

<i>Nom de naissance</i>	- <i>Sully-Alexandrine</i>
<i>Nom d'usage</i>	- <i>Sully-Alexandrine</i>
<i>Prénom</i>	- <i>Laura</i>
<i>Adresse</i>	- <i>17 Ter route de Marcoussis 91310 MONTLHERY</i>

Titre professionnel visé

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

Activité-type 1 : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité.	p. 5
▸ CP 1 Maquetter une application.	p. 5
▸ CP 2 Réaliser une interface statique et adaptable.	p. 10
▸ CP 3 Développer une interface utilisateur web dynamique.	p. 12
Intitulé de l'activité-type n° 2	p. 15
▸ CP 5 Créer une base de données.	p. 15
▸ CP 6 Développer les composants d'accès aux données.	p. 18
▸ CP 7 Développer la partie back-end d'une application web ou web mobile.	p. 23
Titres, diplômes, CQP, attestations de formation (facultatif)	p. 29
Déclaration sur l'honneur	p. 30
Documents illustrant la pratique professionnelle (facultatif)	p. 31
Annexes (Si le RC le prévoit)	p. 32

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.

CP 1 - Maquetter une application.

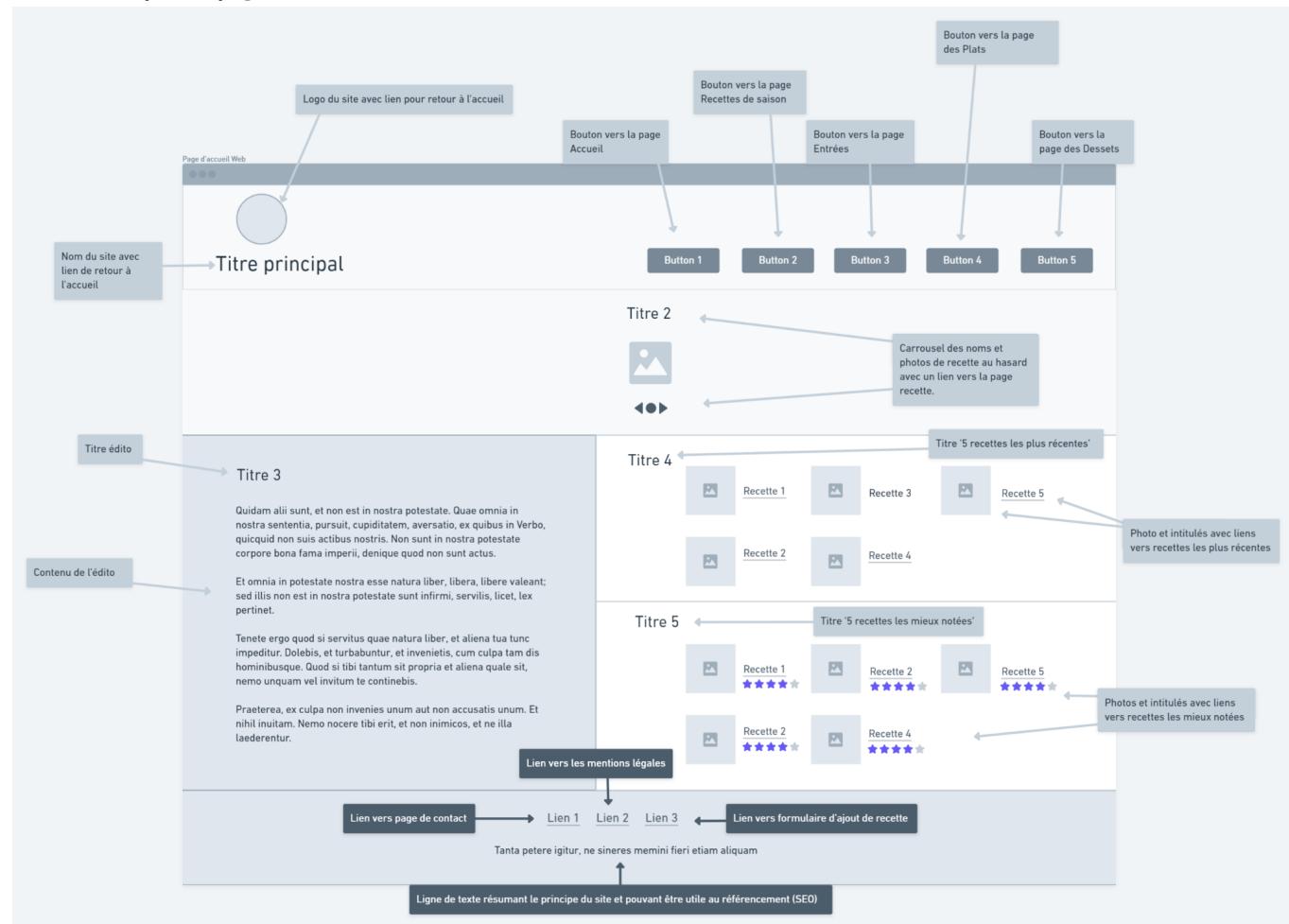
1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Au cours de la formation, nous avons abordé et défini ce que sont les **wireframes** et leur importance pour la préparation d'un projet, car elles permettent de faire un zonage des différentes fonctionnalités des pages d'un site.

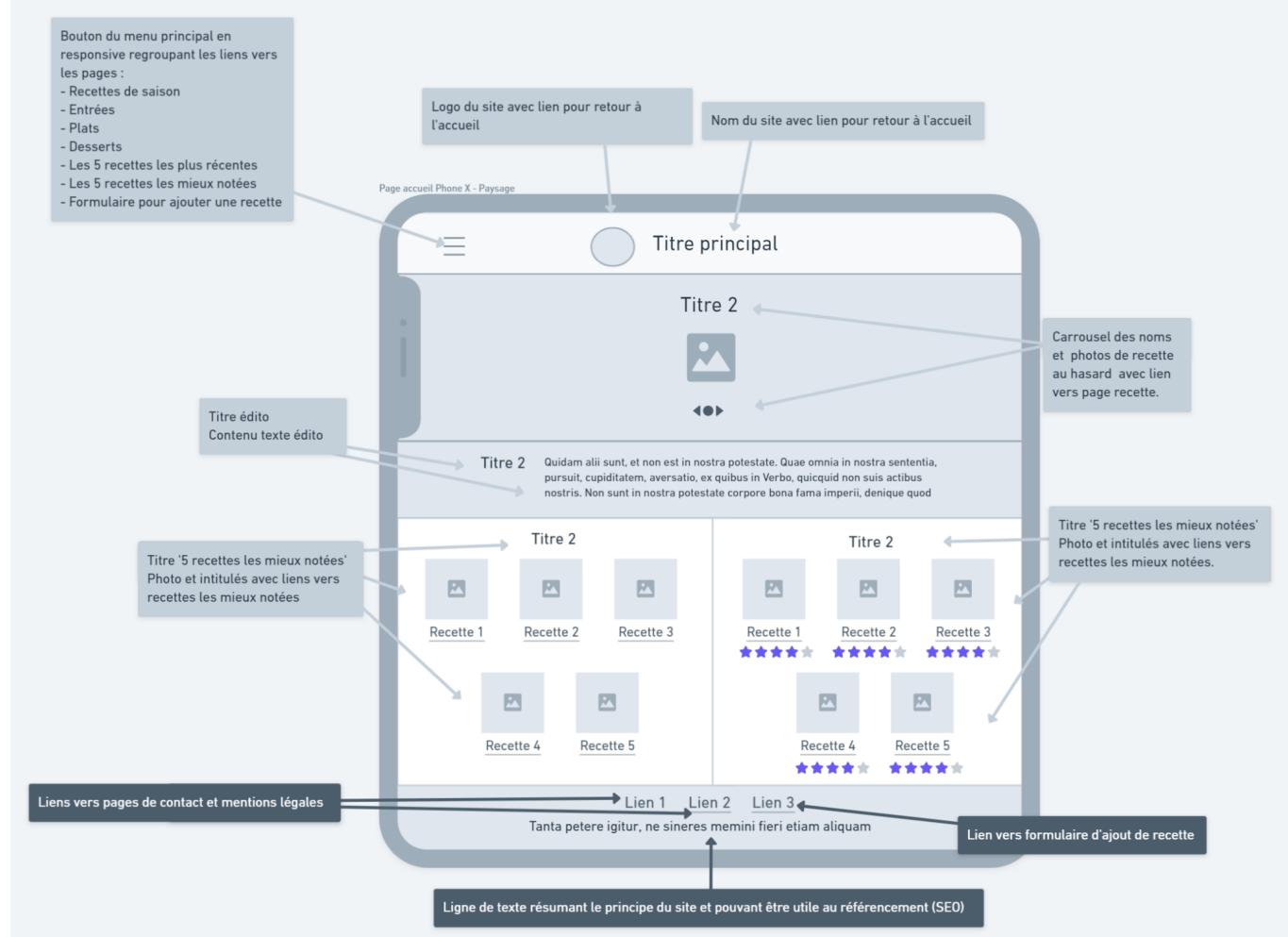
Par la suite, nous avons mis en application ces acquis pour un site en **Responsive Web Design**, regroupant des recettes de cuisine, de manière à pouvoir travailler sur la nature des emplacements des différents composants des pages demandées et de légendier leur fonctionnalité.

Les trois formats attendus : mobile, tablette et desktop.

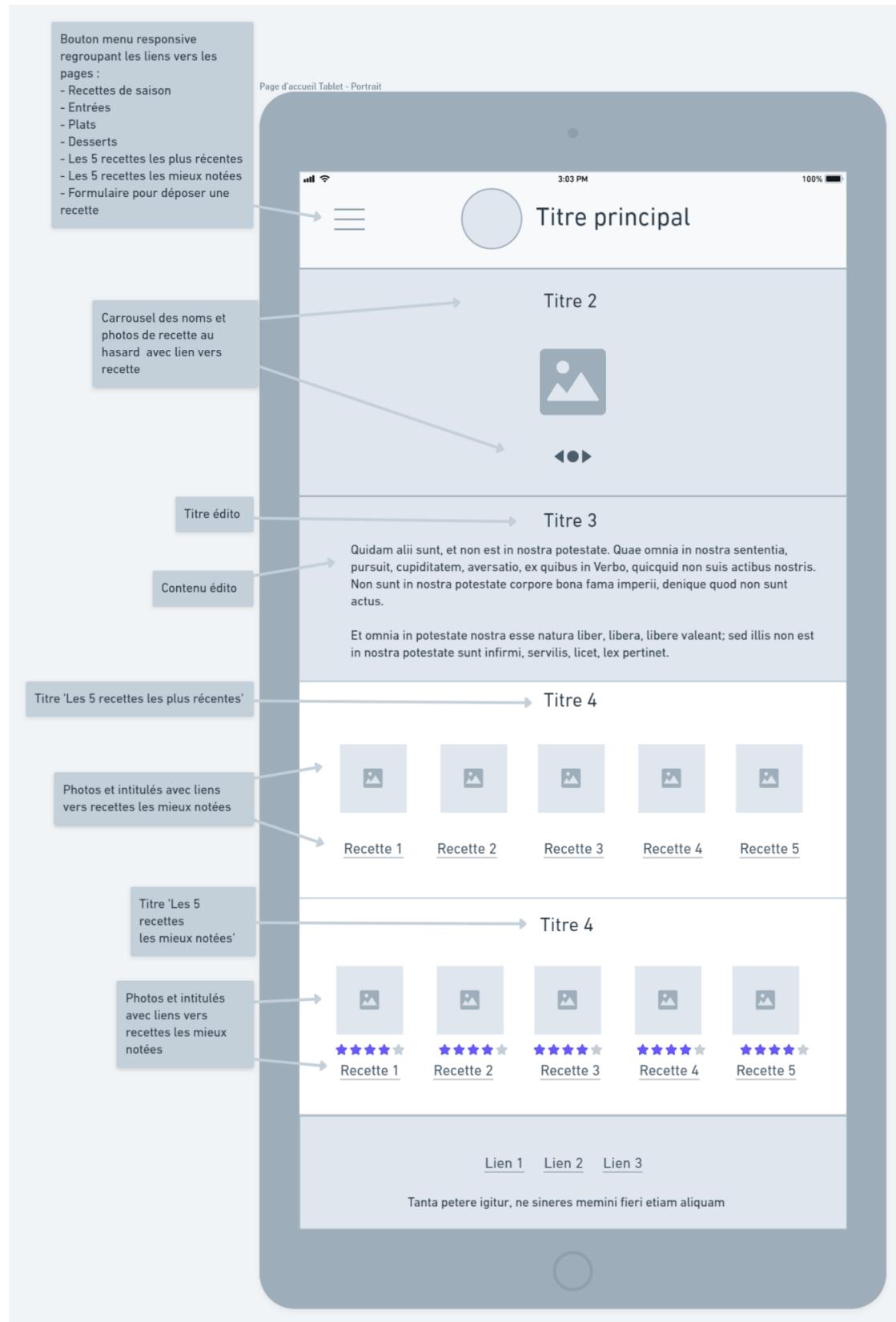
Rendu desktop de la page d'accueil :



Rendu mobile de la page d'accueil :



Rendu tablette de la page d'accueil :



Les users stories :

Description du contenu des fonctionnalités à développer. En tant que {X} je veux {Y} afin de {Z}

User stories

Fiche récap : <https://kourou.oclock.io/ressources/fiche-recap/user-stories/#spécifications>

Accueil

En tant que	Je veux	Afin de (si besoin/nécessaire)
visiteur	pouvoir visualiser le logo sur toutes les pages	accéder intuitivement à la page d'accueil
visiteur	pouvoir accéder à des recettes d'entrées	-
visiteur	pouvoir accéder à des recettes de saison	-
visiteur	pouvoir accéder à des recettes de plats	-
visiteur	pouvoir accéder à des recettes de desserts	-
visiteur	pouvoir visualiser un édito	être informer des nouveautés toutes les semaines
visiteur	pouvoir visualiser une recette nouvelle à chaque chargement de page	-
visiteur	pouvoir accéder à 5 recettes les plus récentes	naviguer plus rapidement sur le site
visiteur	pouvoir accéder à 5 recettes les mieux notées	naviguer plus rapidement sur le site
visiteur	pouvoir accéder à un formulaire de contact	contacter le responsable du site ou tout autre personne lié au site.
visiteur	pouvoir consulter les mentions légales	connaitre la raison sociale du site et me mettre en confiance
visiteur	pouvoir proposer mes recettes	-

Recette

En tant que	Je veux	Afin de (si besoin/nécessaire)
visiteur	pouvoir visualiser la photo de la recette choisi	-
visiteur	pouvoir visualiser la liste des ingrédients	-
visiteur	pouvoir visualiser les étapes de la préparation	-

2. Précisez les moyens utilisés :

Pour la réalisation des wireframes, j'ai utilisé l'outil whimscal.com qui met à disposition un espace de travail facile à prendre en main avec des fonctionnalités prédéfinies, afin d'avoir rapidement un bon rendu.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce challenge.

4. Contexte

Nom de l'entreprise, organisme ou association ➤ *O'clock*

Chantier, atelier, service ➤ *Formation*

Période d'exercice ➤ Du : **28/04/2021** au : **29/04/2021**

5. Informations complémentaires (facultatif)

Lien vers le repository GitHub : [Projet 10' Minutes O'Four](https://github.com/JulietteLafitte/Projet_10_Minutes_OFour) .

Activité-type 1

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

CP 2 - Réaliser une interface web statique et adaptable.

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

Au cours de la saison trois de la formation, nous avons eu une journée de révision avec plusieurs projets dans le but de mettre en pratique les différents langages abordés lors des trois saisons écoulées. Au cours de cette journée de révision, j'ai choisi le projet d'intégration HTML/ CSS et JavaScript. Le challenge était de faire l'intégration d'une maquette de blog en exemple et en bonus, le responsive à créer.

J'ai repris cet exercice de zéro pour les besoins de la CP2.

Extrait du code HTML 'Block 1' (cf: annexe 1)

Extrait du code CSS : (cf: annexe 2)

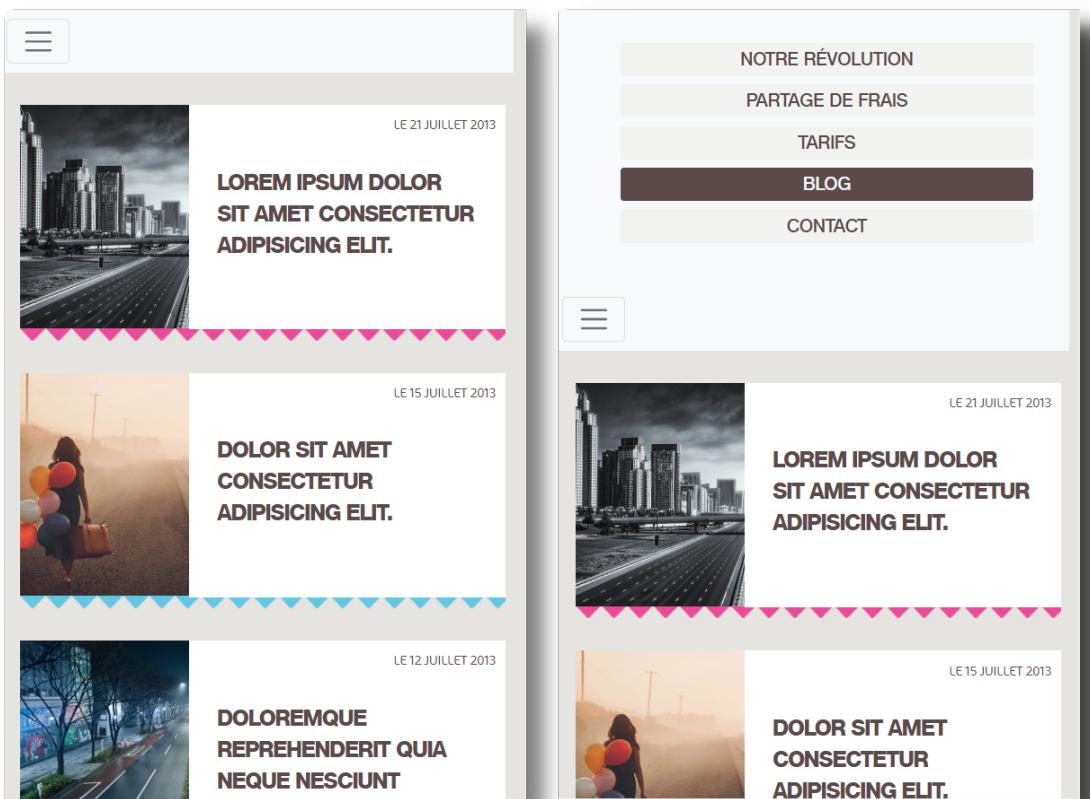
Extrait du code CSS media queries : (cf: annexe 3)

Rendu desktop de la page d'accueil :



The screenshot shows a desktop website layout. At the top, there is a navigation bar with five items: 'NOTRE RÉVOLUTION', 'PARTAGE DE FRAIS', 'TARIFS', 'BLOG' (which is highlighted in a dark box), and 'CONTACT'. To the left of the main content area, there is a vertical pink sidebar featuring a white circular logo with a stylized globe or network pattern. The main content area features a large black and white photograph of a modern city skyline with numerous skyscrapers and a multi-lane highway in the foreground. In the bottom right corner of the image, the date 'LE 21 JUILLET 2013' is visible. To the right of the image, there is a block of text in all-caps: 'LOREM IPSUM DOLOR SIT AMET CONSECTETUR ADIPISCING ELIT.'. Below this, there is a paragraph of placeholder text in French: 'Lorem ipsum dolor sit amet consectetur adipiscing elit. Provident molestias aperiam, maiores incidentum ab quae a dolor vero ipsum quos labore tempora exercitationem ipsam velit consectetur deleniti dolorem maxime iste! Est, labore! Sunt excepturi voluptatibus tenetur optio quisquam ipsa eius blanditlisis, voluptatem parlatur maxime provident vitae expedita quas quaerat nam. In nisi minus, ipsa maiores aspernatur assumenda nulla voluptatum? Hic! Lorem ipsum dolor sit amet, consectetur adipiscing elit. Illo animi exercitationem perspiciatlisis aliquid et ab.' At the bottom of the content area, there is a small 'CATÉGORIE 3' label and two social media icons for Facebook and Twitter.

Rendu mobile de la page d'accueil :



2. Précisez les moyens utilisés :

J'ai utilisé les balises sémantiques HTML pour l'intégration, le CSS pour appliquer le graphisme demandé et la micro-interaction des boutons de navigation.

Les media queries pour le responsive et le framework bootstrap.com pour la barre de navigation principale.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce challenge de révision.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ **O'clock**

Chantier, atelier, service ▶ *Formation challenge révision.*

Période d'exercice ▶ Du : **09/05/2021** au : **25/05/2021**

5. Informations complémentaires (facultatif)

Lien vers le repository GitHub : [Challenge de révision html-css](https://github.com/Challenge-revision/html-css).

Activité-type 1

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

CP 3 - Développer une interface utilisateur web dynamique.

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

Au cours de la saison trois, consacrée au langage **JavaScript**, nous avons effectué un atelier en Pair Programming sur un projet intitulé oLogin.

Le but de cet exercice était de faire interagir un formulaire de connexion avec les actions d'un utilisateur, mettre en place des critères de validation des champs. Ici, lors de la saisie des éléments de connexion, l'utilisateur doit obligatoirement saisir trois caractères minimum et si erreur, l'envoi du formulaire est bloqué et les champs en erreur sont signalés par un code couleur ainsi que l'affichage d'un message afin que l'utilisateur les corrige et valide sa connexion.

Lors de cet atelier au cours de la formation, je n'ai pas codé, alors j'ai repris de zéro cet exercice pour les besoins de la CP3.

Formulaire de connexion avec les champs en erreur en rouge et le message signalétique.

The form is titled "Connexion". It contains two input fields: "Identifiant" (Identifier) with the value "a" and "Mot de passe" (Password) with the value ". ". Below each field is a validation message: "Obligatoire - doit contenir au minimum 3 caractères". A red button in the center says "Merci de renseigner 3 caractères minimum". At the bottom right is a black button labeled "Connexion".

Extrait de code JavaScript :

Dans cet extrait de code, j'utilise la méthode **preventDefault** dans la fonction handler **formIsValid**, ce qui me permet d'empêcher le comportement par défaut du navigateur lorsque la saisie utilisateur ne correspond pas aux critères de validité. Ces derniers sont définis dans la fonction **inputGetValue**. Elle cible un input récupère la valeur saisie et fait interagir le **DOM (Document Object Model)** en modifiant la classe avec la propriété **classList** de **.valid** en vert et **d.invalid** en rouge puis le message d'erreur est affiché dans la div **#errors** grâce à la propriété **innerHTML**.

Si la saisie des informations est correct ou en cas de rectification des erreurs par l'utilisateur la div **#errors** se vide ainsi que le tableau **errorArea** grâce à la fonction **errorClear**, cela annule la méthode **preventDefault** et valide la soumission du formulaire.

```
// Ici si le tableau errorArea n'est pas vide alors j'empêche le comportement par défaut de l'événement submit
// et j'affiche un message d'erreur dans la div errors
formIsValid: function(evt) {
    if(app.errorArea.length > 0)
    {
        // https://developer.mozilla.org/fr/docs/Web/API/Event/preventDefault
        evt.preventDefault();
        app.errors.innerHTML += '<p class="error"> Merci de renseigner 3 caractères minimum </p>';
        console.log( app.errors);
        console.error('il y a des erreurs');
    }
},
// Je récupère la saisie et je la valide ou non avec le critère des 3 caractère minimum et j'applique le changement
// de style css en ajoutant ou en supprimant les class valid et invalid.
inputGetValue: function(inputTarget) {
    let inputValue = inputTarget.value;
    console.log(inputValue);

    if( inputValue.length >= 3)
    {
        inputTarget.classList.add('valid');
        inputTarget.classList.remove('invalid');
        app.errorClear();
    }
    else {
        inputTarget.classList.add('invalid');
        inputTarget.classList.remove('valid');
        app.errorArea [0] = inputValue;
        console.log(app.errorArea);
    }
},
},
```

```
// Ici j'efface le message d'erreur de la div errors et je vide le tableau errorArea de la saisie
// erronée de l'utilisateur.
errorClear: function() {
    // https://developer.mozilla.org/fr/docs/Web/API/Element/innerHTML
    app.errors.innerHTML = '';
    console.log(app.errors);

    app.errorArea.length = 0;
    console.log(app.errorArea.length);
}
```

Formulaire de connexion avec les champs valides en vert

Connexion

Identifiant

Obligatoire - doit contenir au minimum 3 caractères

Mot de passe

Obligatoire - doit contenir au minimum 3 caractères

Connexion

2. Précisez les moyens utilisés :

J'ai utilisé l'éditeur de code Visual Studio Code et les langages HTML, CSS, JAVASCRIPT et le DOM interface de programmation.

3. Avec qui avez-vous travaillé ?

Lors de l'atelier au cours de la formation en Pair Programming avec un camarade de promotion et par la suite seule.

4. Contexte

Nom de l'entreprise, organisme ou association ➤ *O'clock*

Chantier, atelier, service ➤ *Atelier*

Période d'exercice ➤ Du : **26/05/2021** au : **27/05/2021**

5. Informations complémentaires (*facultatif*)

Lien vers le repository : [Atelier oLogin](#).

Activité-type 2

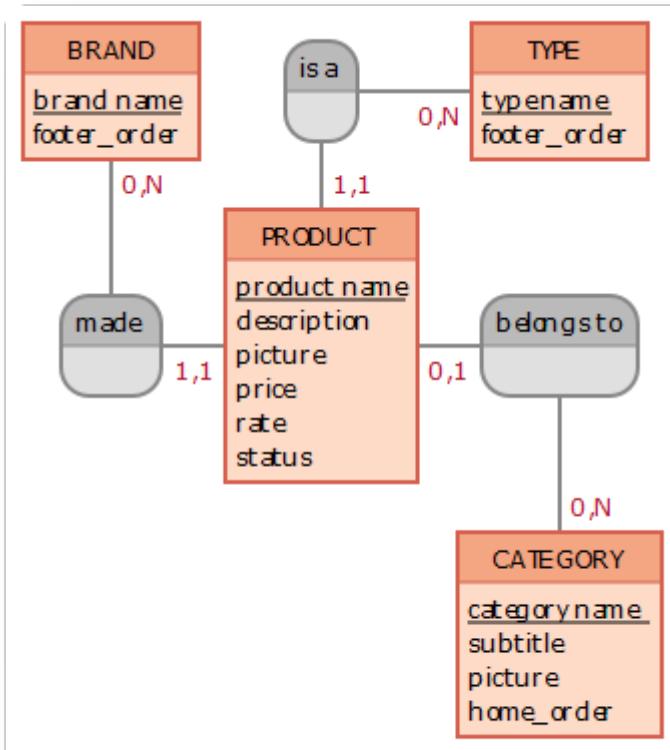
Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.

CP 5 - Créer une base de données.

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

Lors de la saison cinq de la formation, nous avons abordé le monde du back via l'Architecture en commençant par la modélisation d'une **base de données**. Dans cette optique, nous avons appris à créer un **dictionnaire de données** et un **MCD** (Modèle Conceptuel de Données).

Tout au long de cette saison, nous avons fait évoluer, sous forme de challenge, un projet fil rouge nommé oShop, site de e-commerce, regroupant plusieurs marques de chaussures, au cours duquel nous avons pu modéliser la **base de données**.



Le modèle conceptuel de données.

En fonction des éléments fournis, je l'ai découpé en plusieurs entités représentées par un rectangle, quatre ici : BRAND, TYPE, PRODUCT et CATEGORY. Pour ces différentes entités j'ai défini des attributs propres à chacune d'elles.

Ensuite, j'ai déterminé les associations représentées par une rectangle arrondis, avec un lien verbalisé entre deux entités, et les cardinalités, elles représentent la quantité minimum(gauche), et maximum(droit) qui peut exister entre deux entités A et B.

Association made :

Une marque peut être créée sans avoir de produit associé et pour une marque donnée je peux avoir plusieurs produits associés => 0..N.

Un produit ne peut être créé sans au moins une marque associée et un produit ne peut pas être fabriqué par plusieurs marques => 1..1.

Association belong to :

Un produit peut être créé sans appartenir à une catégorie et un produit donné ne peut appartenir qu'à une seule catégorie. => 0..1.

Une catégorie peut être créée sans être associée à un produit et une catégorie peut avoir plusieurs produits => 0..N.

A l'aide du **MCD**, j'ai pu déterminer le nombre de **tables** à créer également, déterminer la **table maîtresse** grâce aux **discriminants** des autres tables BRAND, TYPE et CATEGORY présents dans la table PRODUCT.

BRAND (brand name, footer_order)

TYPE (type name, footer_order)

PRODUCT (product name, description, picture, price, rate, status, brand name, type name, category name)

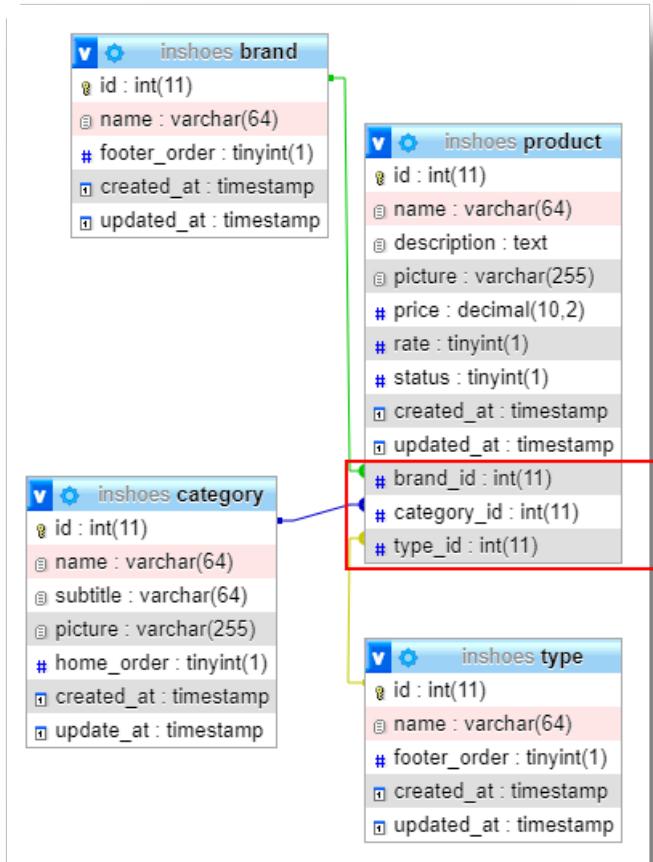
CATEGORY (category name, subtitle, picture, home_order)

le **dictionnaire de données**, qui m'a permis de déterminer les métadonnées pour chaque propriété de table (**échantillon du DDD**).

Dictionnaire de données

Produits (product)

Champ	Type	Spécificités	Description
id	INT	PRIMARY KEY, NOT NULL, UNSIGNED, AUTO_INCREMENT	L'identifiant de notre produit
name	VARCHAR(64)	NOT NULL	Le nom du produit
description	TEXT	NULL	La description du produit
picture	VARCHAR(128)	NULL	L'URL de l'image du produit
price	DECIMAL(10,2)	NOT NULL, DEFAULT 0	Le prix du produit
rate	TINYINT(1)	NOT NULL, DEFAULT 0	L'avis sur le produit, de 1 à 5
status	TINYINT(1)	NOT NULL, DEFAULT 1	Le statut du produit (1=dispo, 2=non dispo)
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	La date de création du produit
updated_at	TIMESTAMP	NULL	La date de la dernière mise à jour du produit
brand	entity	NOT NULL	La marque (autre entité) du produit
category	entity	NULL	La catégorie (autre entité) du produit
type	entity	NOT NULL	Le type (autre entité) du produit



Le modèle physique de données issu SGBD (Système de Gestion de Base de Données) relationnel MySQL.

Il crée un identifiant (id) unique à chaque table qui permettra de les identifier afin de les manipuler dans le code PHP.

Il met également en évidence les **relations** entre les **tables** et crée, au besoin, une clé étrangère lorsque l'une des cardinalités max. vaut 1.

Comme on peut le voir dans le MCD avec les 3 associations de l'entité PRODUCT, les cardinalités max valent toutes 1 donc les clés étrangères seront créées dans la table PRODUCT

2. Précisez les moyens utilisés :

J'ai utilisé l'application [mocodo](#) pour le MCD, l'éditeur de texte Visual Studio Code pour le dictionnaire de données et le gestionnaire phpMyAdmin pour le MLD.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur les différents challenges de ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ➤ **O'clock**

Chantier, atelier, service ➤ **Formation**

Période d'exercice ➤ Du : **04/11/2020** au : **16/11/2020**

5. Informations complémentaires (facultatif)

Lien vers le repository : [oShop saison 5.](#)

Activité-type 2

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

CP 6 - Développer les composants d'accès aux données.

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

En fin de saison cinq, nous avons eu l'occasion de mettre en pratique l'architecture par le biais d'un parcours pendant lequel, j'ai pu mettre en place le **design pattern MVC** (Modèle Vue contrôleur) la **POO** (Programmation Orientée Objet) et l'instanciation de l'objet **PDO** (PHP Data Objects) pour la connexion à la base de données.

Au cours de ce parcours, j'ai créé une base de données et j'ai configuré le fichier **config.ini**.



```
DB_HOST=localhost  
DB_USERNAME=root  
DB_PASSWORD=  
DB_NAME=sonic
```

extrait du code : fichier config.ini

Grâce à ce fichier point d'accès, je peux déclarer une connexion à ma base de données avec les paramètres clé=valeur sous la forme d'un tableau associatif : serveur 'localhost', un nom d'utilisateur 'root', le cas échéant un mot de passe et le nom de ma base de données 'sonic' que j'ai préalablement créé dans mon **SGBD** (Système de Gestion de Bases de Données) relationnel **MySQL**.

Donc, il faut créer une instance de **PDO** en passant au constructeur trois paramètres qui constituent la chaîne de connexion.

Une base de données **MySQL**, le nom du serveur **localhost** le nom de la base de données, le jeu de caractères, le nom de l'utilisateur et le mot de passe.

Ensuite, l'objet **PDO** permettra à l'aide de PHP d'interroger la **BDD** en analysant le fichier **config.ini**.

extrait du code : classe utilitaire Database.php

```
<?php

namespace Sonic\Utils;

use PDO;

// Retenir son utilisation => Database::getPDO()
// Design Pattern : Singleton
class Database {
    /** @var PDO */
    private $dbh;
    private static $_instance;
    private function __construct() {
        // Récupération des données du fichier de config
        // la fonction parse_ini_file parse le fichier et retourne un array associatif
        $configData = parse_ini_file(__DIR__.'/../config.ini');

        try {
            $this->dbh = new PDO(
                "mysql:host={$configData['DB_HOST']};dbname={$configData['DB_NAME']};charset=utf8",
                $configData['DB_USERNAME'],
                $configData['DB_PASSWORD'],
                array(PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING) // Affiche les erreurs SQL à l'écran
            );
        }
    }
}
```

Vient ensuite la mise en place du **MVC** géré avec la **POO**, tout particulièrement les **Models** qui dans ce **design pattern Active Record**, font partie des composants qui vont me permettre d'accéder aux données de la **BDD** en faisant appel à **MySQL** et à **PDO** via des méthodes.

Donc, ici et avec les éléments du **dictionnaire de données** fourni, j'ai créé deux fichiers modèles 'Character' et 'Types' liés à la base de données et un autre 'CoreModel'. Il a permis de mutualiser les propriétés et les méthodes communes aux deux classes 'Character' et 'Types' en utilisant l'**héritage**, un des avantages de la **Programmation Orientée Objet**.

extrait du code : fichier CoreModel.php

```
<?php

namespace Sonic\Models;

class CoreModel {

    protected $id;
    protected $name;
    protected $created_at;
    protected $updated_at;

    /**
     * Get the value of the entity id
     * @return int
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Nous sommes ici dans un docblock
     */
    //
    public function getName() // get récupérer
    {
        return $this->name;
    }

    /**
     * Set the value of name
     * @return self
     */
    public function setName($name) // set modifier en écriture
    {
        $this->name = $name;

        return $this;
    }

    /**
     * Get the value of created_at
     */
    public function getCreatedAt()
    {
        return $this->created_at;
    }

    /**
     * Get the value of updated_at
     */
    public function getUpdatedAt()
    {
        return $this->updated_at;
    }
}
```

Dans le modèle, j'utilise le principe **Active Record** qui me permet de créer une méthode qui accèdera aux informations de ma **BDD**, Dans laquelle, je récupère la connexion à ma **BDD** avec **PDO** défini dans la **classe utilitaire Database**.

extrait du code : model Character.php

```
● ● ●

// Méthode retournant la liste de tous les personnages.
public function findAllCharacters() {

    // En premier se connecter à la BDD en récupérant de connexion PDO qui se trouve dans le fichier
    DataBase
    $pdoDBConnexion = Database::getPDO();
    dump($pdoDBConnexion);

    // On écrit notre requête sql qui nous permettra d'afficher les différentes infos souhaitées des
    personnes.
    $sql = 'SELECT `character`.`id`, `character`.`name`, `character`.`description`,
    `character`.`picture`, `type`.`name`
    AS`type_name` FROM `character` INNER JOIN `type` ON `type`.`id` = `character`.`type_id` WHERE
    `character`.`type_id` ORDER BY `character`.`name` ';
    // dump($sql);
```

Quand ma connexion est validée, je crée ma requête **sql**, qui me permettra de sélectionner les éléments que je veux afficher à ma **vue** via le **contrôleur**. Il sert de lien entre le fichier **index.php**, dans lequel j'initialise les routes que j'ai associées aux méthodes définies dans les **contrôleurs**.

J'exécute les données trouvées par ma requête **sql**, avec **query**. Elle me retourne un jeu de résultats sous un objet **PDOStatement** (**requête préparée**).

extrait du code : model Character.php

```
● ● ●

// Exécution de la requête avec la connexion
$pdoStatement = $pdoDBConnexion->query($sql);
```

Pour finir, je récupère et je retourne la ligne de résultat sous la forme d'un tableau associatif.

extrait du code : model Character.php



```
// Récupération des résultats
$charactersList = $pdoStatement->fetchAll(PDO::FETCH_ASSOC);
// dump($charactersList);

return $charactersList;
}
```

2. Précisez les moyens utilisés :

J'ai utilisé l'éditeur de texte Visual Studio Code pour écrire mon code PHP, HTML et CSS et le gestionnaire MySQL pour la BDD.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce parcours.

4. Contexte

Nom de l'entreprise, organisme ou association ➤ *O'clock*

Chantier, atelier, service ➤ Formation

Période d'exercice ➤ Du : **14/11/2020** au : **17/11/2020**

5. Informations complémentaires (*facultatif*)

Lien vers le repository GitHub : [Parcours saison 5.](#)

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

Activité-type 2

CP 7 - 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 :

Lors du mois de spécialisation **Symfony**, nous avons développé un projet fil rouge nommé MovieDB, un site d'agence de production de films.

Le but était, de faire évoluer ce site au cours des différents challenges, chacun ayant pour but d'appliquer une nouvelle notion du framework. Nous avons pu voir que Symfony utilise un **ORM (Object Relational Mapping)** nommé **Doctrine** qui fait le lien entre les entités et les classes du projet gérées en **POO** et les tables gérées sous une base de données relationnelle. **DBAL (Database Abstraction Layer)**, il gère les méthodes de base **find(\$id)**, **findAll()**, **findBy()** et **findOneBy()** par le biais du **Repository** et la connexion **PDO** via le fichier point d'accès **'.env'**.

La première étape de ce projet était, la création de la partie **front-end** qui consistait à afficher les listes des films, afin de pouvoir les consulter, donc j'ai mis en place le **MVC** avec des routes accessibles par tout utilisateur lambda.

La deuxième étape était la création du **back-end** avec un contrôle des accès en base de données.

Donc, dans un premier temps, j'ai créé les différents contrôleur avec les nouvelles routes de l'administration ainsi que les méthodes de gestion des formulaires, pour pouvoir afficher, ajouter, éditer et supprimer dans le but de gérer les modifications des éléments du site en base de données.

extrait du code ; contrôleur Movie partie admin méthode d'édition.

```
/*
 * @Route("/admin/movie/edit/{id}", name="admin_movie_edit", methods={"GET", "POST"}, requirements={"id"="\d+"})
 */
public function edit(Movie $movie, Request $request): Response
{
    $form = $this->createForm(MovieType::class, $movie);

    $form->handleRequest($request);

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

        $movie->setUpdatedAt(new \DateTime());
        $entityManager->flush();

        $this->addFlash('success', 'Movie ` ' . $movie->getTitle() . ' ` a bien été mis à jour !');

        return $this->redirectToRoute('admin_movie');
    }

    return $this->render('back/movie/edit.html.twig', [
        'form' => $form->createView(),
        'movie' => $movie,
    ]);
}
```

extrait du code : Formulaire Movie partie admin

```
<?php

namespace App\Form;

use App\Entity\Movie;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\ChoiceType;
use Symfony\Component\Form\Extension\Core\Type\DateType;
use Symfony\Component\Form\Extension\Core\Type\FileType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Validator\Constraints\NotBlank;
use Symfony\Component\Validator\Constraints\Type;

class MovieType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('title', null,
                [
                    'label' => 'Titre du film : ',
                    'constraints' => [
                        new NotBlank(['message' => 'Le champ "Movie" est vide'])
                    ]
                ])
            ->add('genres', null, [
                'label' => 'Genres associés : ',
                'expanded' => false, // on passe à true si on veut afficher une liste de case à cocher
            ])
            ->add('image', null,
                [
                    'label' => 'Affiche du film : '
                ])
        ;
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => Movie::class,
        ]);
    }
}
```

Dans un second temps j'ai traité la partie authentification des utilisateurs, quand il se connecte de même que les autorisations d'accès à certaines fonctionnalités du **back-end** .

Donc, pour la sécurité j'ai utilisé le **bundle maker 'user'** qui instaure, par défaut, à tout nouvel utilisateur un rôle USER, il permet de créer une gestion d'authentification d'un éventuel utilisateur aux données enregistrées en base.

Ensuite, j'ai utilisé le **bundle maker 'auth'** pour générer un formulaire de login sécurisé avec une gestion de la route d'authentification, récupération des identifiants: à partir des identifiants, récupérer l'utilisateur, vérifier que l'utilisateur correspond bien aux identifiants enregistrés en base et par défaut le rediriger vers la route de la page d'accueil du **back-office** : 'admin_movie'.

extrait du code : LoginAuthenticator.php redirection après connexion utilisateur

```
● ● ●

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, string
$providerKey)
{
    // ce if permet de rediriger l'utilisateur sur la page qu'il a demandé à l'origine
    if ($targetPath = $this->getTargetPath($request->getSession(), $providerKey)) {
        return new RedirectResponse($targetPath);
    }

    return new RedirectResponse($this->urlGenerator->generate('admin_movie'));
    // throw new \Exception('TODO: provide a valid redirect inside '.__FILE__);
}
```

Par la suite et pour déterminer les accès à certaines fonctionnalités du site, j'ai créé un rôle administrateur, auquel est attribué l'accès à différentes routes lui permettant d'afficher, éditer et supprimer n'importe quel utilisateur du **back-office**, mais aussi la gestion d'ajout de films.

extrait du code : fichier security.yaml

```
● ● ●

# 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: ^/admin/user/edit, roles: ROLE_USER }
    - { path: ^/admin/[^/]+/(new|edit|delete), roles: ROLE_ADMIN }
    - { path: ^/admin/user, roles: ROLE_ADMIN }
    - { path: ^/admin/, roles: ROLE_USER }
    # - { path: ^/profile, roles: ROLE_USER }
```

Pour finir, j'ai mis en place un menu en fonction de l'utilisateur authentifié.

Lors de la visite d'un anonyme, le site affiche la page d'accueil avec la liste des films et un bouton de connexion qui donnera accès à la partie administration du site.

rendu du menu ; visiteur anonyme



Donc, j'ai utilisé la variable '`app.user`' disponible dans le moteur de templates Twig. Elle me permet de poser une condition en fonction de l'objet en session, donc de l'utilisateur authentifié et de restreindre les accès à certains boutons du menu en fonction du rôle connecté.

extrait du code : base.html.twig condition app.user

```
● ● ●
{% if app.user %}

    <nav class="navigation">
        <ul class="nav-principal">
            <li class="nav-principal-item">
                <a class="nav-principal-link" {% block menu_movie %} {% endblock %}" href="{{ path('admin_movie') }}>Films</a></li>

            <li class="nav-principal-item">
                <a class="nav-principal-link" {% block menu_genre %} {% endblock %}" href="{{ path('admin_genre') }}>Genres</a></li>

            <li class="nav-principal-item">
                <a class="nav-principal-link" {% block menu_person %} {% endblock %}" href="{{ path('admin_person') }}>Acteurs</a></li>
```

En utilisant l'annotation `is_granted` de **Symfony** j'ai restreint l'accès au bouton utilisateur du menu qui permet de gérer la liste des utilisateurs, au rôle d'administrateur.

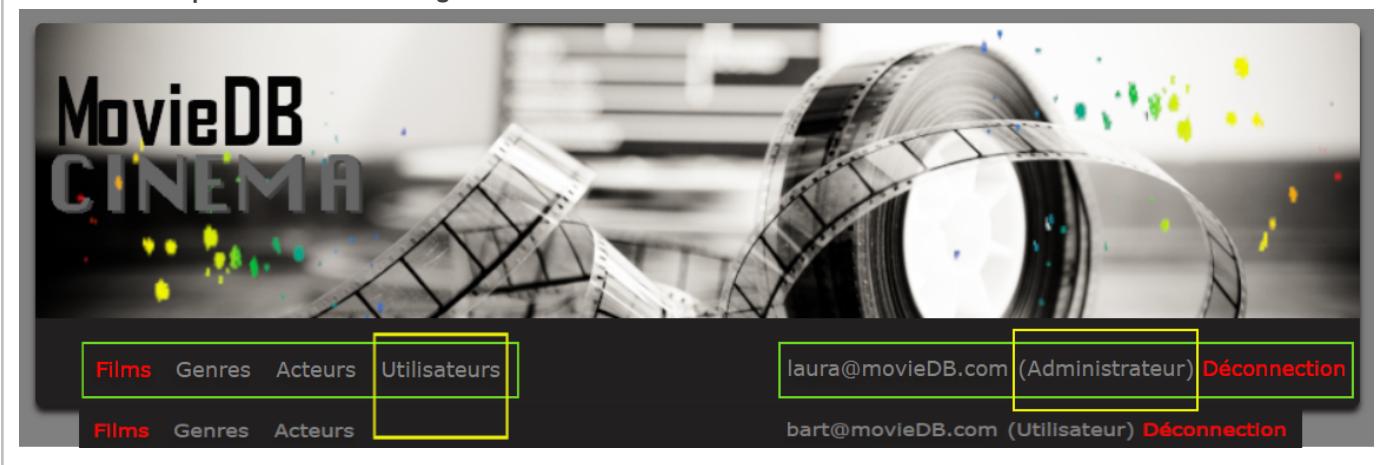
extrait du code : `base.html.twig` condition `is_granted`

```
{% if is_granted('ROLE_ADMIN') %}  
    <li class="nav-principal-item">  
        <a class="nav-principal-link" href="{{ path('admin_user') }}><span class="user-red">Utilisateurs</span></a></li>  
    {% endif %}
```

Pour pouvoir afficher le rôle dans le menu après la connexion de l'utilisateur j'ai ajouté dans l'entité User une fonction `getDisplayRole`. Elle permet d'afficher le rôle en fonction de l'utilisateur connecté.

```
/**  
 * @see UserInterface  
 */  
public function getDisplayRole(): string  
{  
    if (in_array("ROLE_ADMIN", $this->roles))  
    {  
        return "Administrateur";  
    }  
    else  
    {  
        return "Utilisateur";  
    }  
    return "";  
}
```

rendu du menu après connexion des usagers



2. Précisez les moyens utilisés :

Symfony, Twig et Doctrine.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur les challenges

4. Contexte

Nom de l'entreprise, organisme ou association ➤ *O'clock*

Chantier, atelier, service ➤ *Formation*

Période d'exercice ➤ Du : *11/01/2021* au : *22/01/2021*

5. Informations complémentaires (*facultatif*)

Lien vers le repository GitHub : [MovieDB](#).

Titres, diplômes, CQP, attestations de formation

(facultatif)

Déclaration sur l'honneur

Je soussigné(e) [Laura Sully-Alexandrine]

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 Monthéry le

28 juin 2021

pour faire valoir ce que de droit.

Signature :



Documents illustrant la pratique professionnelle

(*facultatif*)

Intitulé
Cliquez ici pour taper du texte.

ANNEXES

Annexe 1 extrait du code : challenge révision-css-blog - index.html

```
● ● ●

<main id="container">
    <!-- BLOCK 1 -->
    <article id="one">
        <section class="left"><a href="#"></a></section>

        <section class="left-mobile"><a href="#"></a></section>

        <section class="right">
            <aside class="right-date">
                <time datetime="2013-07-21">le 21 juillet 2013</time>
            </aside>

            <aside class="right-article">
                <h2 class="right-title"><a class="right-title-link" href="#"> Lorem ipsum dolor
sit amet consectetur adipisicing elit.</a></h2>
                <p class="right-content">
                    Lorem ipsum dolor sit amet consectetur adipisicing elit. Provident molestias
aperiam, maiores incidunt ab quae a dolor vero ipsum quos labore tempora exercitationem ipsam velit
                    consectetur deleniti dolorem maxime iste!
                    Est, labore! Sunt excepturi voluptatibus tenetur optio quisquam ipsa eius
                    blanditiis, voluptatem pariatur maxime provident vitae expedita quas quaerat nam. In nisi minus, ipsa
                    maiores aspernatur assumendum nulla voluptatum? Hic!
                    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Illo animi
                    exercitationem perspiciatis aliquid et ab.
                </p>
            </aside>

            <aside class="category-social">
                <ul class="category-list">
                    <li class="category-item"><a class="category-link" href="#">catégorie 3</a>
                </ul>
                <ul class="social-list">
                    <li class="social-item"><a class="social-link" href="#"></a>
                </li>
                    <li class="social-item"><a class="social-link" href="#"></a></li>
                </ul>
            </aside>
        </section>
    </article>
```

Annexe 2 extrait du code : challenge révision-css-blog - style.css

```
● ● ●

#one,
#three {
    background-image: url('../images/pink-triangle.png');
}

#two,
#four {
    background-image: url('../images/blue-triangle.png');
}

#one,
#two,
#three,
#four {
    width: 100%;
    height: 528px;
    margin-top: 6%;
    display: flex;
    flex-direction: row;
    justify-content: center;
    background-position-y: bottom;
    background-repeat: repeat-x;
    box-shadow: 5px -1px 9px rgb(208, 206, 206);
}

.category-list,
.social-list {
    display: flex;
}

.footer-text,
.nav-footer {
    color: #5a4a4a;
    font-size: 0.9em;
}

.left-mobile {
    display: none;
}
```

Annexe 3 extrait du code : media queries challenge révision-css-blog - style.css

```
/*----- @Media Queries -----*/  
  
@media only screen and (max-width: 480px) {  
  
    #container {  
        width: 480px;  
    }  
    header {  
        display: none;  
    }  
    .navbar {  
        display: block;  
    }  
    .navbar-item {  
        margin: 2%;  
        text-align: center;  
        line-height: 30px;  
        list-style-type: none;  
        border-radius: 3px;  
        background-color: #F3F3F2;  
    }  
    .navbar-item:hover {  
        color: #F3F3F2;  
        background-color: #5a4a4a;  
    }  
    .navbar-link {  
        color: #5a4a4a;  
        text-align: center;  
    }  
    .navbar-link:hover {  
        color: #F3F3F2;  
    }  
    .active-r {  
        color: #F3F3F2;  
        background-color: #5a4a4a;  
    }  
    #logo,  
    .left,  
    .right-content,  
    .category-social {  
        display: none;  
    }  
}
```

Annexe 4 extrait du code : projet MovieDB MovieController.php - Back-end

```
/*
 * @Route("/admin/movie/new", name="admin_movie_add")
 */
public function add(Request $request): Response
{
    $movie = new Movie();
    // je crée un objet form type
    $form = $this->createForm(MovieType::class, $movie);
    // cette méthode va vérifier si un formulaire html a été soumis en post
    // et si ce formulaire concerne l'entité Genre
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid())
    {
        // ici tout est ok, les champs sont valides et on peut continuer

        $movie = $form->getData();

        $entityManager = $this->getDoctrine()->getManager();

        $entityManager->persist($movie);
        $entityManager->flush();

        $this->addFlash('success', 'Movie ` ' . $movie->getTitle() . ' ` a bien été ajouté !');
        // on enregistre en bdd par exemple
        // puis on redirige
        return $this->redirectToRoute('admin_movie');
    }

    return $this->render('back/movie/add.html.twig', [
        'form' => $form->createView(),
    ]);
}

/**
 * @Route("/admin/movie/delete/{id}", name="admin_movie_delete", methods="GET")
 */
public function delete(Movie $movie, EntityManagerInterface $entityManagerInterface): Response
{
    $entityManagerInterface->remove($movie);

    $entityManagerInterface->flush();
    $this->addFlash('success', 'Movie ` ' . $movie->getTitle() . ' ` a bien été mis supprimé !');

    return $this->redirectToRoute('admin_movie');
}
```