        if (error) errors.push(error);
    }
    return errors;
},

displayErrors: function(errors) {
    app.errorsArea.innerHTML = '';
    for (var index = 0; index < errors.length; index++) {
        // app.errorsArea.innerHTML += '<p class="error">' + errors[index] + '</p>'
        var errorParag = document.createElement('p');
        errorParag.classList.add('error');
        errorParag.textContent = errors[index];
        app.errorsArea.appendChild(errorParag);
    }
}
```

Elle crée un paragraphe, utilise la classe error qui vient de notre css,

->

```
#errors {
    margin-top: 1em;
}

.error {
    border-radius: 4px;
    background-color: #ff3860;
    color: #fff;
    font-size: .9em;
    padding: 1em;
    margin-bottom: 1em;
    line-height: 1.4;
}
```

et remplit les champs d'erreurs détectées.

Une fois toutes les erreurs rectifiées par l'utilisateur, il ne lui reste plus qu'à cliquer sur le bouton de connexion et profiter du site qu'il souhaite visiter.

---

## 2. Précisez les moyens utilisés :

J'ai utilisé VS Code ainsi que le cours présenté par le professeur. Utilisation de JavaScript ainsi que PHP, html et css.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour cet exercice en fin de journée.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ O'Clock

Chantier, atelier, service ▶ O>Login

Période d'exercice ▶ Du : 02/06/2019

## Activité-type 2 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 :

Durant le mois de spécialisation (Symfony 4), nous avons été évalué sur nos connaissances générales. Nous avons eu une semaine complète pour développer un projet de type FAQ avec la gestion utilisateur, le mcd, dictionnaire de données, la base donnée, une partie administrateur, et différentes gestions de possibilité utilisateur, ainsi que l'esthétique du site.

Le dictionnaire de données est très important car il nous donne toutes les informations dont on a besoin pour chaque entité dans notre base de données. Chaque donnée doit être nommée et décrite. Il s'agit également de préciser son type (booléen, nombre, texte court, texte long, date, etc...) Il nous indique aussi où chaque entité est reliée et comment. Ce travail, s'il est effectué correctement, nous facilitera le projet une fois terminé.

Pour ce projet, nous allons donc commencer par consulter le dictionnaire de données rempli dans son tableau :

->

#### QUESTION

Champ	Type	Spécialités	Description
id	INT	PRIMARY KEY, NOT NULL, AUTO_INCREMENT	Identifiant de la question
entitled	VARCHAR(64)	NOT NULL	La question
created_at	TIMESTAMP	NOT NULL, DATE_TIME	Date création
updated_at	TIMESTAMP	DATE_TIME	Date de modification
User_id	INT	NOT NULL	id de l'utilisateur

#### ANSWER

Champ	Type	Spécialités	Description
id	INT	PRIMARY KEY, NOT NULL, AUTO_INCREMENT	Identifiant de la réponse
reply	LONGTEXT	NOT NULL	La réponse
created_at	TIMESTAMP	NOT NULL, DATE_TIME	Date de création
updated_at	TIMESTAMP	DATE_TIME	Date de modification
user_id	INT	NOT NULL, UNSIGNED, FOREIGN KEY REFERENCES User(id)	Id de l'utilisateur
question_id	INT	NOT NULL, UNSIGNED, FOREIGN KEY REFERENCES question(id)	Id de la question

Voici le tableau avec toutes les informations de l'entité question :

Id  
La question  
Date de création  
Date de modification  
Id de l'utilisateur qui a écrit la question

Le tableau avec les informations des réponses :

Id  
La réponse  
Date de création  
Date de modification  
Id de l'utilisateur qui a écrit la réponse  
Id de la question dont on répond

## TAG

Champ	Type	Spécialités	Description
id	INT	PRIMARY KEY, NOT NULL, AUTO_INCREMENT	Identifiant
title	VARCHAR(255)	NOT NULL	Nom du tag
created_at	TIMESTAMP	NOT NULL	Date de création du lieu
updated_at	TIMESTAMP	NULL	Date de modification du lieu

### Tableau avec les informations des tags :

Id  
Le nom du tag  
Date de création  
Date de modification

## User

Champ	Type	Spécialités	Description
id	INT	PRIMARY KEY, NOT NULL, AUTO_INCREMENT	Identification unique par membre
username	VARCHAR(255)	NOT NULL	Nom d'utilisateur
email	VARCHAR(255)	NOT NULL	Email
rôle	JSON	NOT NULL	Rôle de l'utilisateur
created_at	TIMESTAMP	NOT NULL, DATE_TIME	Date de création

### Dictionnaire des utilisateurs :

Id  
Le nom de l'utilisateur  
L'email  
Le rôle de l'utilisateur  
Date de création

## QUESTION\_TAG

Champ	Type	Spécialités	Description
id	INT	PRIMARY KEY, NOT NULL, AUTO_INCREMENT	Identification de la relation question_tag
question_id	INT	NOT NULL, UNSIGNED, FOREIGN KEY REFERENCES question(id)	Id de la question
tag_id	INT	NOT NULL, UNSIGNED, FOREIGN KEY REFERENCES tag(id)	Id du tag

### Dictionnaire de relation entre question et tag :

Id  
Id de la question  
Id du tag

Chaque entité dispose de plusieurs informations comme le type et la spécialité.

Commençons par les différents types :

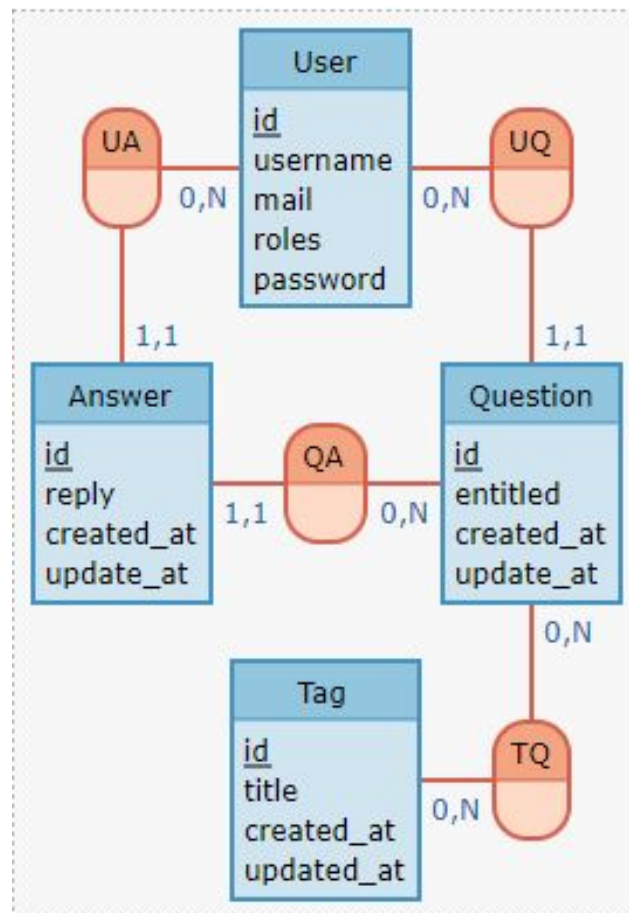
- Numérique avec des nombres entiers (positif, négatif {UNSIGNED}, {INT} en précisant le nombre de chiffre que l'on veut entre parenthèse, une limite {ZEROFILL}, nombres décimaux, etc...).
- Alphanumérique (CHAR, VARCHAR avec le nombre de caractères maximum entre parenthèses, TEXT, JSON, etc.).
- Temporel (DATE juste la date, DATETIME la date et l'heure, TIME juste l'heure, YEAR, TIMESTAMP, etc...).
- 

Il y a aussi les spécialités :

- NULL ou NOT NULL pour préciser si le champ doit être rempli.
- AUTO\_INCREMENT pour incrémenter les champs que l'on souhaite à la création de notre nouvelle donnée (très utilisé pour les id).
- FOREIGN\_KEY pour créer des clefs étrangères.

Passons au MCD (Modèle Conceptuel des Données), Le MCD est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux à l'aide de diagrammes codifiés dont les éléments suivants font partie :

- Les entités (1 rectangle = 1 objet) ;
- Les propriétés (la liste des données de l'entité) ;
- Les relations qui expliquent et précisent comment les entités sont reliées entre elles (les ovales avec leurs « pattes » qui se rattachent aux entités) ;
- Les cardinalités (les petits chiffres au-dessus des « pattes »).

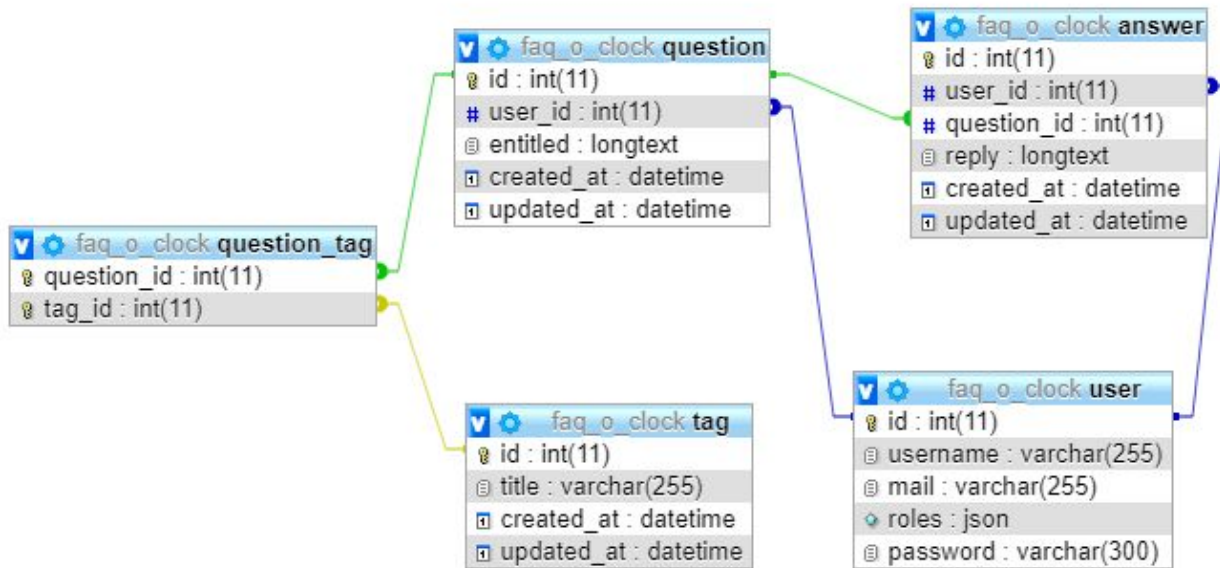


Il existe trois types de relation :

- One to One (un à un), il n'existe qu'un et un seul enregistrement dans la table étrangère pour chaque enregistrement de la table principale.
- One to Many (un à plusieurs), il existe un ou plusieurs enregistrements connexes dans la table étrangère pour chaque enregistrement de la table principale.
- Many to Many (plusieurs-à-plusieurs) Il existe plusieurs enregistrements connexes dans la table étrangère pour chaque enregistrement de la table principale et il existe plusieurs enregistrements dans la table principale pour chaque enregistrement de la table étrangère.



Un aperçu de phpMyAdmin avec les relations une fois saisies en base de données :



Pour créer la base de données SQL (qui veut dire Langage de requête structurée. (En anglais : Structured Query Language), j'ai utilisé, pour cet exercice, Doctrine, un outil très utile, avec Symfony 4, qui aide à créer des bases de données et plus encore (je vais aller à l'essentiel pour après installation de doctrine et de tout ce qu'il faut pour l'utiliser).

Dans un premier temps, je me branche à ma base données :

->

```
.env.local x
.env.local
1 DATABASE_URL=mysql://root:@127.0.0.1:3306/faq_o_clock
2
```

Avec la commande suivant dans mon terminal je crée la base de données :

->

```
Windows PowerShell
PS C:\Users\keval\wamp\www\Evaluation\FAQ-O-Clock> php bin/console doctrine:database:create
```

Avec la commande suivant je crée ou modifie une entité :

->

```
Windows PowerShell
PS C:\Users\keval\wamp\www\Evaluation\FAQ-O-Clock> php bin/console make:entity
```

Une fois la commande entrée, je n'ai plus qu'à saisir les caractéristiques que nous avons vues ci-dessus avec le dictionnaire de données et le MCD :

->

```
3  Class name of the entity to create or update:
4  > Product
5
6  New property name (press <return> to stop adding fields):
7  > name
8
9  Field type (enter ? to see all types) [string]:
10 > string
11
12 Field length [255]:
13 > 255
14
15 Can this field be null in the database (nullable) (yes/no) [no]:
16 > no
17
18 New property name (press <return> to stop adding fields):
19 > price
20
21 Field type (enter ? to see all types) [string]:
22 > integer
23
24 Can this field be null in the database (nullable) (yes/no) [no]:
25 > no
26
27 New property name (press <return> to stop adding fields):
28 >
29 (press enter again to finish)
```

Je réitère la dernière commande autant de fois que je souhaite créer une nouvelle entité ou de nouveaux champs.

Une fois terminé, je peux vérifier mes entités et envoyer le tout en base de données avec nouvelles commandes.

- php bin/console make:migration

Qui créera un fichier .php à envoyer à la base de données.

- php bin/console make:migration:migrate

Qui enverra et créera la base de données.

Voilà une solution simple et efficace pour créer et gérer une base de données.

---

## 2. Précisez les moyens utilisés :

J'ai utilisé principalement Symfony 4, Bootstrap 4, CoMocodo, PhpMyAdmin Doctrine et certaines Bundle qu'il propose et bien sûr VS Code.

## 3. Avec qui avez-vous travaillé ?

Travail effectué seul durant l'évaluation en fin de spécialisation Symfony 4.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ O'Clock

Chantier, atelier, service ▶ FAQ-O-Clock

Période d'exercice ▶ Du : 20/09/2019 au : 02/09/2019

**Activité-type 2** 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 :**

Durant notre mois de spécialisation, nous avons effectué un projet conducteur « Hell' O Ciné ». Un remake du célèbre site français « allo ciné » qui présente les détails des films sortis et à venir.

On peut consulter sur ce site, les castings, les sorties etc... !

Hell O Ciné est conçu sur le même principe mais avec des fonctionnalités plus poussées à savoir que nous pouvons être administrateur et donc voir, créer, modifier et supprimer n'importe quelle information de notre base donnée.

Avant toute chose il nous faut nous connecter à notre base de données que l'on a créée en amont. Pour cela, il y a plusieurs possibilités comme la classe PDO (Représente une connexion entre PHP et un serveur de base de données). Mais, dans notre cas avec Symfony 4, il nous faut utiliser le fichier '.env' qui se trouve à la racine de notre projet.

'env' :

->

```
###> doctrine/doctrine-bundle ###
# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-url
# For an SQLite database, use: "sqlite://%kernel.project_dir%/var/data.db"
# Configure your db driver and server_version in config/packages/doctrine.yaml
# DATABASE_URL=mysql://db_user:db_password@127.0.0.1:3306/db_name
DATABASE_URL=mysql://root:kevalois@127.0.0.1:3306/movie_db
###< doctrine/doctrine-bundle ###
```

Au niveau de la ligne 'DATABASE\_URL=' nous devons enregistrer les informations de connexion pour se connecter au serveur mysql.

Dans notre cas, nous sommes sur un projet en local donc l'adresse ip est 127.0.0.1:3306 (bien sûr mysql doit être installé sur notre machine).

Nous pouvons donc créer un fichier .env.local pour éviter que celui-ci ne se mélange au projet car les développeurs qui travaillent sur ce projet n'auront pas forcément le même nom de base de données ou d'utilisateur ou de mot de passe.

Cette partie adresse est personnelle pour le moment. Elle n'est pas encore en production et en développement donc pas encore finalisée.

Ensuite nous devons remplacer 'db\_user' par 'root' (l'utilisateur), 'db\_password' par 'kevalois' (mot de passe) et 'movie\_db' est le nom de la table dans notre base de données.

## Connection à phpMyAdmin



**Bienvenue dans phpMyAdmin**

**Langue - Language**

Français - French

**Connexion**

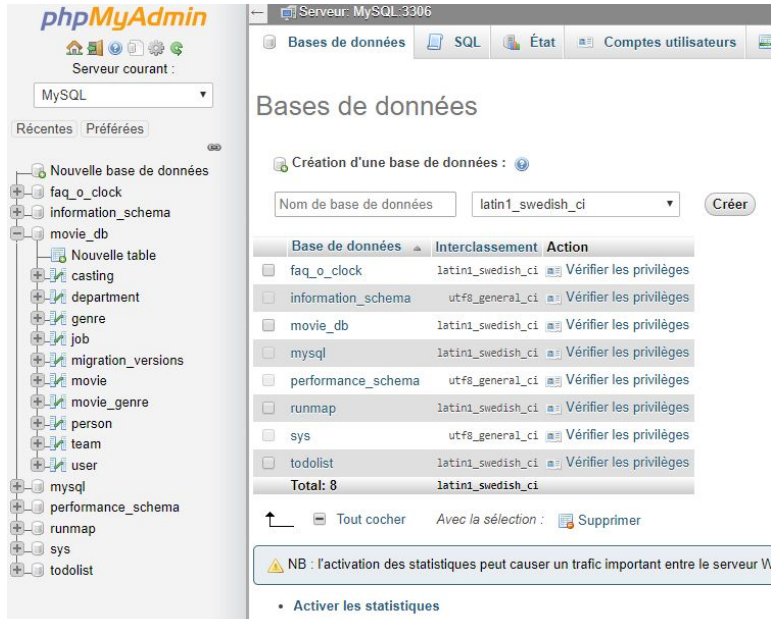
Utilisateur :

Mot de passe :

Choix du serveur :

**Exécuter**

## Toutes les tables créer



phpMyAdmin

Serveur courant : MySQL

Récentes Préférées

- Nouvelle base de données
- faq\_o\_clock
- information\_schema
- movie\_db
  - Nouvelle table
  - casting
  - department
  - genre
  - job
  - migration\_versions
  - movie
  - movie\_genre
  - person
  - team
  - user
- mysql
- performance\_schema
- runmap
- sys
- todolist

**Bases de données**

Création d'une base de données :

Nom de base de données : latin1\_swedish\_ci **Créer**

Base de données	Interclassement	Action
faq_o_clock	latin1_swedish_ci	Vérifier les privilèges
information_schema	utf8_general_ci	Vérifier les privilèges
movie_db	latin1_swedish_ci	Vérifier les privilèges
mysql	latin1_swedish_ci	Vérifier les privilèges
performance_schema	utf8_general_ci	Vérifier les privilèges
runmap	latin1_swedish_ci	Vérifier les privilèges
sys	utf8_general_ci	Vérifier les privilèges
todolist	latin1_swedish_ci	Vérifier les privilèges
<b>Total: 8</b>	<b>latin1_swedish_ci</b>	

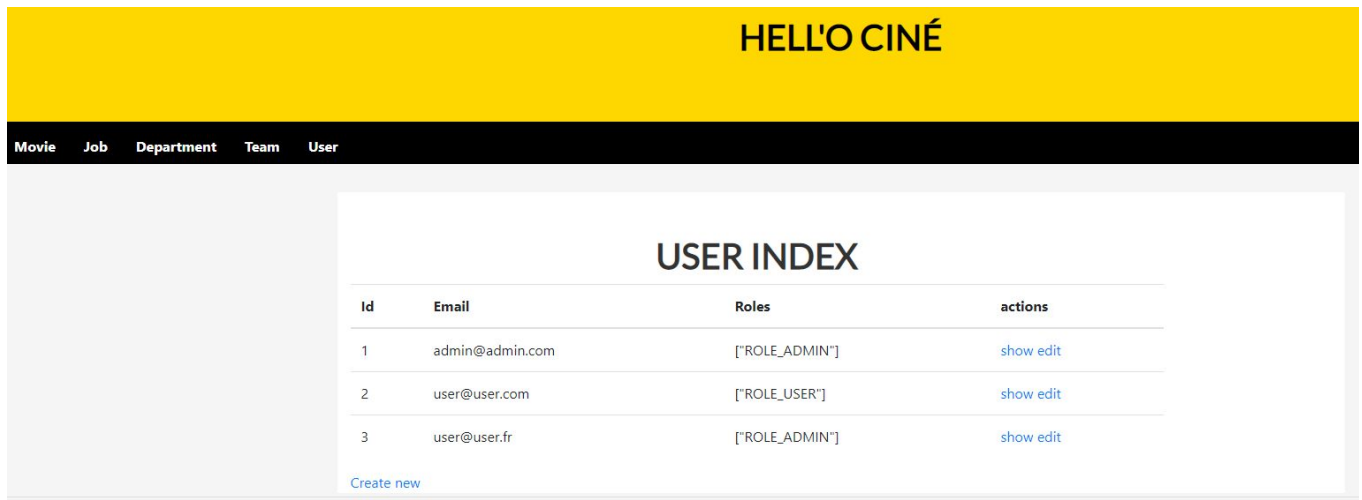
↑ Tout cocher Avec la sélection : Supprimer

**NB :** l'activation des statistiques peut causer un trafic important entre le serveur V

- Activer les statistiques

Nous sommes connectés à notre base de données créée en amont. Nous pouvons donc commencer à avancer sur notre projet et nous allons maintenant nous concentrer (pour cet exemple) sur l'entité 'user'. Pour ce projet, j'ai utilisé le framework Symfony 4.

Nous voulons tout d'abord visualiser la liste de tous les 'user' :



**HELL'O CINÉ**

Movie Job Department Team **User**

**USER INDEX**

Id	Email	Roles	actions
1	admin@admin.com	["ROLE_ADMIN"]	<a href="#">show edit</a>
2	user@user.com	["ROLE_USER"]	<a href="#">show edit</a>
3	user@user.fr	["ROLE_ADMIN"]	<a href="#">show edit</a>

[Create new](#)

Quand on clique sur User, la liste de tous les utilisateurs s'affiche. Pour cela, le code utilisé est celui-ci :  
->

```

17 class UserController extends AbstractController
18 {
19     /**
20      * @Route("/", name="user_index", methods={"GET"})
21      */
22     public function index(UserRepository $userRepository): Response
23     {
24         return $this->render('backend/user/index.html.twig', [
25             'users' => $userRepository->findAll(),
26         ]);
27     }

```

Symfony nous facilite le travail en faisant appel au repository de 'User' avec la fonction 'findAll()' pour récupérer tous les utilisateurs de la base de données. Nous renvoyons le tout dans la template 'backend/user/index.html.twig'.

Pour utiliser le système de Repository avec Symfony 4, nous devons l'appeler en haut de la page de notre code :

'use App\Repository\UserRepository;'

->

```

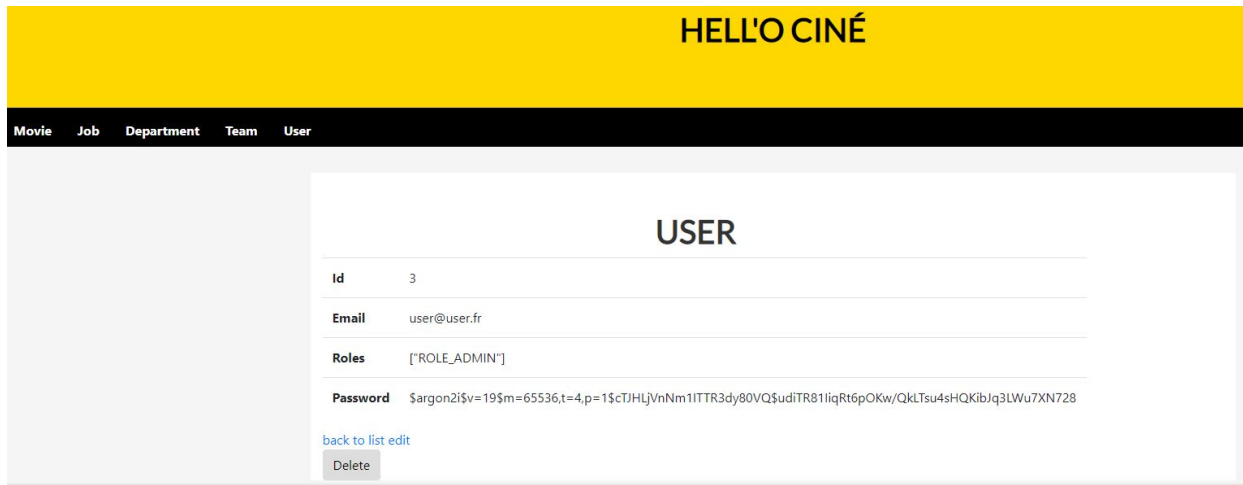
1  <?php
2
3  namespace App\Controller\Backend;
4
5  use App\Entity\User;
6  use App\Form\UserType;
7  use App\Repository\UserRepository;
8  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
9  use Symfony\Component\HttpFoundation\Request;
10 use Symfony\Component\HttpFoundation\Response;
11 use Symfony\Component\Routing\Annotation\Route;
12 use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;
13

```

Une fois ces étapes effectuées, nous pouvons utiliser toutes les informations de notre base de données 'User' dans la template où nous avons envoyé les informations.

Nous voulons maintenant voir les informations d'un utilisateur en particulier quand on clique sur l'utilisateur que l'on souhaite. Pour cela voici le résultat souhaité :

->



USER	
<b>Id</b>	3
<b>Email</b>	user@user.fr
<b>Roles</b>	["ROLE_ADMIN"]
<b>Password</b>	\$argon2i\$v=19\$m=65536,t=4,p=1\$cTJHljVnNm1ITTR3dy80VQ\$udiTR81liqRt6pOKw/QkLTsu4sHQKibJq3LWu7XN728

[back to list edit](#)

Delete

Nous voyons les différentes informations de l'utilisateur sélectionné. Pour cela nous devons coder de la manière suivante :

->

```
60  /**
61  * @Route("/{id}", name="user_show", methods={"GET"})
62  */
63  public function show(User $user): Response
64  {
65      return $this->render('backend/user/show.html.twig', [
66          'user' => $user,
67      ]);
68  }
```

En cliquant sur notre utilisateur, l'id nous est renvoyé dans l'url et donc nous pouvons sélectionner directement dans notre base de données grâce à 'use App\Entity\User;' qui nous renvoie la 'Response'. Nous pouvons donc utiliser la variable '\$user' qui contient les informations de l'utilisateur suivant l'id.

Voyons maintenant comment créer et enregistrer un nouvel utilisateur :

On accède au formulaire de création d'utilisateur, dans notre cas nous utilisons les formulaires Symfony 4 :

->

```
1 <?php
2
3 namespace App\Form;
4
5 use App\Entity\User;
6 use Symfony\Component\Form\AbstractType;
7 use Symfony\Component\Form\FormBuilderInterface;
8 use Symfony\Component\OptionsResolver\OptionsResolver;
9 use Symfony\Component\Form\Extension\Core\Type\ChoiceType;
10
11 class UserType extends AbstractType
12 {
13     public function buildForm(FormBuilderInterface $builder, array $options)
14     {
15         $builder
16             ->add('email')
17             ->add('roles', ChoiceType::class,
18                 array(
19                     'multiple' => true,
20                     'expanded' => true,
21                     'choices' => array(
22                         'Admin' => 'ROLE_ADMIN',
23                         'User' => 'ROLE_USER',
24                     )
25                 ))
26             ->add('password')
27         ;
28     }
29
30     public function configureOptions(OptionsResolver $resolver)
31     {
32         $resolver->setDefaults([
33             'data_class' => User::class,
34         ]);
35     }
36 }
37
```

Notre formulaire est créé grâce à la commande :

'php bin/console make:form'

Ainsi nous retrouvons le formulaire généré grâce à cette commande. Nous pouvons à présent le personnaliser suivant les informations que l'utilisateur remplir.

Dans notre nous utilisons le 'ChoiceType::class' pour proposer le un choix sur plusieurs possibilités pour les rôles 'ADMIN' ou 'USER'.

Formulaire vide

Formulaire rempli

HELLO CINÉ	
<div>Movie Job Department Team User</div> <div><h3>CREATE NEW USER</h3><div>Email</div><div>Roles<ul style="list-style-type: none"><li><input checked="" type="checkbox"/> Admin</li><li><input type="checkbox"/> User</li></ul></div><div>Password</div><div><div>Save</div><div>back to list</div></div></div>	<div>Movie Job Department Team User</div> <div><h3>CREATE NEW USER</h3><div>Email</div><div>Roles<ul style="list-style-type: none"><li><input checked="" type="checkbox"/> Admin</li><li><input type="checkbox"/> User</li></ul></div><div>Password</div><div><div>Save</div><div>back to list</div></div></div>



Voyons maintenant le code utilisé pour cette création d'utilisateur :

```
29  /**
30  * @Route("/new", name="user_new", methods={"GET", "POST"})
31  */
32  public function new(Request $request, UserPasswordEncoderInterface $passwordEncoder): Response
33  {
34      $user = new User();
35      $form = $this->createForm(UserType::class, $user);
36      $form->handleRequest($request);
37
38      if ($form->isSubmitted() && $form->isValid()) {
39
40          // Mot de passe qui vient du formulaire
41          $password = $user->getPassword();
42          // On l'encode via le passwordEncoder reçu depuis la méthode du contrôleur
43          $encodedPassword = $passwordEncoder->encodePassword($user, $password);
44          // On écrase le mot de passe avec le mot de passe encodé
45          $user->setPassword($encodedPassword);
46
47          $entityManager = $this->getDoctrine()->getManager();
48          $entityManager->persist($user);
49          $entityManager->flush();
50
51          return $this->redirectToRoute('backend_user_index');
52      }
53
54      return $this->render('backend/user/new.html.twig', [
55          'user' => $user,
56          'form' => $form->createView(),
57      ]);
58  }
```

Ligne 34 : on crée un nouvel utilisateur.  
Ligne 35 : on crée un formulaire (vide) avec les champs nécessaires selon l'User Type :: classe'.  
Ligne 36 : on remplit notre formulaire créé avec les informations que l'utilisateur a remplies.  
Ligne 38 : si le formulaire est soumis et correctement rempli, on peut passer à la suite. Sinon on ne peut pas soumettre le formulaire.  
Ligne 40 à 45 : ( sécurité) on encode de mot de passe.  
Ligne 47 : on fait appel au 'manager' pour enregistrer les données  
Ligne 48 : on remplit les données de notre utilisateur.  
Ligne 49 : on enregistre les tous dans la base de données.

On redirige vers la liste des utilisateurs si toutes les informations on étaient enregistrées en base de données.

HELL'O CINÉ			
Movie	Job	Department	Team User
USER INDEX			
Id	Email	Roles	actions
1	admin@admin.com	["ROLE_ADMIN"]	<a href="#">show edit</a>
2	user@user.com	["ROLE_USER"]	<a href="#">show edit</a>
4	kevalois@htmail.fr	["ROLE_ADMIN"]	<a href="#">show edit</a>
<a href="#">Create new</a>			

Voyons maintenant comment modifier un utilisateur pour cela retournons dans la liste des utilisateurs :

->

HELL'O CINÉ

MovieJobDepartmentTeamUser

USER INDEX

Id	Email	Roles	actions
1	admin@admin.com	["ROLE_ADMIN"]	<a href="#">show edit</a>
2	user@user.com	["ROLE_USER"]	<a href="#">show edit</a>
3	user@user.fr	["ROLE_ADMIN"]	<a href="#">show edit</a>

[Create new](#)

Nous allons faire une modification de rôle pour l'id 2 'user@user.com'. pour cela nous allons cliquer sur 'edit' :

->

HELL'O CINÉ

MovieJobDepartmentTeamUser

EDIT USER

Email

user@user.com

Roles

☐ Admin

☒ User

Password

Update

[back to list](#)

Delete

Comme nous pouvons le voir le rôle de l'utilisateur est en 'User'.

->

Nous enlevons le rôle 'User' et le passons en 'Admin' et on valide la modification.

HELL'O CINÉ

MovieJobDepartmentTeamUser

EDIT USER

Email

user@user.com

Roles

☒ Admin

☐ User

Password

Update

[back to list](#)

Delete

Page 34

La modification dans le code fonctionne comme suit :

->

```
70  /**
71  * @Route("/{id}/edit", name="user_edit", methods={"GET","POST"})
72  */
73  public function edit(Request $request, User $user, UserPasswordEncoderInterface $passwordEncoder): Response
74  {
75      // On stocke le mot de passe courant
76      $currentPassword = $user->getPassword();
77      // On passe le mot de passe à vide
78      $user->setPassword('');
79
80      $form = $this->createForm(UserType::class, $user);
81      // Ici, le nouveau mot de passe sera renseigné dans $user
82      $form->handleRequest($request);
83
84      if ($form->isSubmitted() && $form->isValid()) {
85
86          // Si le mot de passe n'est pas modifié, on conserve l'ancien
87          if (empty($user->getPassword())) {
88              $user->setPassword($currentPassword);
89              // Sinon, on encode le nouveau mot de passe
90          } else {
91              // Mot de passe qui vient du formulaire
92              $password = $user->getPassword();
93              // On l'encode via le passwordEncoder reçu depuis la méthode du contrôleur
94              $encodedPassword = $passwordEncoder->encodePassword($user, $password);
95              // On écrase le mot de passe avec le mot de passe encodé
96              $user->setPassword($encodedPassword);
97          }
98
99      $this->getDoctrine()->getManager()->flush();
100
101      return $this->redirectToRoute('backend_user_index');
102  }
103
104  return $this->render('backend/user/edit.html.twig', [
105      'user' => $user,
106      'form' => $form->createView(),
107  ]);
108  }
```

Ligne 76 à 78 : on s'occupe du mot de passe

Ligne 80 : on crée un formulaire (vide) avec les champs nécessaires selon l'User Type :: classe'.

Ligne 82 : on remplit notre formulaire créé avec les informations que l'utilisateur a remplies.

Ligne 84 : si le formulaire est soumis et correctement rempli, on peut passer à la suite. Sinon on ne peut pas soumettre le formulaire.

Ligne 87 à 96 : on garde ou on modifie le mot de passe suivant comment est rempli le formulaire par l'utilisateur.

Ligne 99 : on enregistre les modifications en base de données.

On redirige (ligne 101) vers la liste des utilisateurs si toutes les informations ont été modifiées et enregistrées en base de données.

->

HELL'O CINÉ			
Movie	Job	Department	Team User
USER INDEX			
Id	Email	Roles	actions
1	admin@admin.com	["ROLE_ADMIN"]	<a href="#">show edit</a>
2	user@user.com	["ROLE_ADMIN"]	<a href="#">show edit</a>
4	kevalois@hotmail.fr	["ROLE_ADMIN"]	<a href="#">show edit</a>
5	toto@toto.com	["ROLE_USER"]	<a href="#">show edit</a>
<a href="#">Create new</a>			

Nous remarquons que la modification de l'id 2, 'user@user.com' est devenu 'Admin'.

Enfin, voyons comment supprimer un utilisateur :

->

HELL'O CINÉ			
Movie Job Department Team User			
USER INDEX			
Id	Email	Roles	actions
1	admin@admin.com	['ROLE_ADMIN']	<a href="#">show</a> <a href="#">edit</a>
2	user@user.com	['ROLE_USER']	<a href="#">show</a> <a href="#">edit</a>
3	user@user.fr	['ROLE_ADMIN']	<a href="#">show</a> <a href="#">edit</a>
<a href="#">Create new</a>			

Nous choisissons de supprimer l'id 3, 'user@user.fr'. Pour cela il nous faut voir le détail de celui-ci en cliquant sur 'show'.

->

127.0.0.1:8000 indique  
Are you sure you want to delete this item?

HELL'O CINÉ	
Movie Job Department Team User	
USER	
Id	3
Email	user@user.fr
Roles	['ROLE_ADMIN']
Password	\$argon2i\$v=19\$m=65536,t=4,p=1\$YIRIQ0VvT0dYMOI4b38UbaSpU/3u8qoTNw5l8doN2fV9lmgNkFhysZFhysZFXgAlxhA2gU
<a href="#">back to list edit</a>	
<input type="button" value="Delete"/>	

On le supprime avec le bouton 'Delete'. Voyons le comportement du code :

->

```
110  /**
111   * @Route("/{id}", name="user_delete", methods={"DELETE"})
112   */
113   public function delete(Request $request, User $user): Response
114   {
115       if ($this->isCsrfTokenValid('delete', $user->getId(), $request->request->get('_token'))) {
116           $entityManager = $this->getDoctrine()->getManager();
117           $entityManager->remove($user);
118           $entityManager->flush();
119       }
120
121       return $this->redirectToRoute('backend_user_index');
122   }
```

Dans un premier temps, nous vérifions la validité du jeton CSRF (Cros-site request forgery) pour la sécurité des données. Puis, pour supprimer dans la base de données, on utilise la méthode 'remove()' qui effacera ce que l'on indiquera dans notre variable donc notre 'User'. Puis on 'flush()' pour enregistrer définitivement l'effacement.

Puis, on fait redirection vers notre page 'backend\_user\_index'

->

HELL'O CINÉ

Movie

Job

Department

Team

User

USER INDEX

Id	Email	Roles	actions
1	admin@admin.com	["ROLE_ADMIN"]	<a href="#">show edit</a>
2	user@user.com	["ROLE_USER"]	<a href="#">show edit</a>

Create new

Nous avons examiné comment voir, créer, modifier et supprimer une ou plusieurs données dans notre base de données.

Mais Doctrine, nous propose une solution plus rapide pour faire à notre place, le code vu juste avant. Afin de voir, modifier, supprimer nos données, nous pouvons utiliser le générateur de CRUD : 'php bin/console generate:doctrine:crud'

Il fera de lui-même les contrôleurs sur l'entité que l'on souhaitera. Très pratique mais à utiliser seulement si nous n'avons pas de choses particulières à mettre en place dans ces contrôleurs.

## 2. Précisez les moyens utilisés :

J'ai utilisé mon éditeur de texte VS Code, le terminal pour générer les formulaires et entités ainsi que la documentation de Symfony.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul, c'était un exercice sur plusieurs jours pendant la spécialisation Symfony 4

## 4. Contexte

Nom de l'entreprise, organisme ou association ► O'Clock

Chantier, atelier, service ► Hell'O Ciné

Période d'exercice ► Du : 09/092019 au : 21/09/2019

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

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

Pour cette partie, nous allons nous concentrer sur la sécurité qui a été très importante durant ce projet. Pour cela, nous avons beaucoup d'outils plus ou moins performants.

Dans cet exemple, je vais vous montrer comment j'utilise Symfony 4 pour avoir un site internet le plus sécurisé possible.

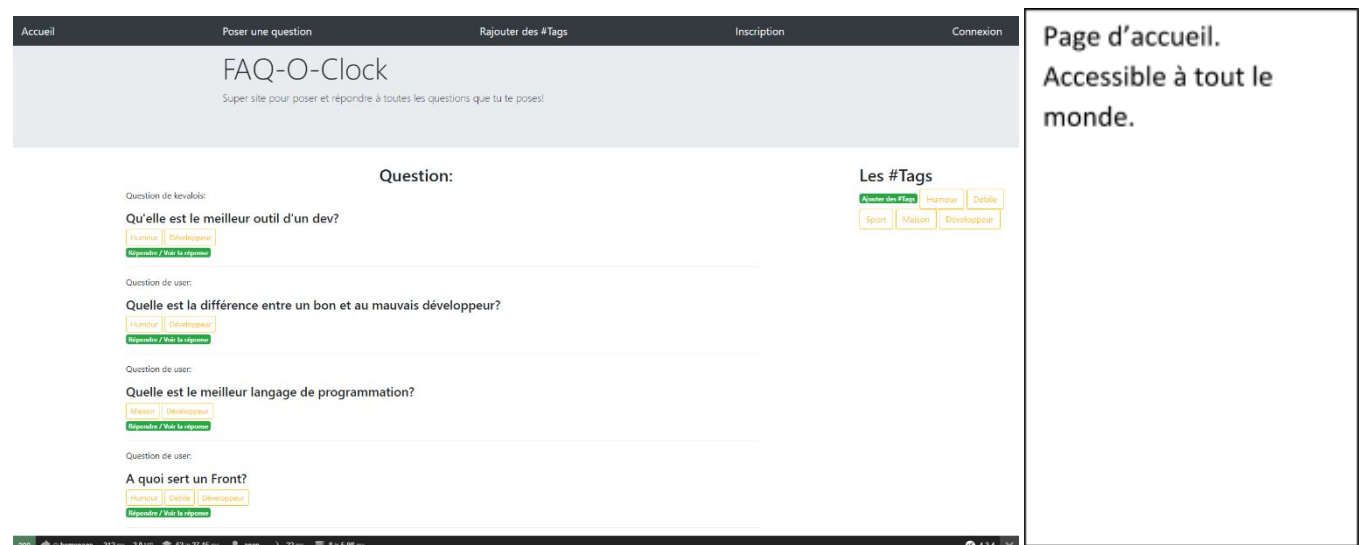
Autant pour les utilisateurs suivant leurs rôles, que pour le mot de passe et les différentes contraintes mais aussi pour éviter de noter n'importe quelles informations dans notre base de données.

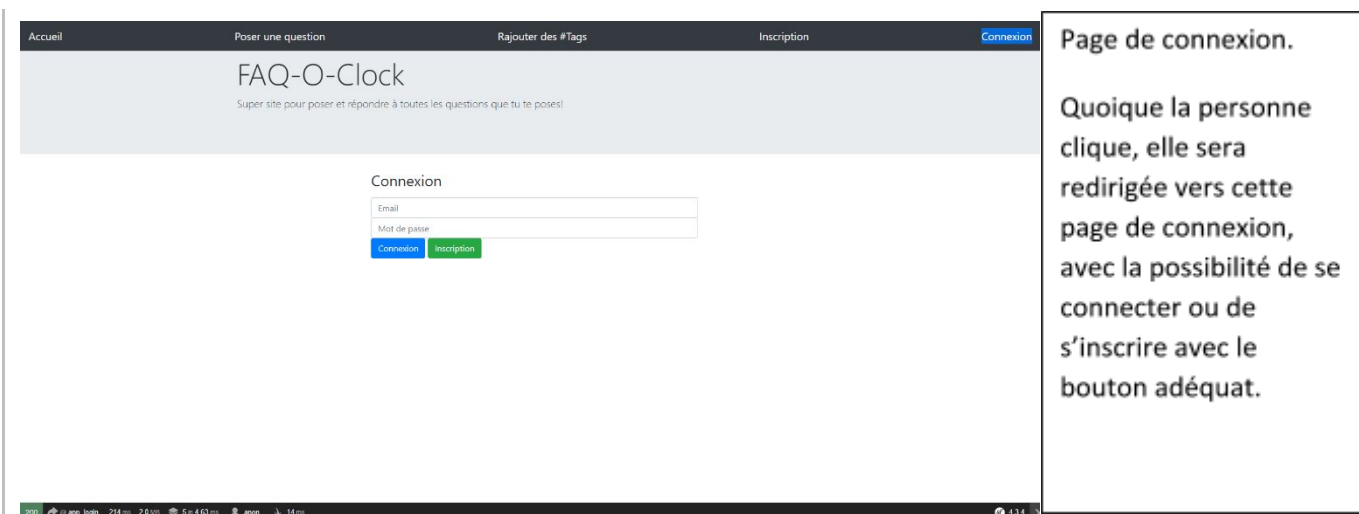
Je vais vous présenter tout cela avec le projet d'évaluation présenté en fin de spécialisation de Symfony 4.

Tout d'abord, voyons comment sécuriser un mot de passe avec Symfony 4. Il nous offre des outils simples et efficaces à mettre en place pour la sécurité de nos données. Commençons par les rôles attribués aux utilisateurs du site.

Les rôles sont très importants pour les sites où une inscription est possible.

Généralement, le fait de se connecter nous donne accès à plus de possibilités et plus de fonctionnalités. Dans notre exemple, si nous ne sommes pas connectés, nous ne pourrions pas participer aux réponses ou même les voir. Il ne sera pas possible également de poser des questions ou créer de #tags. Donc la seule possibilité dans notre cas, est celle de consulter les questions posées par les autres utilisateurs. Voici le résultat en image :





Pour arriver à ce résultat, nous devons nous rendre dans notre code dans le dossier :  
config -> packages -> security.yaml

```

39     access_control:
40         # On ouvre l'accès au login au user anonyme (surtout si votre appli est "privée")
41         - { path: ^/login$, roles: IS_AUTHENTICATED_ANONYMOUSLY }
42
43         # USER
44         - { path: ^/backend/[a-z]+/, roles: ROLE_USER }
45
46     role_hierarchy:
47         ROLE_ADMIN: ROLE_USER

```

A la ligne 41, nous autorisons les utilisateurs avec le rôle 'IS\_AUTHENTICATED\_ANONYMOUSLY', donc anonyme, pour qu'ils aient accès simplement à la page d'accueil, au formulaire de login ainsi que d'inscription.

En sécurité, nous donnons accès au 'user' pour toutes les url commençant par '/backend/...'.  
Pour cela, nous devons faire attention à la façon dont nous nommons les routes des contrôleurs :

```

66     /**
67     * @Route("/backend/question/{id}", name="question_show", methods={"GET", "POST"}, requirements={"id"="\d+"})
68     */
69     public function show(Question $question, Request $request, ObjectManager $manager)
70     {
71         if (!$question) {
72             throw $this->createNotFoundException('Question introuvable');
73         }
74         $answer = new Answer();

```

Ligne 67, nous remarquons que notre route pour la fonction show est bien '/backend/question/{id}', donc les utilisateurs qui auront accès à cette fonction seront juste les utilisateurs avec le rôle 'user'.  
On a seulement à répéter l'exercice pour chaque contrôleur et avec même la possibilité de complexifier le système des rôles pour d'autres autorisations suivant le rôle de chacun sur le site :

- Rôle 'User'
- Rôle 'Modérateur'
- Rôle 'Admin'

- Etc.

Pour donner un rôle à l'utilisateur quand il s'inscrit dans notre exemple, j'ai décidé qu'il serait automatiquement à l'inscription un 'User'. Pour cela, je suis passé par fonction 'new' dans le contrôleur 'UserController'.

->

```

15  /**
16  * @Route("/new", name="user_new", methods={"GET","POST"})
17  */
18  public function new(Request $request, UserPasswordEncoderInterface $passwordEncoder): Response
19  {
20      $user = new User();
21      $rolesArr = array('ROLE_USER');
22      $form = $this->createForm(RegistrationType::class, $user, [
23          'validation_groups' => ['Default', 'new'],
24      ]);
25      $form->handleRequest($request);
26
27      if ($form->isSubmitted() && $form->isValid()) {
28
29          // Mot de passe qui vient du formulaire
30          $password = $user->getPassword();
31          // On l'encode via le passwordEncoder reçu depuis la méthode du contrôleur
32          $encodedPassword = $passwordEncoder->encodePassword($user, $password);
33          // On écrase le mot de passe avec le mot de passe encodé
34          $user->setPassword($encodedPassword);
35
36          $user->setRoles($rolesArr);
37
38          $entityManager = $this->getDoctrine()->getManager();
39          $entityManager->persist($user);
40          $entityManager->flush();
41
42          return $this->redirectToRoute('app_login');
43      }
44
45      return $this->render('security/new.html.twig', [
46          'user' => $user,
47          'formNew' => $form->createView(),
48      ]);
49  }
50  }
51  }

```

Ligne 21 : je crée et prépare ma variable dans un tableau au format JSON avec le rôle que je souhaite.

Ligne 36 : une fois mon formulaire d'inscription soumis et validé j'enregistre le rôle avec les données de ma variable.

Ligne 38 à 40 : comme nous l'avons vu plus tôt on enregistre le tout dans la base de données.

Pour éviter de noter n'importe quelle information dans nos champs en base de données, MySQL, nous demandons, comme nous l'avons vu, dans la création de base de données ci-dessus, le nombre de caractères, le type, si on veut des chiffres etc...

Mais nous pouvons faire en sorte de le vérifier et de mettre des contraintes directement dans le code au niveau des entités.

->



```

1 <?php
2
3 namespace App\Entity;
4
5 use Doctrine\Common\Collections\ArrayCollection;
6 use Doctrine\Common\Collections\Collection;
7 use Doctrine\ORM\Mapping as ORM;
8 use Symfony\Component\Security\Core\User\UserInterface;
9 use Symfony\Component\Validator\Constraints as Assert;
10 use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
11
12 /**
13  * @ORM\Entity(repositoryClass="App\Repository\UserRepository")
14  * @UniqueEntity(
15  *     fields={"mail"},
16  *     message="email déjà utilisé"
17  * )
18  */
19 class User implements UserInterface
20 {
21     /**
22      * @ORM\Id()
23      * @ORM\GeneratedValue()
24      * @ORM\Column(type="integer")
25      */
26     private $id;
27
28     /**
29      * @ORM\Column(type="string", length=255)
30      */
31     private $username;
32
33     /**
34      * @ORM\Column(type="string", length=255)
35      * @Assert\Email()
36      */
37     private $mail;
38
39     /**
40      * @ORM\Column(type="json")
41      */
42     private $roles = [];
43
44     /**
45      * @ORM\Column(type="string", length=300)
46      * @Assert\Length(min="4", minMessage="4 caractères minimum")
47      */
48     private $password;
49
50     /**
51      * @ORM\OneToMany(targetEntity="App\Entity\Question", mappedBy="user")
52      */
53     private $questions;
54
55     /**
56      * @ORM\OneToMany(targetEntity="App\Entity\Answer", mappedBy="user", orphanRemoval=true)
57      */
58     private $answers;
59
60     public function __construct()
61     {
62         $this->questions = new ArrayCollection();
63         $this->answers = new ArrayCollection();
64     }
65
66     public function getId(): ?int
67     {
68         return $this->id;
69     }
70
71     public function getUsername(): ?string
72     {
73         return $this->username;

```

Ligne 15 et 16 : nous voulons que le mail de notre base de données soit unique, pour cela nous faisons appel une fonction '@UniqueEntity()' qui vérifiera si le mail existe déjà en base de données. Si le mail existe déjà il nous renverra une erreur sinon il laissera dérouler la fonction (vois ci-dessous).

Ligne 35 : nous voulons absolument un format 'Email' donc la fonction '@Assert\Email()' va vérifier si nous avons bien un email. Idem un message nous alertera si le mail n'est pas correctement écrit. (D'autres possibilité existe pour confirmer que le mail existe mais je ne l'ai est pas travaillé pour le moment)

Ligne 46 : nous voulons que notre mot de passe contient au moin 4 caractère pour cela nous utilisons la fonction '@Assert\Length (min="4", minMessage="4 caractères minimum")',  
 Nous avons la possibilité de personnaliser le message d'erreur. On aurait très bien pu demander un nombre de caractère max avec le message qui va avec (voir exemple ci-dessous).

# Inscription

Username

user2

Mail

**ERROR** email déjà utilisé

user@user.fr

Password

**ERROR** 4 caractères minimum

..

Inscription

Symfony nous propose bien plus de possibilités, à nous de choisir ce que l'on souhaite contraindre.

## 2. Précisez les moyens utilisés :

J'ai utilisé VS Code, les formulaires et la sécurité de Symfony 4 ainsi que la documentation et les cours que l'on a vu avant.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet car c'était une évaluation que nous avons eu pendant la spécialisation

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ O'Clock

Chantier, atelier, service ▶ FAQ-O-Clock

Période d'exercice ▶ Du : 20/09/2019 au : 02/09/2019

## 5. Informations complémentaires (facultatif)

## Titres, diplômes, CQP, attestations de formation

*(facultatif)*

Intitulé	Autorité ou organisme	Date
Brevet Technicien Collaborateur d'architecte	Académie de Grenoble	07/2009
BEP Technique de l'architecture et de l'habitat	Académie de Grenoble	07/2007

## Déclaration sur l'honneur

Je soussigné Kévin ALOIS ,

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 Sérézin-du-Rhône

le 12/12/2019

pour faire valoir ce que de droit.

Signature :

A handwritten signature in black ink, consisting of several fluid, overlapping strokes that form a stylized representation of the name.

## Documents illustrant la pratique professionnelle

*(facultatif)*

Intitulé
Cliquez ici pour taper du texte.

## ANNEXES

*(Si le RC le prévoit)*

Prototype WEB :



## PORTFOLIO



Opus igitur est dicere  
possit dura omni specie,  
"Tu autem in specie, non  
videntur, nec omnino res

[en savoir plus](#)



Opus igitur est dicere  
possit dura omni specie,  
"Tu autem in specie, non  
videntur, nec omnino res

[en savoir plus](#)



Opus igitur est dicere  
possit dura omni specie,  
"Tu autem in specie, non  
videntur, nec omnino res

[en savoir plus](#)



Opus igitur est dicere  
possit dura omni specie,  
"Tu autem in specie, non  
videntur, nec omnino res

[en savoir plus](#)



Opus igitur est dicere  
possit dura omni specie,  
"Tu autem in specie, non  
videntur, nec omnino res

[en savoir plus](#)



Opus igitur est dicere  
possit dura omni specie,  
"Tu autem in specie, non  
videntur, nec omnino res

[en savoir plus](#)

Ensembles des de  
travaux effectué  
pendant et après la  
formation  
Clicable et renvoi  
vers une modal

## EXPERIENCES

Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicetatis libertatisque



Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicetatis libertatisque



Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicetatis libertatisque



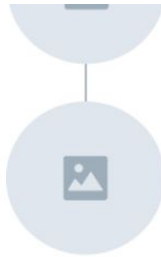
Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo



Experience autre  
que le  
développement



Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicитatis libertatisque



omnino desit illud quo solo  
felicитatis libertatisque

## FORMATION

Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicитatis libertatisque



Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicитatis libertatisque



Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicитatis libertatisque



Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicитatis libertatisque



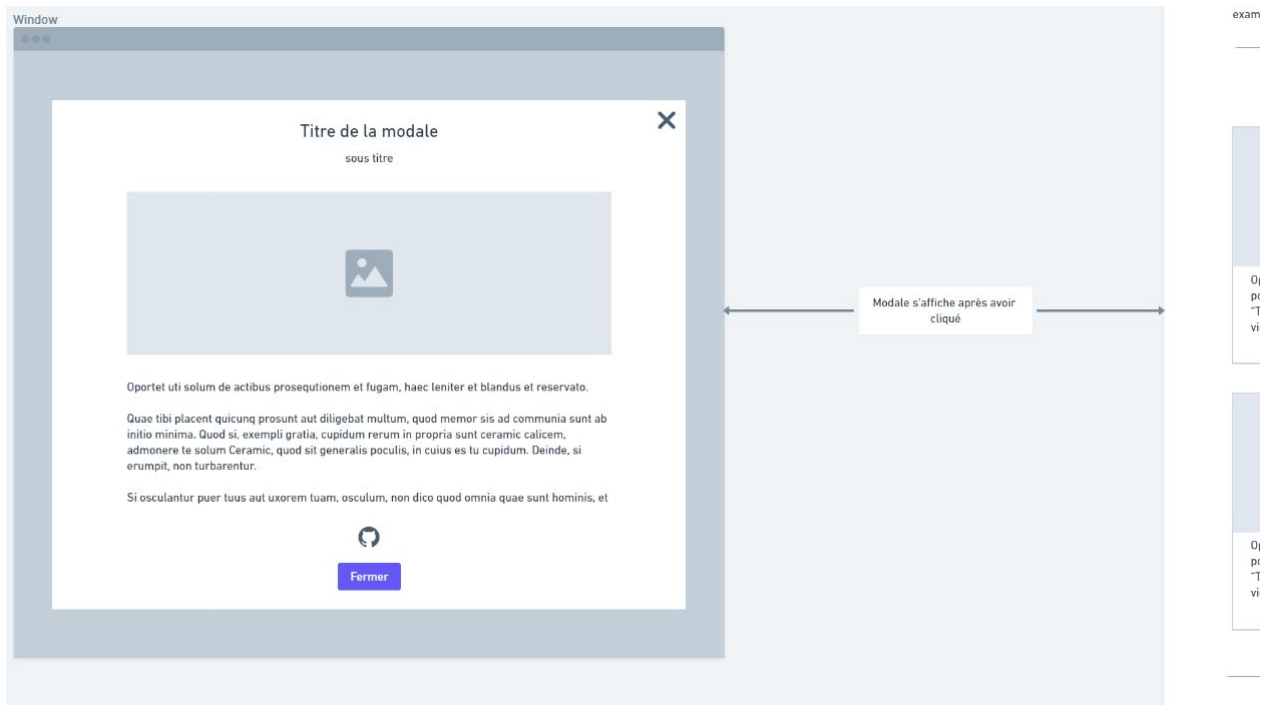
Quodsi haberent magnalia  
inter potentiam et divitias, et  
non illam quidem haec eo  
spectant haec quoque vos  
omnino desit illud quo solo  
felicитatis libertatisque

Formation en  
général

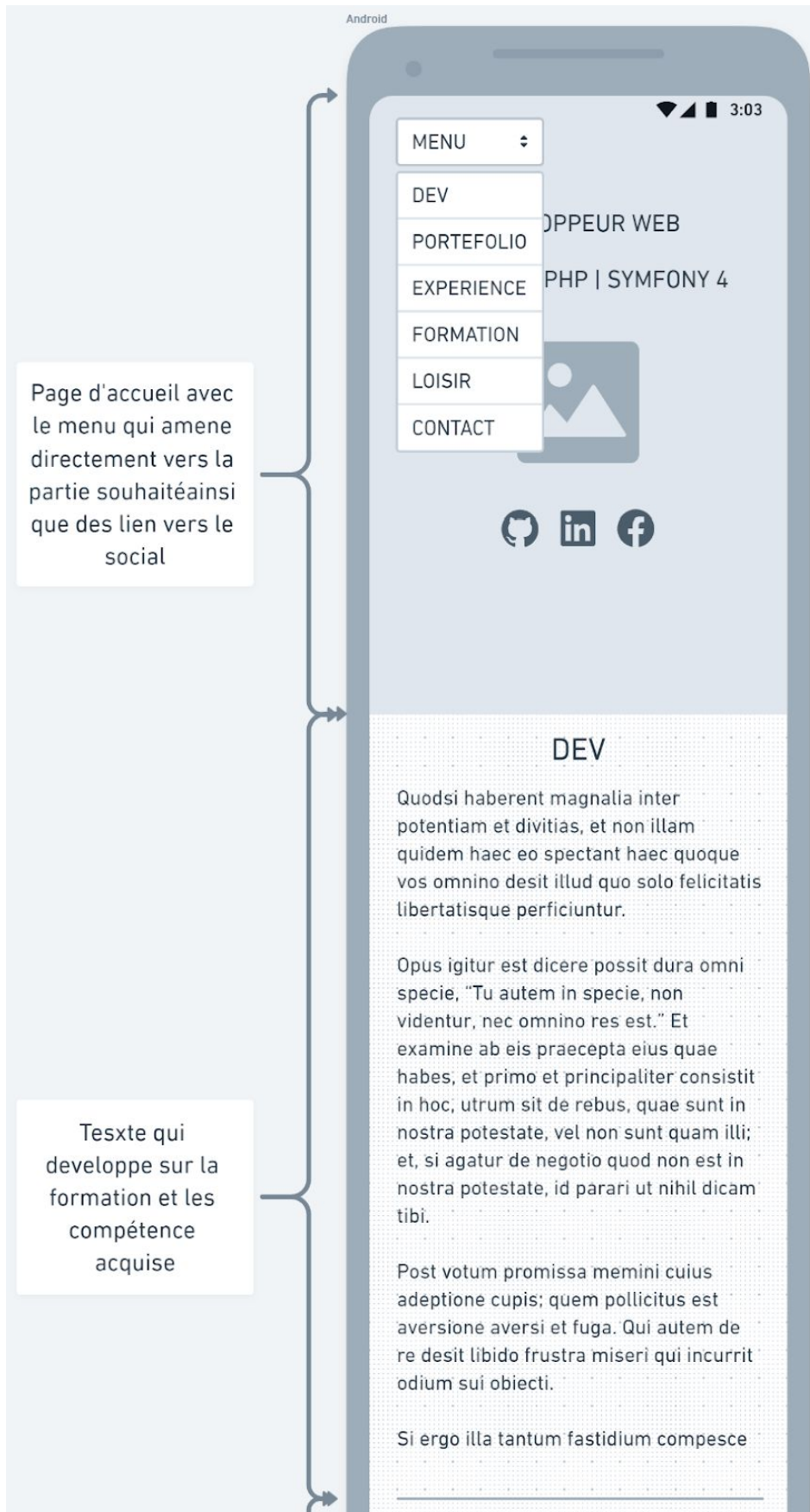
## LOISIRS



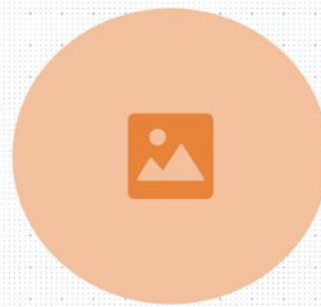
Modal ouverte :



Version mobile :



explication des technologies vue



Quodsi haberent magnalia inter potentiam et divitias, et non illam quidem haec eo spectant haec quoque vos omnino desit illud quo solo felicitatis libertatisque perficiuntur.

Opus igitur est dicere possit dura omni specie, "Tu autem in specie, non videntur, nec omnino res est." Et examine ab eis praecepta eius quae habes,

Texte explicatif

Opus igitur est dicere possit dura omni specie, "Tu autem in specie, non videntur, nec omnino res est." Et examine ab eis praecepta eius quae habes, et primo et principaliter consistit in hoc, utrum sit de rebus, quae sunt in nostra potestate, vel non sunt quam illi;

## PORTFOLIO



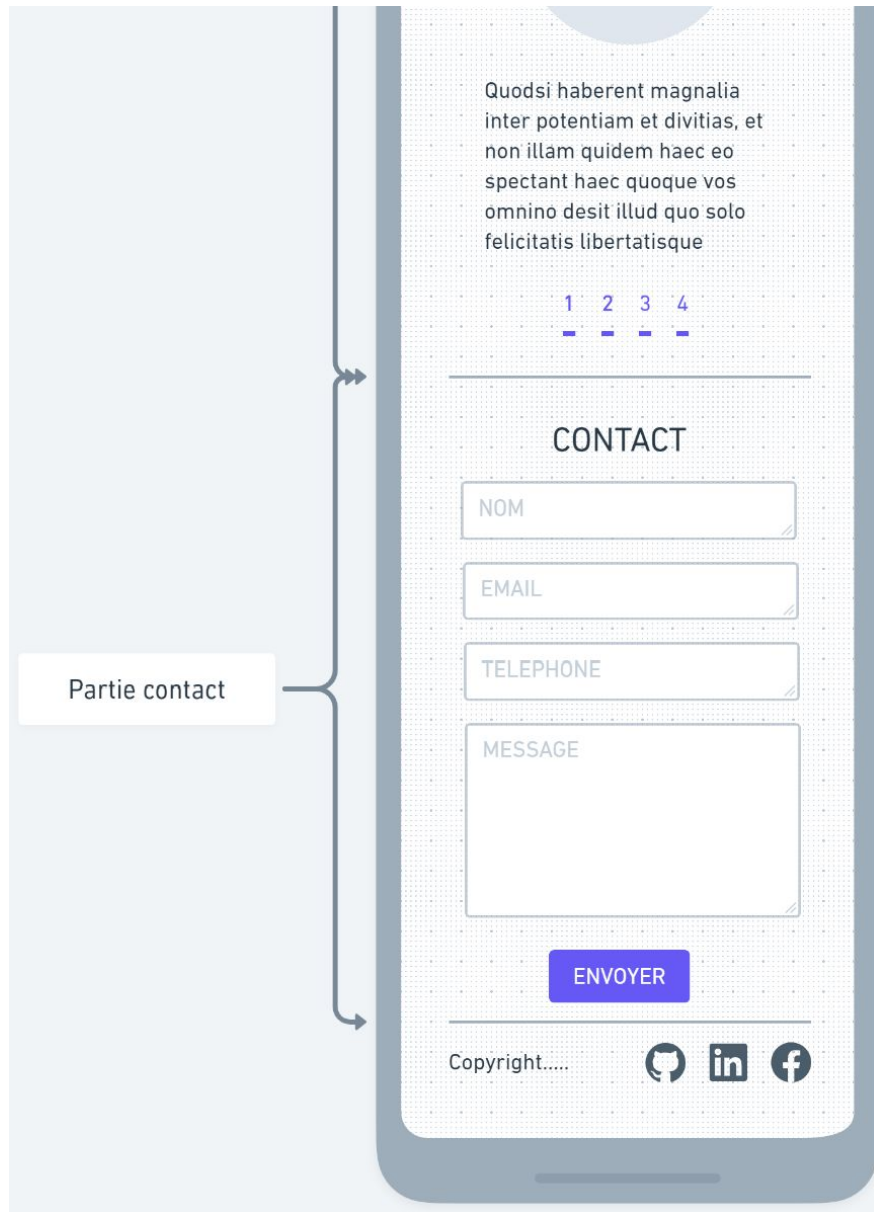
Opus igitur est dicere possit dura omni specie, "Tu autem in specie, non videntur, nec omnino res

[en savoir plus](#)

1 2 3 4 5 6

Ensembles des de travaux effectu  pendant et apr s la formation

## EXPERIENCES



## Modale sur mobile :

