

# Salut Les Zikettes !

## Dossier de projet

---

Projet réalisé dans le cadre de la présentation au Titre Professionnel Développeur Web et Web Mobile - Présenté et soutenu par Manon Le Bars



# Sommaire

A/ Introduction à Salut les Zikettes!	4
B/ Compétence du référentiel	4
B1/ Activité type 1 - Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité	4
CP1 - Maquetter une application	4
CP2 - Réaliser une interface utilisateur web statique et adaptable	4
CP3- Développer une interface utilisateur web dynamique	9
B2/ Activité type 2 - Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité	11
CP5- Créer une base de données	11
CP6- Développer les composants d'accès aux données	11
CP7- Développer la partie back-end d'une application web ou web mobile	15
C/ Résumé du projet	20
Présentation	20
Problématique	20
Public visé	21
D/ Cahier des charges	21
D1/ Fonctionnement du site	21
Rôles d'utilisateurs	21
User stories	21
V1.	21
V2.	22
Arborescence du site	23
D2/ MVP et versions du site	24
MVP	24
Autres versions du site	24
D3/ Aspects visuels	24

Wireframes	24
Charte graphique	31
Pour la partie front office du site	31
Pour la partie back office du site	31
D4/ Spécifications techniques	32
Technologies utilisées	32
MCD	33
Dictionnaire de données	34
E/ Organisation de l'équipe	38
Présentation de l'équipe	38
Répartition des rôles	38
Organisation de travail	38
Outils utilisés	39
Programme des sprints	39
F/ Réalisations personnelles	40
Aspects visuel et fonctionnels :	40
Intégrations :	41
Les formulaires :	44
G/ Jeu d'essai	57
H/ Veilles	72
Description de la veille	72
Veille significative	73
Situation de recherche	74
Sa traduction en français	76
Conclusion	77

## A/ Introduction à Salut les Zikettes!

Ce projet a été réalisé au cours de la formation “développement web et web mobile” à l’école O’Clock. Cette formation s’est déroulée de janvier 2021 à juillet 2021, représentant 700 heures intensives complètement en téléprésentiel. Celle-ci comprend 3 mois de socle durant lesquels sont enseignés HTML5, CSS3, JavaScript, PHP et MySQL. Suivi ensuite d’un mois de spécialisation sur le framework Symfony et d’un mois de projet en condition professionnelle.

J’ai également eu l’opportunité d’obtenir le certificat Opquast (Maîtrise de la qualité en projet web) à la fin de cette formation.

## B/ Compétence du référentiel

**B1/ Activité type 1 - Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité**

### CP1 - Maquetter une application

Pour réaliser cette application, il a d’abord été nécessaire que la product owner organise une réunion avec l’association afin de définir leur besoin et les guider sur les fonctionnalités qu’il était possible de mettre en place. À la suite de cette réunion il a été possible de créer les users stories (description simple d’un besoin exprimé par un utilisateur) (voir page 21), pour ensuite créer les wireframes (ou maquette fonctionnelle qui permet de définir les zones et fonctionnalités de l’application sans considération de style) (voir page 24). Nous avons utilisé le site <https://whimsical.com/> pour les concevoir. Ces wireframes ont été présentés à l’association et validés avant de pouvoir commencer à faire les intégrations. A ce stade il a également été définis l’ébauche d’une charte graphique qui s’est affinée par la suite.

### CP2 - Réaliser une interface utilisateur web statique et adaptable

Le site a été créé en mobile first (désigne le fait de penser et concevoir le site pour un terminal mobile) et pensé pour être en responsive design (désigne la faculté d’un site web à s’adapter aux différents terminaux de lecture tel que mobile, tablette et desktop). Nous avons utilisé le framework CSS Bootstrap afin de gagner du temps sur la mise en forme des différents éléments et la mise en place du responsive design.

Bootstrap s’utilise grâce à des classes préformatés à ajouter directement dans notre HTML et propose un système de grille et de breakpoints préformatés permettant de rendre rapidement le site responsive. L’avantage est d’avoir de ce fait un style

standard que l'on peut ensuite personnaliser si besoin. L'inconvénient est que cela induit de mélanger la structure du code ainsi que le style dans le même fichier et peut limiter la créativité en matière de style.

En plus de Bootstrap nous avons utilisé un fichier CSS pour apporter les couleurs et les fonts demandés par l'association.

Pour le Back Office du site nous avons utilisé un thème Bootstrap afin de différencier le front du back et de gagner du temps sur les choix de couleurs et des fonts.

Voici la page d'accueil du site en version mobile :

The screenshot shows a mobile view of the website. At the top, there's a teal header with the text "SALUT LES ZIKETTES!" and a three-line menu icon. Below the header is a photo of two women standing in front of a graffiti wall. The woman on the left wears glasses and a white t-shirt with "HYS" and "MOTRIVALLY". The woman on the right wears a blue t-shirt with "HYS" and "VERI PUNK". The main title "C'EST QUOI SALUT LES ZIKETTES?" is displayed in large, bold, light blue letters. Below the title is a video thumbnail showing two women, with the caption "Salut les Zikettes ! c'est quoi ?" and the handle "@SALUTLES\_ZIKETTES". A large block of text at the bottom provides information about the group, mentioning bimonthly workshops for women, trans women, and non-binary individuals who have never or rarely played music, focusing on fun and group interaction.

C'est quoi au juste, "Salut les Zikettes !" ?

Il s'agit d'ateliers bimestriels de musique et d'empowerment destinés aux femmes, aux femmes trans et personnes non-binaires qui n'ont jamais ou très peu fait de musique, et souhaiteraient en faire. Dans ces ateliers, on appréhende plusieurs instruments, on joue en groupe, on s'écoute, tout ça dans la bienveillance et le fun ! Les ateliers - qui se déroulent à Pantin, et parfois ailleurs en France - sont entièrement gratuits et sont animés par Julie et Victoria, respectivement bassiste et guitariste, qui jouent depuis pas mal d'années dans divers projets punk, indie-pop, post-punk, hardcore... Tout le reste (ou presque) est dit dans la vidéo !

Avec un menu burger déroulable :



Et en version desktop :



On peut voir qu'en version mobile, le menu de navigation est un menu burger déroulable, la vidéo se place au-dessus du texte. En version desktop le menu devient un menu horizontal, la vidéo et le texte sont côté à côté. Le site est responsive !

Ici on utilise Twig et non pas HTML pour la structure du code car il s'agit d'un projet en Symfony dont c'est la technologie. Cela a l'avantage par exemple de mettre directement des conditions dans le Twig pour gérer des affichages.

Dans le code ci-dessous on peut voir les classes utilisées pour rendre la vidéo et le texte responsive. Tout d'abord ils sont enveloppés dans une div qui a pour classe "row". Cela signifie que ces deux éléments se placent sur la même grille. On peut voir sur le texte "<p>" les classes suivantes : col-sm12 col-lg-6.

Cela se traduit comme ceci :

- "col" signifie que cet élément se placera en colonne dans la grille définie plus haut.
- "sm" ou "lg" représentent les breakpoints,
  - sm correspond au breakpoint mobile,
  - lg aux tablettes
  - et xl au desktop.
- Le chiffre correspond à la place que prendra l'élément, la grille Bootstrap est divisée par 12. Ainsi on peut attribuer la taille d'un élément entre 0 et 12.

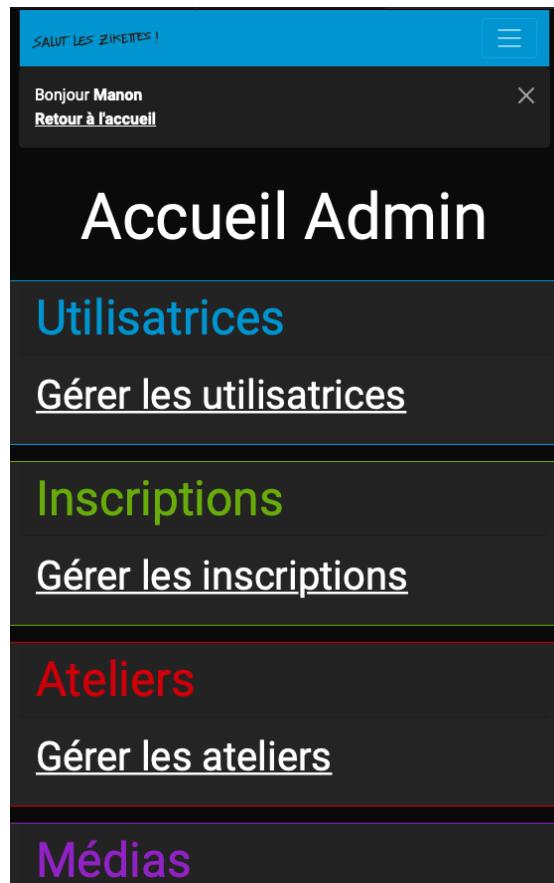
Dans notre cas, col-sm12 veut dire que l'élément se place en colonne au breakpoint mobile sur une largeur de 12 dividendes c'est-à-dire l'ensemble de l'espace disponible. Et col-lg-6 veut dire que l'élément se place en colonne au breakpoint tablette sur une largeur de 6 dividendes c'est à dire la moitié de l'espace disponible. Étant donné que l'élément "<iframe>", faisant partie de la même grille, possède les mêmes classes, sur un breakpoint mobile les deux éléments occupent l'ensemble de l'espace disponible de la grille et se positionne donc l'un au-dessus de l'autre, sur un breakpoint tablette ils se positionnent côté à côté.

Ici on utilise seulement les breakpoint sm et lg car nous avons pensé le site pour que le style soit le même sur une tablette et un desktop.

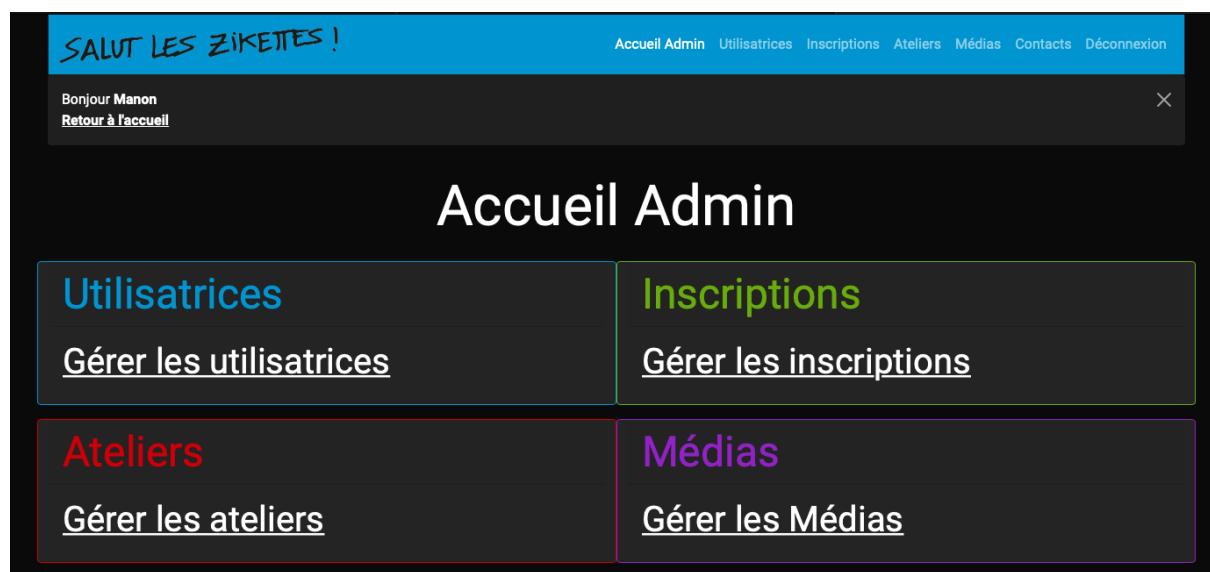
```
1  {% block body %} 
2    <h1 class="p-5 text-center">C'est quoi Salut les Zikettes?</h1>
3    <div class="row">
4      <iframe class="col-sm-12 col-lg-6 pb-3"
5        src="https://www.youtube.com/embed/e7hBXtnLZW4"
6        title="YouTube video player" frameborder="0"
7        allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen>
8      </iframe>
9      <p class="col-sm-12 col-lg-6">
10        C'est quoi au juste, "Salut les Zikettes !" ? <br>
```

Voici la page d'accueil de la page admin, le même principe est utilisé pour l'affichage des différentes cartes.

En version mobile :



Et en version desktop :



## CP3- Développer une interface utilisateur web dynamique

L'utilisation de Bootstrap permet aussi d'utiliser du JavaScript via des classes préformatées, rendant ainsi notre site dynamique.

Ici cela a été utile notamment pour la mise en place d'un carrousel qui défile seul mais réagit également aux actions de l'utilisateur grâce au flèches "suivant" à droite de l'image et "précédent" à gauche. L'état du défilement s'affiche également grâce aux barres de progressions situées en bas de l'image.



Après avoir cliqué sur la flèche de droite, l'image suivante apparaît et la barre de progression se met à jour :

**Les ateliers de musique**



48h d'immersion pour découvrir la musique en groupe, toucher à tout, déconstruire ses peurs, et faire du son !

[S'inscrire](#)

Voici le code concerné :

```

1 <div id="carouselExampleIndicators" class="carousel slide" data-bs-ride="carousel">
2   <div class="carousel-indicators">
3     <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
4     <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="1" aria-label="Slide 2"></button>
5     <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="2" aria-label="Slide 3"></button>
6     <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="3" aria-label="Slide 4"></button>
7     <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="4" aria-label="Slide 5"></button>
8   </div>
9   <div class="carousel-inner">
10    <div class="carousel-item active">
11      
12    </div>
13    <div class="carousel-item ">
14      
15    </div>
16    <div class="carousel-item ">
17      
18    </div>
19    <div class="carousel-item">
20      
21    </div>
22    <div class="carousel-item">
23      
24    </div>
25  </div>
26  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide="prev">
27    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
28    <span class="visually-hidden">Previous</span>
29  </button>
30  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide="next">
31    <span class="carousel-control-next-icon" aria-hidden="true"></span>
32    <span class="visually-hidden">Next</span>
33  </button>
34</div>
```

## B2/ Activité type 2 - Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

### CP5- Créer une base de données

Pour cette CP, nous avons conçu le MCD (voir page 33), puis le dictionnaire des données (voir page 34).

La base de données a été créée grâce à Doctrine. Dans le terminal de l'éditeur de code directement, il est possible de faire la commande suivante : `bin/console doctrine:database:create`. Une fois la base de données créée, il est possible de la même manière de créer les entités avec la commande suivante : `bin/console make:entity`. Une fois l'entité créée, doctrine permet de définir une par une les différentes entrées de l'entité ainsi que leur type et spécificités. De plus on peut définir les types de relations entre les entités. Une fois fait, on crée une migration afin d'enregistrer les changements apportés à une entité grâce à la commande : `bin/console make:migration`, cela crée un fichier que l'on pourra pusher sur Github afin que nos coéquipiers puissent apporter les mêmes changements à leur base de données. Enfin on joue la migration en base de donnée pour que les changements soient effectifs grâce à la commande : `bin/console doctrine:migrations:migrate`.

### CP6- Développer les composants d'accès aux données

Doctrine, qui est l'ORM (interface entre la BDD et l'application) de Symfony intégré par défaut, nous a permis de créer notre base de données, nos entités ainsi que leurs relations. Il va nous permettre également de se connecter à la base de données via PDO et de récupérer les données pour les rendre disponibles dans l'application. L'un des avantages majeurs est qu'il va générer automatiquement le code SQL pour faire les requêtes, ainsi si on déplace le projet sur un autre serveur nous n'aurons pas à nous soucier d'une éventuelle différence de langages.

Pour la connexion à la BDD, Doctrine possède une méthode statique `connection()`. Cette méthode prend en paramètre un DSN, par exemple : `DATABASE_URL="mysql://root:root@localhost:8889/zikettes?serverVersion=6.3"`

À l'étape précédente, lorsque nous avons créé les entités avec Doctrine, cela nous a automatiquement généré le fichier dans le dossier Entity, ainsi que la classe, les getters et les setters. Voici par exemple le fichier de l'entité `Subscribe`:

```
1 <?php
2
3 namespace App\Entity;
4
5 use Doctrine\ORM\Mapping as ORM;
6
7 /**
8 * @ORM\Entity(repositoryClass=SubscribeRepository::class)
9 */
10 class Subscribe
11 {
12     /**
13      * @ORM\Id
14      * @ORM\GeneratedValue
15      * @ORM\Column(type="integer")
16      */
17     private $id;
18
19     /**
20      * @ORM\Column(type="datetime")
21      */
22     private $dateOfBirth;
23
24     /**
25      * @ORM\Column(type="string", length=255)
26      */
27     private $city;
28
29     /**
30      * @ORM\Column(type="text", length=16777215)
31      */
32     private $play;
33
34     /**
35      * @ORM\Column(type="text", length=65535)
36      */
37     private $why;
38
39     /**
40      * @ORM\Column(type="text", length=16777215, nullable=true)
41      */
42     private $how;
43
44     /**
45      * @ORM\Column(type="datetime")
46      */
47     private $createdAt;
48
49     /**
50      * @ORM\Column(type="boolean")
51      */
52     private $isValidated;
53
54     /**
55      * @ORM\ManyToOne(targetEntity=Workshop::class, inversedBy="subscribes")
56      */
57     private $workshop;
58
59     /**
60      * @ORM\ManyToOne(targetEntity=User::class, inversedBy="subscribes")
61      */
62     private $user;
63
64     /**
65      * @ORM\Column(type="text", nullable=true)
66      */
67     private $pronoun;
68 }
```

```

1  public function __construct()
2  {
3      $this->isValidated = false;
4      $this->createdAt = new \DateTime();
5  }
6
7
8  public function getId(): ?int
9  {
10     return $this->id;
11 }
12
13 public function getDateOfBirth(): ?\DateTimeInterface
14 {
15     return $this->dateOfBirth;
16 }
17
18 public function setDateOfBirth(?\DateTimeInterface $dateOfBirth): self
19 {
20     $this->dateOfBirth = $dateOfBirth;
21
22     return $this;
23 }
24
25 public function getCity(): ?string
26 {
27     return $this->city;
28 }
29
30 public function setCity(?string $city): self
31 {
32     $this->city = $city;
33
34     return $this;
35 }

```

C'est ensuite dans le repository que sera faite la requête sql. Il existe quatre méthodes prédefinies pour récupérer les données :

```

1 /**
2  * @method Subscribe|null find($id, $lockMode = null, $lockVersion = null)
3  * @method Subscribe|null findOneBy(array $criteria, array $orderBy = null)
4  * @method Subscribe[]    findAll()
5  * @method Subscribe[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
6 */

```

Cela va nous permettre de gagner également en sécurité, car nous disposons de l'expertise de Symfony pour prévenir les failles courantes de sécurité. En utilisant Doctrine, on est automatiquement protégé d'injections SQL malveillantes sur les requêtes "classique" vu ci-dessus.

Par ailleurs, il est également possible de mettre en place des méthodes personnalisées selon nos besoins dans ce fichier. Dans ce cas, il faudra s'assurer que les valeurs provenant d'entrées utilisateurs sont bien communiquées comme des placeholders, avec la méthode setParameter. Ainsi, les requêtes personnalisées deviennent des "requêtes paramétrées". Des exemples sont disponibles en commentaires :



The screenshot shows a code editor window with a dark theme. At the top left are three circular icons: red, yellow, and green. The code itself is a snippet of PHP:

```
1  /*
2  public function findOneBySomeField($value): ?Subscribe
3  {
4      return $this->createQueryBuilder('s')
5          ->andWhere('s.exampleField = :val')
6          ->setParameter('val', $value)
7          ->getQuery()
8          ->getOneOrNullResult()
9      ;
10 }
11 */
```

Ces données sont ensuite traitées dans les controllers.

## CP7- Développer la partie back-end d'une application web ou web mobile

L'architecture de cette application est en MVC : Model, View, Controller.

Comme nous l'avons vu précédemment les données sont récupérées de la BDD via les repositories au moyen de méthodes fournies par Symfony, ou bien personnalisées. Ces données sont ensuite traitées par les controllers qui pourront alors afficher les données, en créer de nouvelles via les entrées d'utilisateurs, les modifier et les supprimer. Ces données seront ensuite affichées dans des views (ici les templates).

Pour continuer sur l'exemple des inscriptions, voici un des controllers traitant des données relatives à celles-ci. Il s'agit du Admin/SubscribeController. Grâce à ce controller, on va pouvoir depuis l'interface administration, afficher la liste des inscriptions, modifier une inscription, ou bien encore en supprimer.

Pour afficher la liste on va simplement faire appel au SubscribeRepository et à sa méthode findAll(), puis rendre ce résultat disponible sur le template admin/subscribe/browse.html.twig.

```
1  <?php
2
3  namespace App\Controller\Admin;
4
5  use App\Entity\Subscribe;
6  use App\Form\AdminSubscribeType;
7  use App\Repository\SubscribeRepository;
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
13 /**
14 * @Route("/admin/inscription", name="admin_subscribe_")
15 */
16 class SubscribeController extends AbstractController
17 {
18     /**
19      * @Route("", name="browse")
20      */
21     public function browse(SubscribeRepository $subscribeRepository): Response
22     {
23         $subscribe = new Subscribe;
24         //Check authorizations with voter
25         $this->denyAccessUnlessGranted('ADMIN_SUBSCRIBE', $subscribe);
26
27         return $this->render('admin/subscribe/browse.html.twig', [
28             'subscribes' => $subscribeRepository->findAll(),
29         ]);
30     }
}
```

Pour modifier une inscription en admin, on va cette fois récupérer les données d'un formulaire pour ensuite venir mettre à jour la base de données. On fait appel au formulaire AdminSubscribeType que voici qui va simplement permettre d'agir sur le champ "isValidated" sans modifier les autres informations de l'inscription :

```
● ● ●
1  <?php
2
3  namespace App\Form;
4
5  use App\Entity\Subscribe;
6  use Symfony\Component\Form\AbstractType;
7  use Symfony\Component\Form\Extension\Core\Type\ChoiceType;
8  use Symfony\Component\Form\FormBuilderInterface;
9  use Symfony\Component\OptionsResolver\OptionsResolver;
10
11 class AdminSubscribeType extends AbstractType
12 {
13     public function buildForm(FormBuilderInterface $builder, array $options)
14     {
15         $builder
16
17         ->add('isValidated', ChoiceType::class, [
18             'label' => 'Validation de l\'inscription',
19             'choices' => [
20                 'Valider' => 1,
21                 'Refuser' => 0],
22             'multiple' => false,
23             'expanded' => true,
24         ]);
25     }
26
27     public function configureOptions(OptionsResolver $resolver)
28     {
29         $resolver->setDefaults([
30             'data_class' => Subscribe::class,
31         ]);
32     }
33 }
```

Dans le controller on vérifie que le formulaire a été rempli correctement pour envoyer la nouvelle donnée en BDD, puis on définit un flash message et on redirige vers une autre page.

Enfin on rend disponible dans le template sélectionné, en l'occurrence 'admin/subscribe/edit.html.twig', les informations de l'inscription sélectionnée en fonction de son id, ainsi que le formulaire ciblé plus haut.



```
1  /**
2  * @Route("/modifier/{id}", name="edit", requirements={"id"="\d+"})
3  */
4  public function edit(Subscribe $subscribe, Request $request)
5  {
6      //Check authorizations
7      $this->denyAccessUnlessGranted('ADMIN_SUBSCRIBE', $subscribe);
8
9      $form = $this->createForm(AdminSubscribeType::class, $subscribe);
10     $form->handleRequest($request);
11
12     if ($form->isSubmitted() && $form->isValid()) {
13
14         $this->getDoctrine()->getManager()->flush();
15         //Display success message
16         $this->addFlash(
17             'success',
18             'L\'inscription a bien été modifiée'
19         );
20         return $this->redirectToRoute('admin_subscribe_browse');
21     }
22
23     return $this->render('admin/subscribe/edit.html.twig', [
24         'subscribe'=> $subscribe,
25         'form' => $form->createView()
26     ]);
27 }
```

Les views sont organisées de la façon suivante :

Nous avons élaboré une page de base pour le front office et une autre pour le back office. Ce système nous permet d'appliquer deux styles différents ce qui permet de maintenir une cohérence entre toutes les pages partageant la même base.

Twig nous permet de définir des blocs, ainsi sur la page de base l'architecture des pages est définie, avec un header et un footer qui seront de ce fait communs à toutes les pages héritant de cette base. Dans la partie main de la page un bloc body est défini et c'est ce bloc qui sera personnalisé ensuite pour chaque page.

Pour continuer sur l'exemple précédent, voici la page admin/subscribe/edi.html.twig. Celle-ci hérite de la base admin/base.html.twig comme nous pouvons le voir en début de fichier. Pour chaque page il est défini un titre afin de facilement reconnaître la page courante dans l'onglet de navigation. Ici un thème Bootstrap pour les formulaires est appelé afin que notre formulaire Symfony bénéficie de la mise en forme Bootstrap.

Puis le bloc body est défini, avec pour cette page les informations relatives à l'atelier sélectionné, puis le formulaire d'édition d'inscriptions des admin.

```
1  {% extends 'admin/base.html.twig' %} 
2
3  {% block title %}Valider une Inscription{% endblock %}
4  {% form_theme form 'bootstrap_5_layout.html.twig' %}
5  {% block body %}
6      <h1 class="p-3">Valider une Inscription à un atelier </h1>
7
8      <p class="text-success">Nom et prénom</p>
9      <p>{{subscribe.user.firstname}} {{subscribe.user.lastname}}</p>
10     <p class="text-success">Veut s'inscrire à l'atelier:</p>
11     <p>{{subscribe.workshop.title}}</p>
12     <p class="text-success">Qui a lieu à:</p>
13     <p>{{subscribe.workshop.place}}</p>
14     <p class="text-success">Date de Naissance: </p>
15     <p>{{subscribe.dateOfBirth|date('d-m-Y')}}</p>
16     <p class="text-success">Ville de résidence: </p>
17     <p>{{subscribe.city}}</p>
18     <p class="text-success">
19         Par quel prénom souhaitez-tu que l'on s'adresse à toi ?:
20     </p>
21     <p>{{subscribe.pronoun}}</p>
22     <p class="text-success">
23         As-tu déjà pratiqué un instrument (la voix étant aussi un instrument!)?
24         Si oui, depuis combien de temps et dans quel cadre?:
25     </p>
26     <p>{{subscribe.play}}</p>
27     <p class="text-success">
28         Peux-tu nous dire, en quelques mots, ta motivation et/ou tes attentes concernant l'atelier?:
29     </p>
30     <p>{{subscribe.why}}</p>
31     <p class="text-success">Comment as tu connu "Salut Les Zikettes!"</p>
32     <p>{{subscribe.how}}</p>
33
34     {{ form_start(form) }}
35     {{ form_row(form.isValidate) }}
36     <button class="btn btn-success" type="submit">Valider</button>
37     {{ form_row(form._token) }}
38     {{ form_end(form) }}
39
40  {% endblock %}
```

Symfony dispose d'outils pour se protéger contre les attaques CSRF (Cross Site Request Forgery) notamment au niveau des formulaires (CsrfToken). Il s'agit d'ajouter à la fin de chaque formulaire un token aléatoire, aussi stocké dans la session utilisateur, qui sera validé lorsque l'utilisateur soumet le formulaire. Ce token n'est ni connu ni devinable par un attaquant, il sera donc incapable de fabriquer une fausse requête.

Parmi les outils proposés par Symfony, nous avons utilisé les contraintes de validations pour valider les inputs. Si par exemple il est attendu qu'un email soit fourni par l'utilisateur dans un des champs, alors nous utilisons la contrainte de validation Symfony\Component\Validator\Constraints\Email.

Au niveau de la sécurité, nous avons également mis en place le hachage des mots de passe avant de les envoyer en base de données :



The screenshot shows a code editor window with a dark theme. At the top left are three colored circular icons: red, yellow, and green. Below them is a block of PHP code:

```
1 if ($form->isSubmitted() && $form->isValid()) {  
2     // hash the plain password  
3     $user->setPassword(  
4         $passwordEncoder->hashPassword(  
5             $user,  
6             $form->get('plainPassword')->getData()  
7         )  
8     );
```

Concernant les mots de passe nous avons également décidé de demander 8 caractères minimum, et nous indiquons une phrase de prévention sous le formulaire afin de conseiller les utilisateurs d'utiliser des combinaisons de chiffre, lettre, majuscule et minuscule et des caractères spéciaux. Ce niveau de sécurité permet d'éviter que les attaquants trouvent les mots de passe les plus évidents. Cependant nous n'avons pas mis de contraintes supplémentaires, que les 8 caractères minimum, afin de ne pas gêner les utilisateurs dans leur utilisation de l'application.

## C/ Résumé du projet

### Présentation

Ce projet a été réalisé pour l'association "Salut les Zikettes !" et en collaboration avec celle-ci. C'est une association régie par la loi du 1er juillet 1901 et le décret du 16 août 1901.

Elle a pour objet :

- D'une part, de contribuer au développement de l'expression artistique et culturelle, principalement dans le domaine de la culture.
- D'autre part, l'empowerment et l'émancipation des femmes, des personnes non-binaires et des personnes se définissant comme femme par le biais de la culture afin d'encourager leur visibilité, leur autonomisation et leur accès à de nouvelles opportunités.
- Enfin, par sa stratégie, de sensibiliser et promouvoir l'égalité entre les femmes, les hommes et les personnes non-binaires.

Il s'agit d'ateliers bimestriels de musique et d'empowerment destinés aux femmes (cis et trans) et personnes non-binaires. Au cours de ces ateliers, on appréhende plusieurs instruments, on joue en groupe, on s'écoute, tout ça dans la bienveillance et la bonne humeur ! Les ateliers - qui se déroulent à Pantin - sont entièrement gratuits et sont animés par Julie et Victoria, respectivement bassiste et guitariste, qui jouent depuis plusieurs années dans divers projets punk, indie-pop, post-punk, hardcore...

### Problématique

Leur site actuel a été créé avec Wix, ce qui induit une limitation des choix de personnalisation. L'idée est de développer un site "tout-en-un", s'adaptant aux besoins de l'association. Cela passe par une refonte totale de leur site existant afin de leur proposer l'ajout de fonctionnalités pertinentes : dans un premier temps permettre les inscriptions à des ateliers directement sur le site (actuellement géré par mail), la gestion des membres (actuellement géré sur Facebook), ensuite mettre en place un forum pour favoriser l'échange des membres de l'association (actuellement sur Facebook). Dans un second temps développer une partie blog pour proposer des articles et podcast (actuellement partagé via leur blog et les réseaux sociaux), et enfin mettre en place une plateforme de e-commerce. Cela nous permet également de leur proposer un site responsive développé en mobile first.

## Public visé

Le site est à destination des membres de l'association et de musiciennes (ou futures musiciennes) de tout niveau souhaitant s'inscrire à un atelier. La moyenne d'âge du public visé est de 18 à 50 ans.

## D/ Cahier des charges

### D1/ Fonctionnement du site

#### Rôles d'utilisateurs

Un rôle hérite de tous les droits d'un rôle inférieur.

Il existe 5 niveaux de rôles:

- Anonyme : visiteur non-connecté.
- Utilisateur : visiteur connecté, peut accéder aux inscriptions d'atelier et à son compte
- Utilisateur-membre : visiteur connecté, membre de l'association, peut se connecter à la page forum et accéder à son compte
- Modérateur : visiteur connecté, membre de l'association, peut se connecter à la page forum, accéder à son compte et a les droits de modération sur le forum
- Administrateur : visiteur connecté, membre de l'association, peut accéder à son compte, se connecter à la page forum, a les droits de modération sur le forum, peut accéder à la page admin et gérer les ateliers, les inscriptions, les membres et le contenu de la page média.

#### User stories

V1.

- En tant qu'anonyme je veux accéder à la page d'accueil afin de découvrir l'association.
- En tant qu'anonyme je veux accéder à la page médias afin de découvrir les groupes de musique issus des ateliers.
- En tant qu'anonyme je veux accéder à la page contact afin de contacter l'association.
- En tant qu'anonyme je veux accéder à la page mentions légales afin de pouvoir les consulter
- En tant qu'anonyme je veux accéder à la page de consultation des ateliers afin de consulter les ateliers ouverts aux inscriptions et d'y trouver un lien vers la

page login afin d'être authentifié pour accéder à la page d'inscription aux ateliers.

- En tant qu'anonyme je veux accéder à la page login afin de m'identifier ou de me créer un compte.
- En tant qu'utilisateur je veux accéder à la page d'inscription à un atelier afin de m'inscrire à un atelier.
- En tant qu'utilisateur je veux accéder à la page mon compte à un atelier afin de consulter et modifier les informations liées à mon compte et voir si ma demande d'inscription à un atelier est acceptée.
- En tant qu'admin je veux accéder à la page admin afin d'accéder aux fonctionnalités admin.
- En tant qu'admin je veux créer un nouvel atelier afin de l'afficher sur le site.
- En tant qu'admin je veux accéder aux inscriptions d'un atelier afin de les consulter.
- En tant qu'admin je veux pouvoir valider une inscription à un atelier afin de choisir les participantes.
- En tant qu'admin je veux pouvoir supprimer des inscriptions afin de faire le tri.
- En tant qu'admin je veux accéder à la liste des users et trier en fonction des rôles afin de les consulter.
- En tant qu'admin je veux modifier des rôles à des utilisateurs afin de passer un user non-membre à membre par exemple.
- En tant qu'admin je veux supprimer un user afin de faire le tri ou "bannir" un membre indésirable.
- En tant qu'admin je veux accéder à la liste des éléments de la page média afin de les consulter.
- En tant qu'admin je veux accéder créer un élément de la page média afin de l'afficher sur le site.
- En tant qu'admin je veux modifier un élément de la page média afin de pouvoir faire des modifications en cas d'évolution du groupe ou d'erreur lors de la saisie.
- En tant qu'admin je veux supprimer un élément de la page média afin de pouvoir faire des modifications en cas d'évolution du groupe, ou bien faire le tri.

V2.

- En tant que membre je veux accéder au forum afin de consulter la liste des posts.
- En tant que membre je veux accéder à la page d'un post afin de consulter un post.
- En tant que membre je veux pouvoir poster dans le forum afin d'ajouter un post.
- En tant que membre je veux pouvoir répondre à un post dans le forum afin de commenter le post.
- En tant que membre je veux pouvoir modifier un commentaire dont je suis

l'auteur afin de modérer mes propos par exemple.

- En tant que membre je veux pouvoir supprimer un commentaire dont je suis l'auteur afin de modérer mes propos par exemple.
- En tant que membre je veux archiver un post dont je suis l'auteur afin de clore le post.
- En tant que membre je veux recevoir un e-mail quand on répond à un de mes posts afin d'être informé qu'il y a eu un commentaire.
- En tant que modérateur je veux archiver un post dont je ne suis pas l'auteur afin de clore le post.
- En tant que modérateur je veux supprimer le commentaire de quelqu'un d'autre afin de modérer le forum.
- En tant que modérateur je veux accéder à la liste des tags afin de consulter les tags.
- En tant que modérateur je veux ajouter un tag afin de pouvoir les afficher dans le forum.
- En tant que modérateur je veux modifier un tag afin de rattraper un erreur de frappe.
- En tant que modérateur je veux supprimer un tag afin de faire le tri.

## Arborescence du site

- Accueil
- Ateliers (consultation)
- Média
- Nous contacter
- Connexion
- Crédit de compte
  - Ateliers (inscription)
  - Forum
  - Mon compte
  - Admin
    - Gestion atelier
    - Gestion inscriptions
    - Gestion membre
    - Gestion média
  - Déconnexion
- Mentions légales
- 404
- 403

## D2/ MVP et versions du site

### MVP

- Accueil : Présentation de l'association et de ses activités.
- Inscription aux ateliers : Présentation des ateliers, il faut s'inscrire à la plateforme pour se créer un compte avant de pouvoir s'inscrire à un atelier.
- Média : Présentation des groupes issus des ateliers
- Connexion : Formulaire de connexion
- Création de compte : Formulaire de création de compte
- Page mon compte
- Page d'inscription aux ateliers (disponible seulement en étant connecté)
- Page administrateur : Gestion des inscriptions aux ateliers, des membres, des inscriptions et du contenu de la page média.

### Autres versions du site

Nous avons eu le temps lors de notre mois de projet de commencer à développer la V2.

V2 : Forum : Poster un message, répondre à un message.

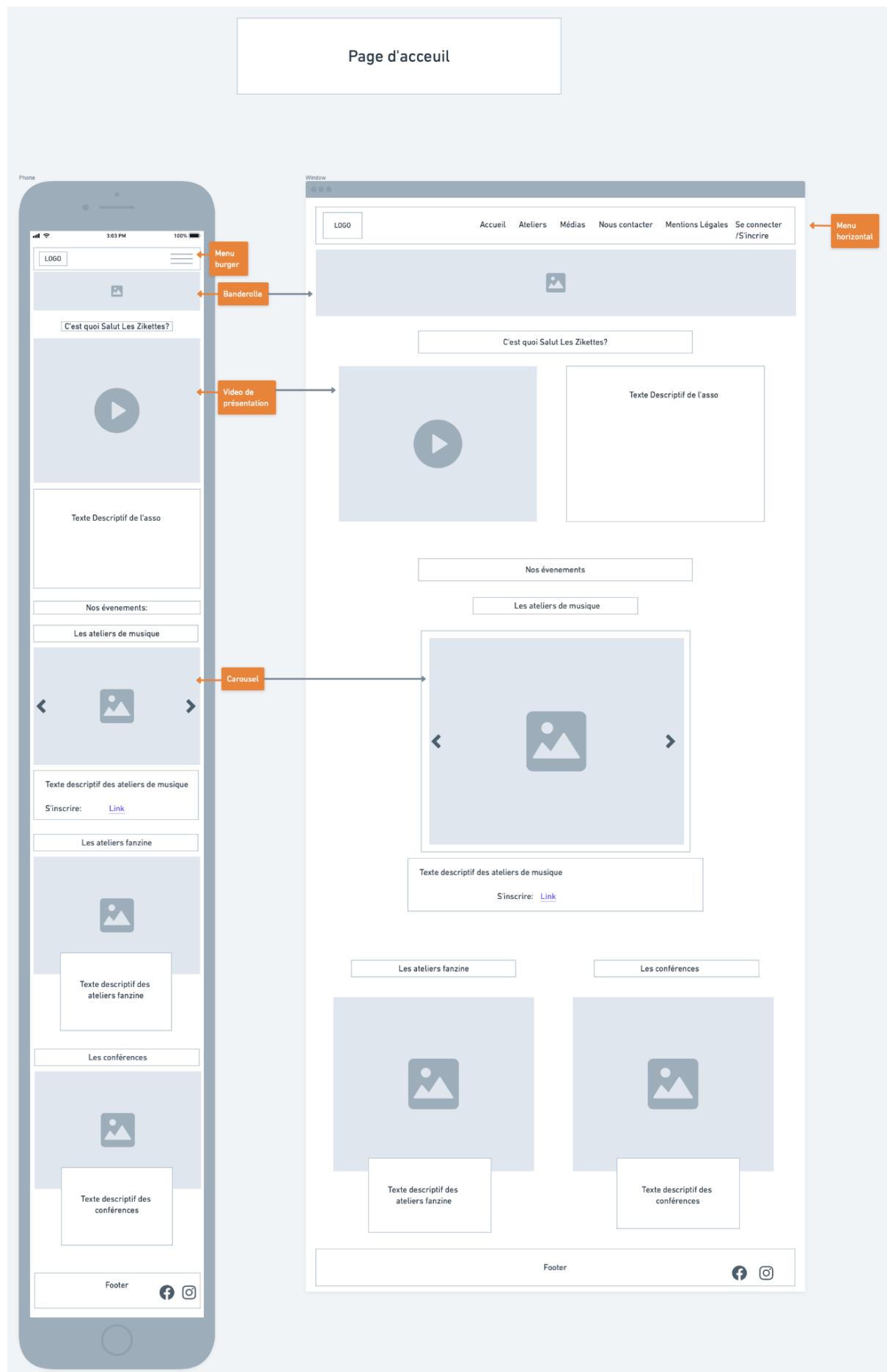
V3 : Blog : Article, tuto et Podcast

V4 : E-commerce : Merch

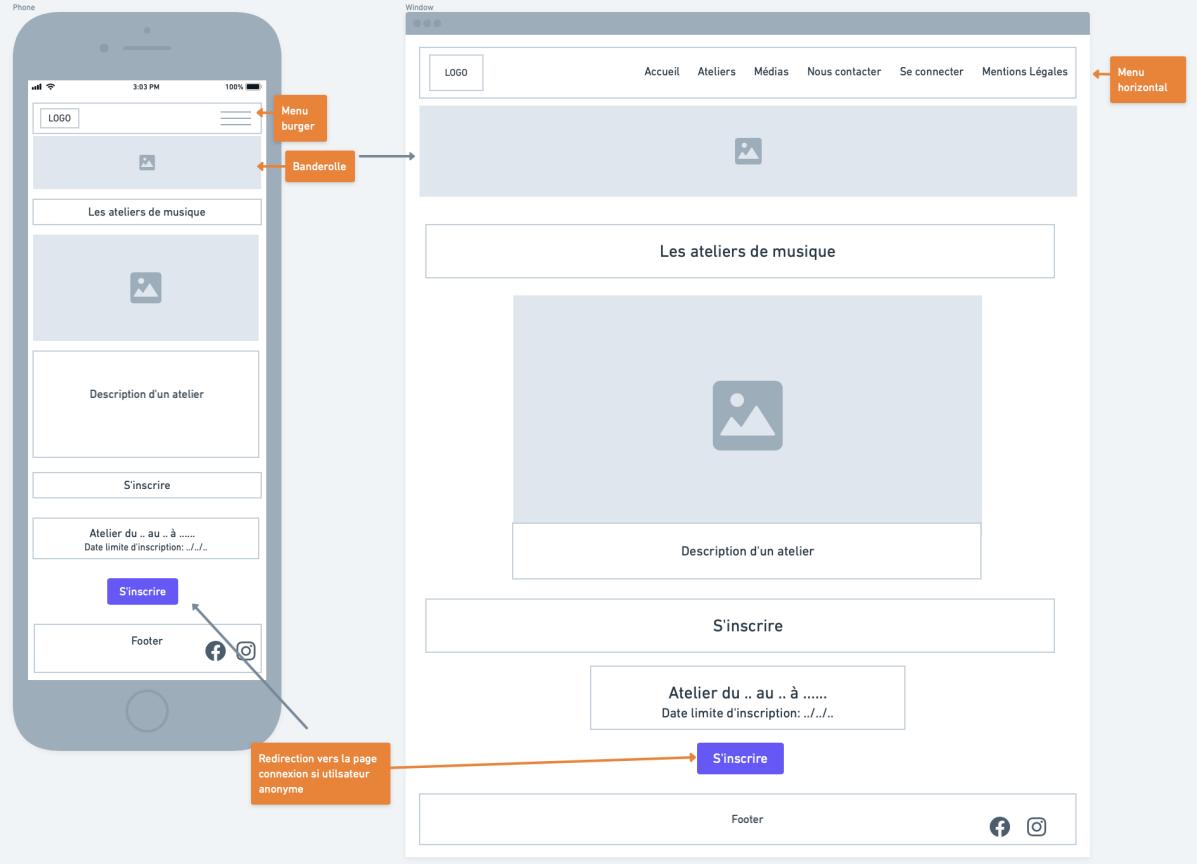
## D3/ Aspects visuels

### Wireframes

Nous avons réalisé les Wireframes (ou maquette fonctionnelle) du site web afin de définir l'architecture des pages, de s'accorder sur la place des fonctionnalités et ainsi aider à la future intégration.



## Page Ateliers Consultation



## Page Ateliers Inscriptions

The diagram illustrates the design of the 'Ateliers Inscriptions' page across two devices: a smartphone (left) and a desktop window (right). The desktop version is annotated with several UI elements:

- Menu burger**: Located at the top left of the header.
- Banderole**: A horizontal banner below the header.
- Logo**: Located in the top left corner of the header.
- Header menu**: Includes links for Accueil, Ateliers, Médias, Contact, Mon compte, and Se déconnecter.
- Image placeholder**: A large image placeholder area with a camera icon.
- Title**: 'Les ateliers de musique'.
- Description**: 'Description d'un atelier'.
- Call-to-action**: 'S'inscrire' button.
- Text input fields**: 'Atelier du .. au .. à ..... Date limite d'inscription: ..../../'.
- Date input field**: 'Date de naissance'.
- Text input field**: 'Ville de résidence'.
- Text input field**: 'Joue tu d'un instrument?'.
- Text input field**: 'Pourquoi participer à un atelier?'.
- Text input field**: 'Comment as tu entendu parler des Zikettes?'.
- Call-to-action**: 'S'inscrire' button.
- Footer**: Includes social media icons for Facebook and Instagram.

The mobile phone design is shown on the left, mirroring the desktop layout but adapted for a smaller screen. It includes a header with a logo and menu, a main content area with an image placeholder, and a footer with social media links.

## Page Contact

The diagram illustrates the design of a 'Page Contact' across two devices: a smartphone (Phone) and a desktop computer (Window). Both versions feature a header with a logo and a horizontal menu bar with links: Accueil, Ateliers, Médias, Nous contacter, Mentions Légales, and Se connecter.

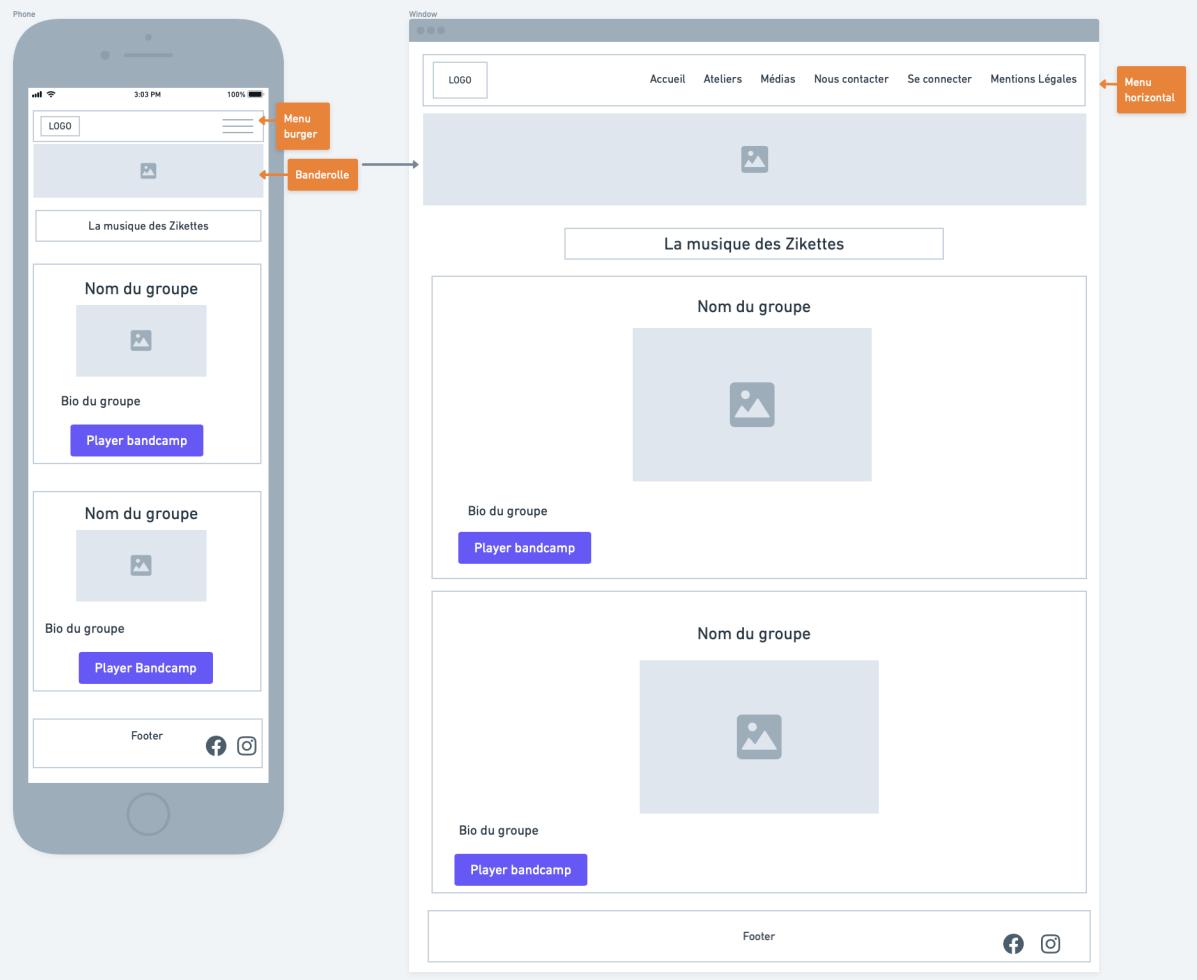
**Mobile (Phone) View:**

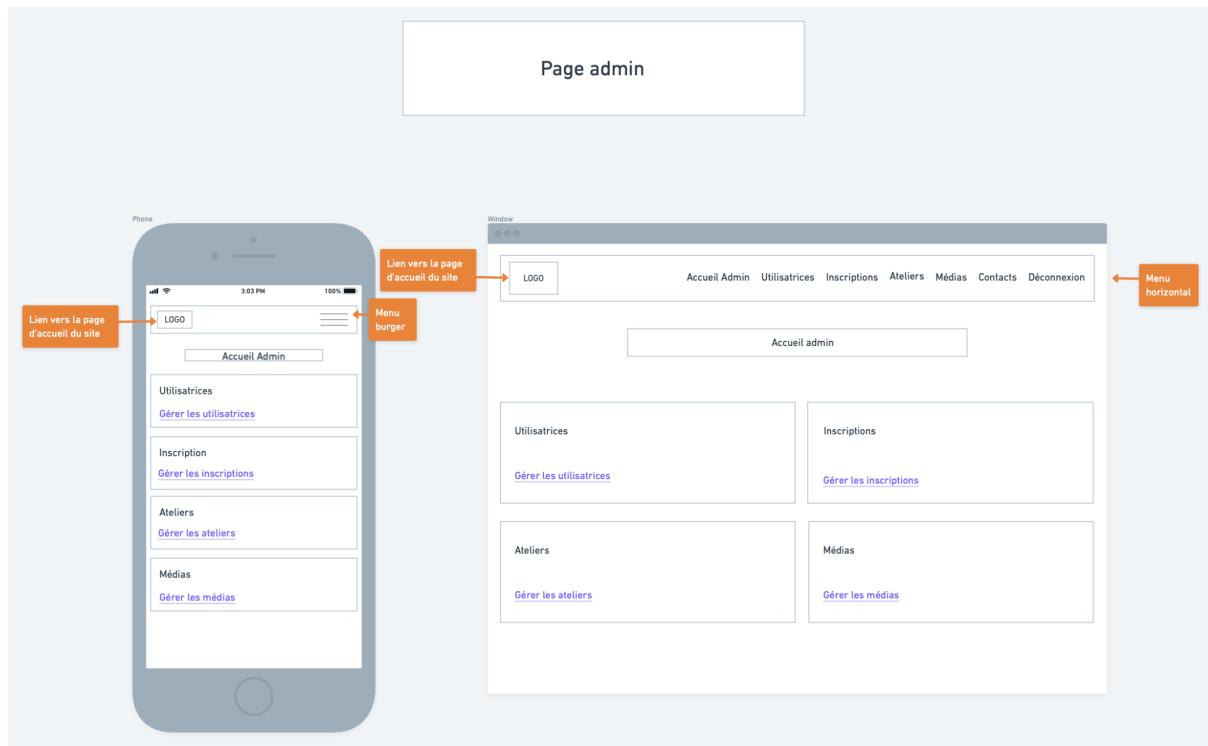
- Header:** Includes a logo and a "Menu burger" icon (indicated by an orange arrow).
- Banner:** A call-to-action banner with the text: "Tu as une question sur l'asso ? Sur les ateliers ? Un témoignage que tu voudrais partager ? Des idées de projet à développer avec nous ? Contacte-nous !".
- Form Fields:** Fields for "Nom" (Name), "E-mail", and "Objet" (Subject).
- Message Area:** A text area for "Rédigez votre message ici" (Compose your message here).
- Send Button:** A blue "Envoyer" button.
- Footer:** Includes social media icons for Facebook and Instagram.

**Desktop (Window) View:**

- Header:** Includes a logo and a "Menu horizontal" icon (indicated by an orange arrow).
- Banner:** A call-to-action banner with the same text as the mobile version.
- Form Fields:** Fields for "Nom", "E-mail", and "Objet".
- Message Area:** A text area for "Rédigez votre message ici".
- Send Button:** A blue "Envoyer" button.
- Footer:** Includes social media icons for Facebook and Instagram.

## Page Médias



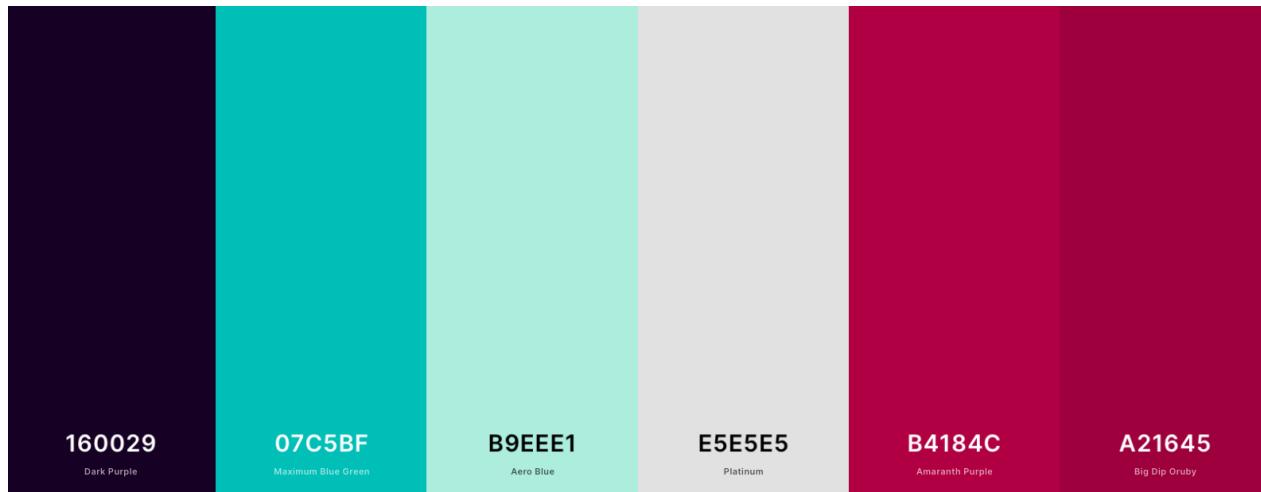


## Charte graphique

La charte graphique a été réalisée en collaboration avec l'association.

Pour la partie front office du site

Il a été défini la palette de 6 couleurs ci-dessous :



Les couleurs en code hexadécimal de la gauche vers la droite :

- #160029
- #07C5BF
- #B9EEE1
- #E5E5E5
- #B4184C
- #A21645

Celles-ci ont été définies en prenant en compte des critères d'accessibilités tel que le contraste afin de faciliter la lecture des contenus aux utilisateurs.

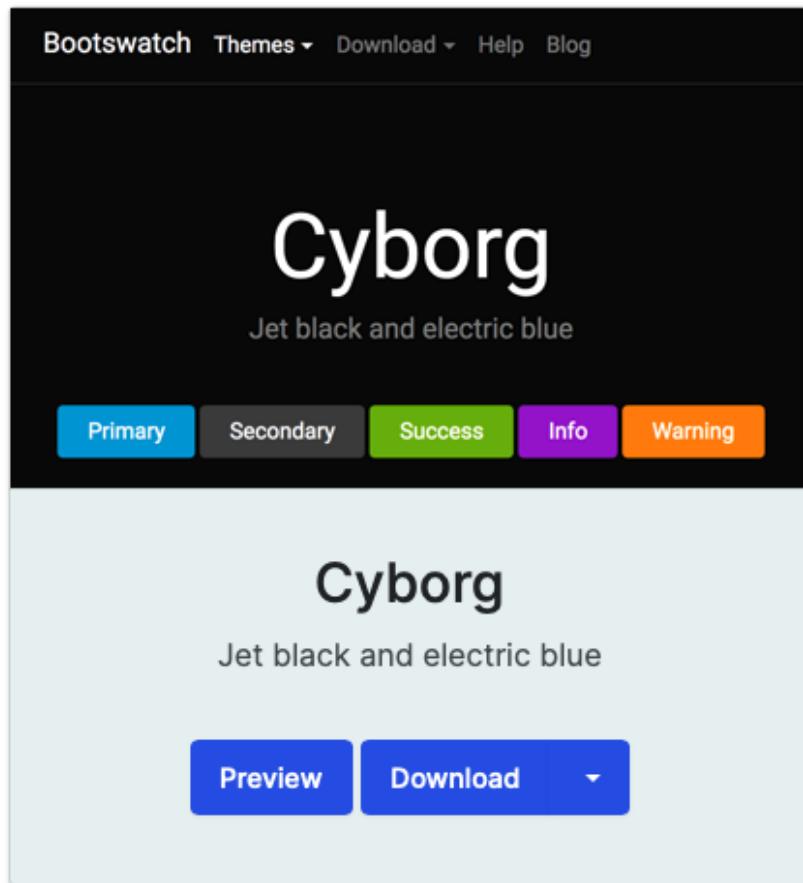
Les fonts choisis sont :

Pour les h1, la font “Inika”.

Pour le reste du site “PT Sans”.

Pour la partie back office du site

Il a été convenu d'utiliser le thème Bootstrap [Cyborg](#) afin de gagner du temps sur l'intégration notamment sur le choix des couleurs et des fonts :



## D4/ Spécifications techniques

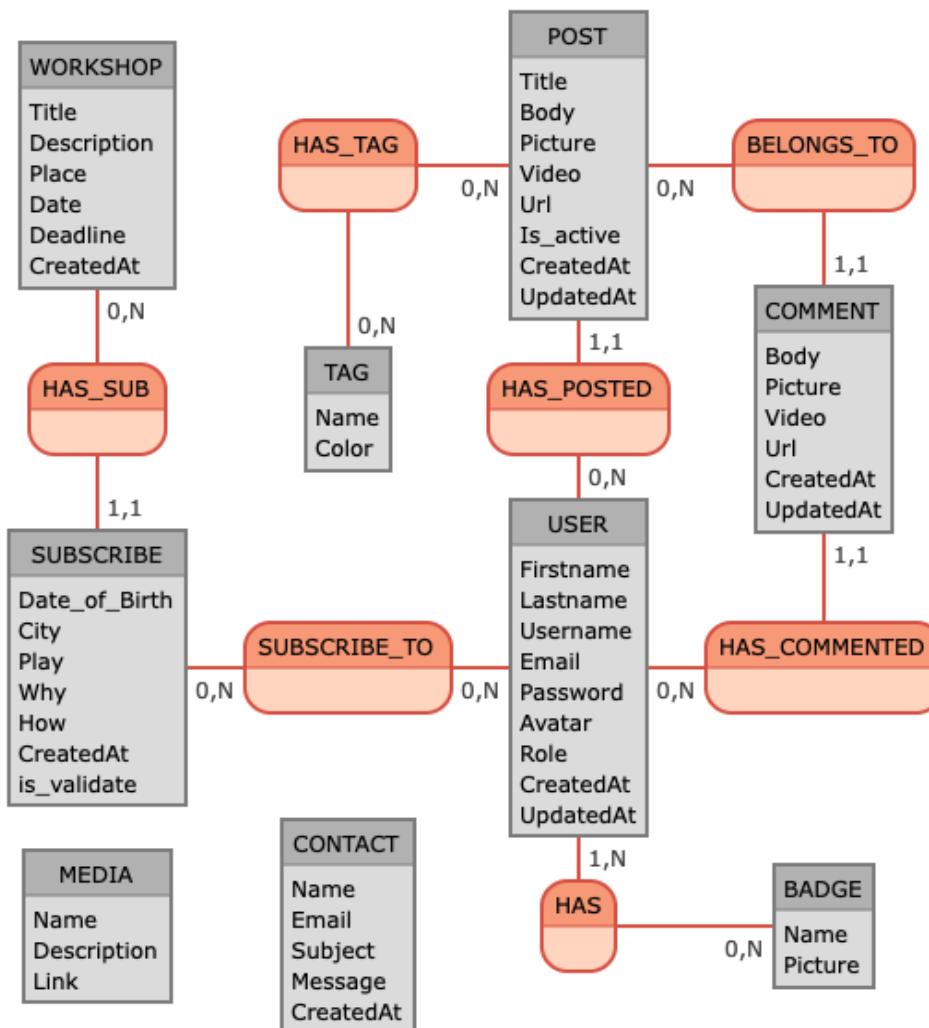
### Technologies utilisées

- **PHP/Symfony** : Symfony est un framework (boîte à outils) de PHP. Ce dernier permet de développer rapidement et simplement des sites web. La facilité de maintenance du code et la stabilité de ce framework en font un choix de qualité pour notre projet.
- **Bootstrap** : Un framework CSS est utilisé afin de faciliter la création du design général d'un site à l'aide notamment de template. Nous l'utilisons afin de gagner du temps sur la partie front-end du projet.
- **CSS** : Utilisé en complément de Bootstrap afin de personnaliser les couleurs notamment et ainsi de respecter la charte graphique.
- **Twig** : Moteur de template, affichage des vues, car utilisé par Symfony.

## MCD

Une fois mis d'accord sur les fonctionnalités, nous avons pu réfléchir à notre MCD pour le MVP et la V2. Cela a nécessité beaucoup de réflexion et de discussion au vu de la complexité des relations entre les entités.

Toutes les entités sont en relation entre elles, hormis les entités Media et Contact. Une des particularités de ce MCD, est que l'entité Subscribe est une table pivot à laquelle nous avons ajouté des informations en plus des clés étrangères User\_id et Workshop\_id. Cela permet d'alléger la table User de certaines informations qui ne sont pertinentes que dans le cadre d'une demande d'inscription, tel que la date de naissance, ou la ville de résidence. De plus, cela permet d'avoir en base de données la date de l'inscription (CreatedAt) et de savoir si celle-ci a été validée ou non par une admin (is\_validate). Par ailleurs, ce système permet de s'inscrire à plusieurs ateliers. Pour des raisons de sécurité, nous avons souhaité que seul un utilisateur ayant un compte sur le site puisse s'inscrire à un atelier. Cela diminue les risques de spam par un boot par exemple, et permet d'avoir un meilleur suivi des utilisateurs.



## Dictionnaire de données

Une fois le MCD terminé nous avons pu mettre en place le dictionnaire de données ci-dessous. Nous avons ajouté les versions afin de pouvoir nous concentrer sur la V1 (MVP) dans un premier temps.

<b>Table user</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Firstname	varchar(64)	not null	Le prénom de l'utilisateur	V1
Lastname	varchar(64)	not null	Le nom de l'utilisateur	V1
Username	varchar(64)		L'alias de l'utilisateur	V1
Email	varchar(64)	not null	l'email de l'utilisateur	V1
Password	varchar(64)	not null	Le MDP de la personne	V1
Avatar	varchar(64)		L'avatar de l'utilisateur	V1
Role	varchar(64)	not null	Le rôle de la personne	V1
Created_at	timestamp	not null, default current_timestamp	La date de création de la personne	V1
Updated_at	timestamp		La date de modification de la personne	V1

<b>Table badge</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Name	varchar(64)	not null	Le nom du badge	V2
Picture	varchar(64)	not null	L'image du badge	V2

<b>Table badge_user</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Badge_id	INT	not null	id du badge	V2
User_id	INT	not null	id du user	V2

<b>Table Post</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Title	varchar(255)	not null	Le titre de la question	V2
Body	text	not null	Le texte du post	V2
Picture	varchar (64)		L'image du post	V2
Url	varchar (255)		L'url du post	V2
Is_active	boolean	not null	Le statut du post	V2
User_id	int	not null	L'identifiant de l'auteur du post	V2
Created_at	timestamp	not null, default current_timestamp	La date de création du post	V2
Updated_at	timestamp		La date de modification du post	V2

<b>Table comment</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Body	TEXT	not null	Le texte du commentaire	V2
Picture	varchar(64)		L'image du post	V2
Url	varchar(255)		L'url du post	V2
User_id	int	not null	L'identifiant de l'auteur du post	V2
Post_id	int	not null	L'identifiant du post	V2
Created_at	timestamp	not null, default current_timestamp	La date de création du post	V2
Updated_at	timestamp		La date de modification du post	V2

<b>Table tag</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Name	varchar( 64)	not null	Le nom du tag	V2
Color	varchar( 64)		La couleur du tag	V2

<b>Table post_tag</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Post_id	int	not null	L'identifiant du post	V2
Tag_id	int	not null	L'identifiant du tag	V2

<b>Table subscribe</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Date_of_Birth	timestamp	not null	La date de naissance de l'utilisateur	V1
City	varchar(128)	not null	La ville de résidence de l'utilisateur	V1
Play	text	not null	L'utilisateur joue-t-il d'un instrument	V1
Why	text	not null	Pourquoi l'utilisateur souhaite-t-il participer à l'atelier.	V1
How	text		De quelles manières l'utilisateur a connu l'association ?	V1
Created_at	timestamp	not null, default current_timestamp	La date de demande d'inscription.	V1
User_id	int	not null	L'identifiant de l'utilisateur	V1
Workshop_id	int	not null	L'identifiant de l'atelier	V1
is_Validate	boolean	not null	L'inscription est-elle validée	V1

<b>Table Workshop</b>				
<b>Champ</b>	<b>Type</b>	<b>Spécificités</b>	<b>Description</b>	<b>Version</b>
Title	varchar(64)	not null	Le titre de l'atelier	V1
Description	varchar(255)	not null	La description de l'atelier	V1
Place	varchar(64)	not null	Le lieu de l'atelier	V1
Date	varchar(64)	not null	La date de l'atelier	V1
Deadline	timestamp	not null	La date limite d'inscription à l'atelier	V1
Created_at	timestamp	not null, default current_timestamp	La date de demande d'inscription.	V1

**Table media**

Champ	Type	Spécificités	Description	Version
Name	varchar (64)	not null	Le nom du groupe	V1
Description	varchar (255)	not null	La biographie du groupe	V1
Link	varchar (255)		Lien d'écoute	V1

**Table contact**

Champ	Type	Spécificités	Description	Version
Name	varchar(255)	not null	Nom de la personne qui envoi le message	V1
Email	varchar(255)	not null	Email de la personne qui envoi le message	V1
Subject	varchar(255)	not null	Sujet du message	V1
Message	longtext	not null	Texte du message	V1
Created_at	timestamp	not null, default current_timestamp	Date de création du message	V1

## E/ Organisation de l'équipe

### Présentation de l'équipe

Nous sommes une équipe de trois développeuses issues de la même formation (Virginie, Justine et moi-même). Nous sommes également toutes les trois spécialisées en back-end et formées sur le framework Symfony.

### Répartition des rôles

- Justine : Contribution lors de la phase de conception du projet, absente ensuite suite à des problèmes de santé.
- Virginie : Lead dev back et git master
- Manon : Product owner, Lead dev front et Scrum master.

### Organisation de travail

Nous avons utilisé la méthode de travail agile Scrum.

Cela se traduit par un découpage de notre mois de projet en 4 sprints, chaque sprint dure une semaine et chacun est rythmé par des daily Scrum (réunion courte en début de journée supervisé par le Scrum master qui sert à faire le point sur ce qui a été fait par chacune la veille et à définir les tâches à effectuer dans la journée) mais également des Scrum review (réunion un peu plus longue à chaque fin de sprint pour faire un point global sur la semaine écoulée, les difficultés rencontrés, les éventuels besoins et axes d'améliorations. Ils servent également à se mettre d'accord sur les objectifs du sprint suivant.).

Toujours en suivant le principe agile, il n'y a pas eu de tâche imposée, chacune a été libre de choisir ces tâches en prenant bien sûr en compte l'ordre de priorité définis au préalable. Nous avons choisi de travailler de 9h à 12h puis de 13h à 15h en étant en contact audio, puis de travailler deux heures (parfois plus selon son envie personnelle) en totale autonomie. Il a été également choisi de travailler séparément hormis pour les tâches les plus complexes pour lesquelles nous avons alors mis en place le pair-programming en utilisant l'option de liveshare de VS code.

## Outils utilisés

Nous avons utilisé les outils suivants :

- Slack : utilisé pour l'échange écrit en conversation de groupe et de message privé. Utile pour s'échanger des liens, s'informer dans quel salon discord se rejoindre etc...
- Discord : utilisé pour les échanges oraux, nous nous en sommes servis pour nos réunions et pour rester connecter la journée pendant "nos heures connectées" cela sert notamment à se poser des questions en cas de blocage, se prévenir en cas de push d'une branche par exemple ou bien pendant le pair programming.
- Trello : outil de gestion de projet en ligne cela permet de découper le projet en planche par exemple le "product backlog" va regrouper toutes les cartes (chaque carte représente une tâche) du projet, le "sprint backlog" va comporter les cartes à traiter durant le sprint courant, "doing" les cartes en cours de traitement et le "done" les cartes traitées. Chaque carte peut être attribuée à une ou plusieurs personnes et sont mobiles pour être assignées d'une planche à l'autre traduisant ainsi leur état d'avancement.
- Google Drive : création d'un dossier permettant de partager à toutes les personnes impliquées dans le projet les différents fichiers relatifs à celui-ci. Par exemple les journaux de bord, document de veille technique, cahier des charges etc...
- Github : outil de versionning, qui permet de travailler à plusieurs sur le même repos et ainsi bénéficier du travail du reste de l'équipe.

## Programme des sprints

- Le sprint 0 : phase de mise en place de notre organisation de travail et de conception de l'application.  
Durant cette phase nous avons définis les rôles de chacune, mis en place les outils de gestion de projet, définit l'organisation des branches et commits sur Github, rédigé notre cahier des charges, créé les wireframes, défini une charte graphique, créé le dictionnaire des routes, créé le dictionnaire de données, créé notre MVC via Mocodo.
- Le sprint 1 : phase de mise en place du projet en développement  
Durant cette phase nous avons préparé l'environnement de développement (vs code, Github, la vm et installer les composants nécessaires, Symfony), créé la base de données et les entités, créé les intégrations, créé les controllers du MVP, gérer les droits selon les rôles utilisateurs, modifier la charte graphique

en suivant des recommandations d'accessibilité, créé les formulaires du MVP, mise en place des templates du MVP.

- Le sprint 2 : Finition du MVP et mise en place de fonctionnalités de la V2.  
Durant cette phase nous avons mis en place des notifications pour les admins, mis en place des messages d'erreur et de succès pour l'utilisateur, affiné les intégrations, configuré l'envoi d'email, personnalisé les pages d'erreur 403, 404 et 500, réglé les derniers bugs du MVP, mis en place une partie de la V2 avec un forum basique.
- Le sprint 3 : Dernier réglages des fonctionnalités mise en place et mise en production du site.  
Durant cette phase nous avons mis en place plus de sécurité avec le double mot de passe, la fonctionnalité de mot de passe oublié et la vérification d'email lors de la création de compte. Nous avons également corrigé les derniers bugs, nettoyé notre code, déployé l'application sur Heroku et préparé la présentation de notre projet en vue du live Youtube.

## F/ Réalisations personnelles

### Aspects visuel et fonctionnels :

Durant le premier sprint qui correspond à la phase de conception, j'ai en tant que product owner, organisé une réunion et échangé quotidiennement avec l'association afin de définir leurs besoins et leur proposer des fonctionnalités. À la suite de ces échanges, j'ai été en mesure de produire les users stories qui ont servi de base de conception.

Dans un second temps et en tant que Lead Dev Front, j'ai produit les différents Wireframes sur le site <https://whimsical.com> et les ai validés avec l'association. Ensuite j'ai pu définir la charte graphique en prenant en compte les demandes de l'association, et de certaines recommandations d'accessibilité. Notamment choisir des couleurs qui permettent un contraste suffisant pour une bonne lisibilité. Pour choisir les couleurs j'ai utilisé le site <https://coolors.co> qui permet de créer une palette, pour tester les contrastes j'ai utilisé le site <https://juicystudio.com/services/luminositycontrastratio.php#specify>.

J'ai également suivi des recommandations de UX et UI design, pour maintenir une cohérence visuelle sur l'ensemble du site. Par exemple, avoir tous les liens de la même couleur sur l'ensemble du site pour ne pas dérouter l'utilisateur et lui permettre de naviguer plus facilement.

## Intégrations :

Durant le second sprint, qui correspond au début du développement du site, j'ai fourni toutes les intégrations. Pendant que mon binôme mettait en place le projet Symfony, j'ai créé les intégrations en HTML. J'ai pu ensuite me servir de cette base HTML pour créer les templates en Twig dans le projet, dans un premier temps en statique, et j'ai pu au fur et à mesure de la semaine les dynamiser avec les données de la BDD.

Je me suis également amusé à intégrer un player Bandcamp dans la page média. Pour cela je suis allé sur la page Bandcamp du groupe ciblé et en cliquant sur Partager/intégrer il est possible de récupérer le code HTML. J'ai ensuite fait des tests pour trouver comment dynamiser ce code via des données de la BDD.

Voici le code obtenu :

```
1 <h1 class="p-5 text-center">La musique des Zikettes</h1>
2 {% for media in medias %}
3   <div class="inset col-sm-12 col-lg-12 p-3 mb-3">
4     <h2 class="text-center pb-3">{{media.name}}</h2>
5     <div class="mx-auto">
6       <div class="row mb-3">
7         
9             <p class="col-sm-12 col-lg-6 text-break">{{media.description}}</p>
10        </div>
11        {% if media.link != null %}
12          <iframe style="border: 0; width: 100%; height: 120px;" src="{{media.link}}"
13            /size=large/bgcol=ffffff/linkcol=f171a2/tracklist=false/artwork=small/transparent=true/" seamless>
14            <a href="{{media.link}}></a></iframe>
15        {% endif %}
16      </div>
17    </div>
18  {% endfor %}
```

Et le résultat :



La présence d'un player Bandcamp était une demande de l'association, mais la manipulation pour récupérer le bon lien pour que cela fonctionne n'étant pas simple, j'ai créé un tutoriel afin qu'elles soient autonomes sur cette partie. Celui-ci est disponible via le back office du site, sur la page médias ci dessous:

A screenshot of a back office interface titled "Les médias". At the top, there's a blue header with the text "SALUT LES ZIKETTES!" and a menu icon. Below the header, a message says "Bonjour Manon" and "Retour à l'accueil". There are two green buttons at the bottom: "Ajouter un média" and "Tuto ajouter un média".

En cliquant sur le lien “Tuto ajouter un média” :

The screenshot shows a mobile application interface. At the top, a blue header bar contains the text "SALUT LES ZIKETTES !". To the right of the header is a menu icon (three horizontal lines) and a close button (an 'X'). Below the header, a dark grey navigation bar displays the text "Bonjour Manon" and a link "Retour à l'accueil". A green "Retour" button is located in the bottom-left corner of this bar. The main content area features a large white title "Tuto pour ajouter un média" centered on a black background.

Pour ajouter un lien à un media suivre la procédure ci dessous:

**Aller sur la page Bandcamp du groupe:**

The screenshot shows a Bandcamp album page for "Bellatrix Boadicea" by Mary Bell. The page has a pink header with tabs for "musique", "marchandise", and "communauté". The main content area features a large image of the album cover, which is a colorful painting of a group of people. Below the cover, there is a play button and a progress bar showing "Consent 00:00 / 01:45". The page also includes a tracklist with nine songs, a "Partager / Intégrer" button, and a "Liste de souhaits" button. On the right side, there is a sidebar with information about the band, including their website (marybelltw.com), social media links (Facebook, Instagram), recommendations, and discography.

## Les formulaires :

Durant le troisième sprint, je me suis occupé des formulaires, tant en back qu'en front. Au vu des fonctionnalités de ce site, les formulaires occupent une place très importante et ont demandé une attention particulière. Tout d'abord en admin (back office), il y a les formulaires pour créer de nouveaux ateliers, médias et utilisatrices ; mais également pour pouvoir modifier ces derniers ainsi que pour valider des inscriptions.

Du côté front office il y a les formulaires pour contacter les membres de l'association, se créer un compte sur le site, se connecter, modifier les informations de son profil et s'inscrire à un atelier.

Voici par exemple un formulaire pour créer un nouvel atelier via le back office :

The screenshot shows a mobile application interface for adding a workshop. At the top, there is a blue header bar with the text "SALUT LES ZIKETTES!" and a menu icon. Below the header, a dark navigation bar displays the text "Bonjour Manon" and a link "Retour à l'accueil". A close button ("X") is located in the top right corner of this bar. The main content area has a black background and features a large white title "Ajouter un atelier". Below the title are several input fields with labels and asterisks indicating they are required:

- Titre de l'atelier \***: An empty input field.
- Description de l'atelier \***: An empty input field.
- Ville \***: An empty input field.
- Date de début et de fin de l'atelier \***: An empty input field.
- Date limite d'inscription à l'atelier \***: A date picker showing "Jan 1 2016".

At the bottom of the form is a green button labeled "Ajouter un atelier".

Symfony dispose d'un builder de formulaire qui permet de définir les types de champs du formulaire et d'ajouter des contraintes de validation.

```
● ○ ●
1  <?php
2
3  namespace App\Form;
4
5  use App\Entity\Workshop;
6  use Symfony\Component\Form\AbstractType;
7  use Symfony\Component\Form\Extension\Core\Type\DateType;
8  use Symfony\Component\Form\Extension\Core\Type\TextareaType;
9  use Symfony\Component\Form\Extension\Core\Type\TextType;
10 use Symfony\Component\Form\FormBuilderInterface;
11 use Symfony\Component\OptionsResolver\OptionsResolver;
12 use Symfony\Component\Validator\Constraints\Length;
13 use Symfony\Component\Validator\Constraints\NotBlank;
14
15
16 class WorkshopType extends AbstractType
17 {
18     public function buildForm(FormBuilderInterface $builder, array $options)
19     {
20         $builder
21             ->add('title', TextType::class, [
22                 'label' => 'Titre de l\'atelier *',
23                 'constraints' =>
24                     new NotBlank(),
25             ])
26             ->add('description', TextareaType::class, [
27                 'label' => 'Description de l\'atelier *',
28                 'constraints' =>
29                     new NotBlank(),
30             ])
31             ->add('place', TextType::class, [
32                 'label' => 'Ville *',
33                 'constraints' =>
34                     new NotBlank(),
35             ])
36             ->add('date', TextType::class, [
37                 'label' => 'Date début et de fin de l\'atelier *',
38                 'constraints' => [
39                     new NotBlank(),
40                     new Length([
41                         'min'=> 8,
42                         'minMessage' => 'Il faut mettre au minimum 8 caractères'
43                     ]),
44                 ],
45             ])
46             ->add('deadline', DateType::class, [
47                 'label' => 'Date limite d\'inscription à l\'atelier *',
48                 'constraints' =>
49                     new NotBlank(),
50             ])
51     ;
52 }
53
54 public function configureOptions(OptionsResolver $resolver)
55 {
56     $resolver->setDefaults([
57         'data_class' => Workshop::class,
58     ]);
59 }
60 }
61
```

Voici le code qui permet de valider le formulaire, affiche un message flash et qui redirige sur la page de l'atelier nouvellement créé :

```
 1  /**
 2   * @Route("/ajouter", name="add")
 3   */
 4  public function add(Request $request): Response
 5  {
 6      $workshop = new Workshop();
 7      //Check access authorizations
 8      $this->denyAccessUnlessGranted('ADMIN_WORKSHOP', $workshop);
 9
10     $form = $this->createForm(WorkshopType::class, $workshop);
11
12     $form->handleRequest($request);
13
14     if ($form->isSubmitted() && $form->isValid()) {
15         $em = $this->getDoctrine()->getManager();
16         $em->persist($workshop);
17         $em->flush();
18         //Display success message
19         $this->addFlash(
20             'success',
21             'L\'atelier a bien été ajouté'
22         );
23
24         return $this->redirectToRoute('admin_workshop_read', ['id' => $workshop->getId()]);
25     }
26
27     return $this->render('admin/workshop/add.html.twig', [
28         'form' => $form->createView(),
29     ]);
30 }
```

Voici la page de l'atelier nouvellement créé qui va également afficher les inscriptions associées.

The screenshot shows a mobile application interface. At the top, there is a blue header bar with the text "SALUT LES ZIKETTES!" and a menu icon. Below the header, a dark overlay displays a confirmation message: "Bonjour Manon", "Retour à l'accueil", and "L'atelier a bien été ajouté". To the right of this message is a close button (X). The main content area has a black background. It features a large white title "Atelier Super atelier". Below the title, a red section header reads "Informations relatives à l'atelier:". Under this header, several details are listed in red text: "Description de l'atelier:", "ça va être super chouette!", "Cet atelier a lieu à:", "Paris", "Aux dates:", "Du 13 octobre au 15 octobre 2021", "La date limite d'inscription est le:", "11-10-2021", "Atelier crée le:", and "30-09-2021". At the bottom of the main content area, there is a green section header "Les Inscriptions:".

Il va ensuite s'afficher sur la page des ateliers du front office :

## S'inscrire

### Super atelier

Date limite d'inscription: **11-10-2021**

Cet atelier aura lieu à Paris Du **13 octobre au 15 octobre 2021**

ça va être super chouette!

[S'inscrire](#)

J'ai également mis en place un algorithme dans le controller pour afficher une phrase si aucun atelier n'est ouvert aux inscriptions. Celui-ci récupère la date limite d'inscription et la compare avec la date du jour, si celle-ci est égale ou supérieure avec la date du jour, "1" s'ajoute au compteur. Cela permet de compter le nombre d'ateliers ouverts aux inscriptions.

```

1  /**
2   * @Route("/atelier", name="workshop_")
3  */
4  class WorkshopController extends AbstractController
5  {
6      /**
7       * @Route("", name="browse")
8       */
9      public function browse(WorkshopRepository $workshopRepository): Response
10     {
11         $numberOpenWorkshop = 0;
12         $currentDate = new \DateTime('now');
13         $workshops = $workshopRepository->findAll();
14
15         /* Algo to count the number of open workshops */
16         foreach ($workshops as $workshop){
17             /* We get the deadline of each workshop */
18             $deadline = $workshop->getDeadline();
19
20             /* If the deadline of the workshop is superior or equal than the current date,
21              it's mine that the workshop is open to subscribes */
22             if ( $deadline >= $currentDate ){
23                 /* We had "1" to the variable $numberOpenWorkshop for each open workshop*/
24                 $numberOpenWorkshop += 1;
25             }
26         }
27
28
29         return $this->render('workshop/browse.html.twig', [
30             'workshops' => $workshopRepository->findAll(),
31             'numberOpenWorkshop' => $numberOpenWorkshop,
32
33         ]);
34     }

```

Ensuite dans le template il y a deux conditions, la première affiche les ateliers seulement si la date limite d'inscriptions n'est pas dépassée. La seconde se sert de l'algorithme et affiche une phrase si aucun atelier n'est ouvert aux inscriptions.

Cette méthode n'est certainement pas la plus efficace, nous aurions pu avec plus de temps, par exemple mettre en place une requête SQL personnalisée plus précise pour ne charger que les ateliers ouverts aux inscriptions afin de ne pas charger trop de données inutilement sur cette page.

```

1  {% for workshop in workshops %}
2
3  {% if date(workshop.deadline) > date() %}
4      <div class="inset p-3 mt-5 mb-5 d-grid justify-content-center">
5          <h3 class="text-center pb-3">{{workshop.title}}</h3>
6          <p class="text-center">Date limite d'inscription: <strong>{{workshop.deadline|date('d-m-Y')}}</strong></p>
7          <p>Cet atelier aura lieu à <strong>{{workshop.place}} {{workshop.date}}</strong></p>
8          <p>{{workshop.description}}</p>
9          <button type="button" class="btn btn-lg m-3 mx-auto">
10             {% if is_granted('IS_AUTHENTICATED_FULLY') %}
11                 <a class="btn-a" href="{{ path('workshop_read', {'id': workshop.id}) }}>S'inscrire</a>
12                 {% endif %}
13                 {% if is_granted('IS_ANONYMOUS') %}
14                     <a class="btn-a" href="{{ path('app_login') }}>Se connecter pour s'inscrire</a>
15                 {% endif %}
16             </button>
17         </div>
18     {% endif %}
19
20 {% endfor %}
21
22
23 {% if numberOpenWorkshop == 0 %}
24     <div class="inset p-3 mt-5 mb-5">
25         <p class="">Il n'y a pas d'atelier ouvert aux inscriptions pour le moment</p>
26     </div>
27 {% endif %}

```

Il a fallu également réfléchir à la réutilisation ou non du même formulaire pour le front et le back office en fonction des droits utilisateurs.

Par exemple, il y a trois différents formulaires concernant les utilisateurs :

Le premier, celui de la création de compte n'a besoin de renseigner que le nom, prénom, adresse, email et mot de passe. Il n'est pas pertinent par exemple à ce stade de demander de renseigner un pseudo. En revanche, il est nécessaire que l'utilisateur accepte les termes d'utilisation du site.

Voici le formulaire :

[Retour](#)

# CRÉER UN COMPTE

Prénom \*

Nom \*

E-mail \*

Mot de passe \*

Répète le mot de passe \*

Le mot de passe doit faire 8 caractères minimum,  
pense à utiliser des majuscules, minuscules,  
chiffres et caractères spéciaux pour plus de  
sécurité

[Conditions générales](#)

Accepter les conditions générales d'utilisation \*

[Créer un compte](#)

Et son code :

```

1 <?php
2
3 namespace App\Form;
4
5 use App\Entity\User;
6 use Symfony\Component\Form\AbstractType;
7 use Symfony\Component\Form\Extension\Core\Type\CheckboxType;
8 use Symfony\Component\Form\Extension\Core\Type\EmailType;
9 use Symfony\Component\Form\Extension\Core\Type\PasswordType;
10 use Symfony\Component\Form\Extension\Core\Type\RepeatedType;
11 use Symfony\Component\Form\Extension\Core\Type\TextType;
12 use Symfony\Component\Form\FormBuilderInterface;
13 use Symfony\Component\OptionsResolver\OptionsResolver;
14 use Symfony\Component\Validator\Constraints\IsTrue;
15 use Symfony\Component\Validator\Constraints\Length;
16 use Symfony\Component\Validator\Constraints\NotBlank;
17
18 class RegistrationFormType extends AbstractType
19 {
20     public function buildForm(FormBuilderInterface $builder, array $options)
21     {
22         $builder
23             ->add('email', EmailType::class, [
24                 'label' => 'E-mail *',
25             ])
26             ->add('agreeTerms', CheckboxType::class, [
27                 'label' => 'Accepter les conditions générales d\'utilisation *',
28                 'mapped' => false,
29                 'constraints' => [
30                     new IsTrue([
31                         'message' => 'Tu dois accepter les conditions générales d\'utilisation pour pouvoir t\'inscrire',
32                     ]),
33                 ],
34             )
35             ->add('plainPassword', RepeatedType::class, [
36                 'label' => 'Mot de passe *',
37                 // instead of being set onto the object directly,
38                 // this is read and encoded in the controller
39                 'mapped' => false,
40                 'attr' => ['autocomplete' => 'new-password'],
41                 'type' => PasswordType::class,
42                 'first_options' => ['label' => 'Mot de passe *'],
43                 'second_options' => ['label' => 'Répète le mot de passe *'],
44                 'invalid_message' => 'Les mots de passe doivent être identiques.',
45                 'constraints' => [
46                     new NotBlank([
47                         'message' => 'Merci d\'entrer un mot de passe' ,
48                     ]),
49                     new Length([
50                         'min' => 8,
51                         'minMessage' => 'Ton mot de passe doit faire 8 caractères minimum',
52                         // max length allowed by Symfony for security reasons
53                         'max' => 4096,
54                     ]),
55                 ],
56             ])
57             ->add('firstname', TextType::class, [
58                 'label' => 'Prénom *',
59                 'constraints' => [
60                     new NotBlank(),
61                 ],
62             ])
63             ->add('lastname', TextType::class, [
64                 'label' => 'Nom *',
65                 'constraints' => [
66                     new NotBlank(),
67                 ],
68             ])
69
70     ;
71 }
72
73 public function configureOptions(OptionsResolver $resolver)
74 {
75     $resolver->setDefaults([
76         'data_class' => User::class,
77     ]);
78 }
79 }
```

Le deuxième formulaire concerne la modification des informations du compte utilisateur. Pour celui-ci, on affiche donc le champ pseudo, il n'est en revanche pas pertinent de lui demander d'accepter à nouveau les termes d'utilisation.

Voici le formulaire :

The screenshot shows a mobile application interface for editing a user profile. At the top, a green header bar displays the text "Bonjour Manon" on the left and a close button "X" on the right. Below the header, a pink button labeled "Retour" is visible. The main content area has a dark background and features a large teal title "MON COMPTE" centered at the top. Below the title, there are four input fields with labels: "Prénom" (First Name) with the value "Manon", "Nom" (Last Name) with the value "Le Bars", "E-mail" (Email) with the value "lebarsmanon1@gmail.com", and "Pseudo" (Nickname). The "Pseudo" field is currently empty. Below these fields is a instruction "Confirme ton mot de passe pour effectuer les modifications" (Confirm your password to perform the modifications), followed by a password input field and a "Modifier" (Update) button. A secondary button labeled "Modifier le mot de passe" (Change password) is also present.

Et son code :



```
1  <?php
2
3  namespace App\Form;
4
5  use App\Entity\User;
6  use Symfony\Component\Form\AbstractType;
7  use Symfony\Component\Form\Extension\Core\Type\EmailType;
8  use Symfony\Component\Form\Extension\Core\Type>PasswordType;
9  use Symfony\Component\Form\Extension\Core\Type\TextType;
10 use Symfony\Component\Form\FormBuilderInterface;
11 use Symfony\Component\OptionsResolver\OptionsResolver;
12 use Symfony\Component\Validator\Constraints\Length;
13 use Symfony\Component\Validator\Constraints\NotBlank;
14
15
16 class UserType extends AbstractType
17 {
18     public function buildForm(FormBuilderInterface $builder, array $options)
19     {
20         $builder
21             ->add('email', EmailType::class, [
22                 'label' => 'E-mail',
23             ])
24             ->add('firstname', TextType::class, [
25                 'label' => 'Prénom',
26                 'constraints' => [
27                     new NotBlank(),
28                 ]
29             ])
30             ->add('lastname', TextType::class, [
31                 'label' => 'Nom',
32                 'constraints' => [
33                     new NotBlank(),
34                 ]
35             ])
36             ->add('username', TextType::class, [
37                 'label' => 'Pseudo',
38                 'required' => false,
39             ])
40             ->add('password', PasswordType::class, [
41                 'label' => 'Confirme ton mot de passe pour effectuer les modifications',
42                 'mapped' => false,
43                 'invalid_message' => 'Mot de passe invalide',
44                 'constraints' => [
45                     new Length([
46                         'min' => 8,
47                         'minMessage' => 'Ton mot de passe doit faire 8 caractères minimum',
48                         // max length allowed by Symfony for security reasons
49                         'max' => 4096,
50                     ]),
51                 ],
52             ])
53         ;
54     }
55 }
56
57 public function configureOptions(OptionsResolver $resolver)
58 {
59     $resolver->setDefaults([
60         'data_class' => User::class,
61     ]);
62 }
63 }
```

Enfin le dernier concerne la gestion des utilisateurs dans la partie admin.

Ici tous les champs sont présents (hormis l'acceptation des termes d'utilisations) pour permettre aux admins de créer de nouveaux utilisateurs, mais également le champ rôle qui va permettre aux utilisatrices de choisir le rôle de chaque utilisateur. Celui-ci n'était pas présent dans les autres formulaires afin d'éviter que des utilisateurs s'arrogent de droits admins sans leur autorisation par exemple. A chaque création de compte, c'est automatiquement le rôle user qui est attribué.

Voici le formulaire et son code :

Bonjour Manon

[Retour à l'accueil](#)

## Ajouter une utilisatrice

Prénom \*

Nom \*

Email \*

Mot de passe \*

Répète le mot de passe \*

Le mot de passe doit faire 8 caractères minimum, pense à utiliser des majuscules, minuscules, chiffres et caractères spéciaux pour plus de sécurité

Rôle \*

- Administrateur
- Modérateur
- Membre
- Utilisateur

**Ajouter une utilisatrice**

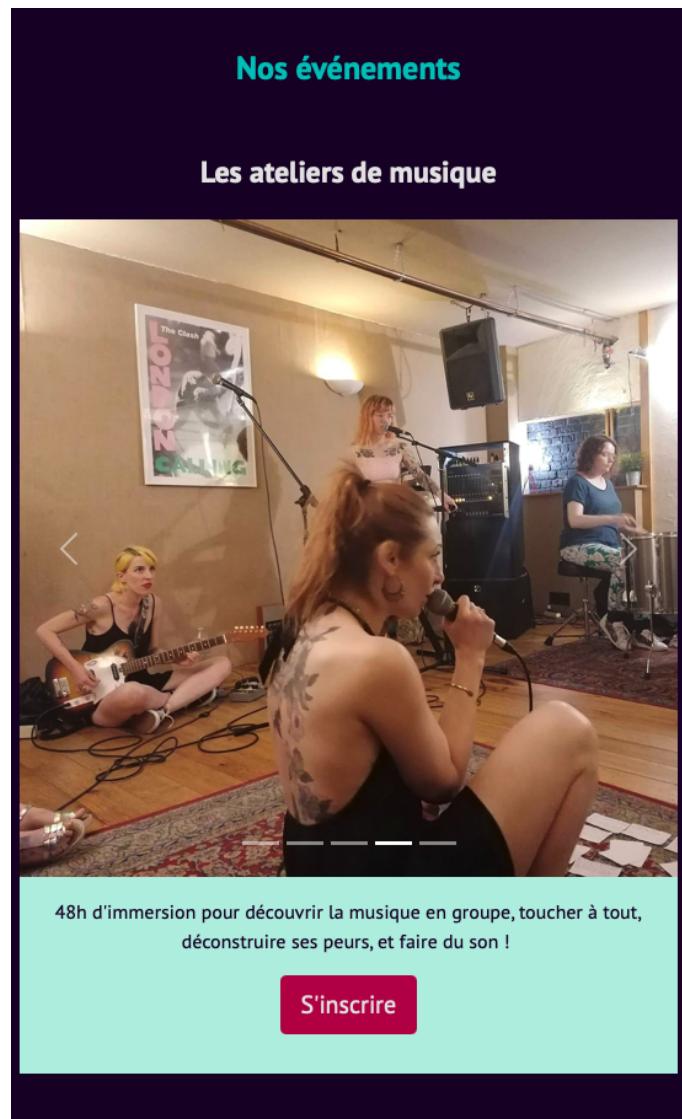
```

1 <?php
2
3 namespace App\Form;
4
5 use App\Entity\User;
6 use Symfony\Component\Form\AbstractType;
7 use Symfony\Component\Form\CallbackTransformer;
8 use Symfony\Component\Form\Extension\Core\Type\ChoiceType;
9 use Symfony\Component\Form\Extension\Core\Type\EmailType;
10 use Symfony\Component\Form\Extension\Core\Type>PasswordType;
11 use Symfony\Component\Form\Extension\Core\Type\RepeatedType;
12 use Symfony\Component\Form\Extension\Core\Type\TextType;
13 use Symfony\Component\Form\FormBuilderInterface;
14 use Symfony\Component\OptionsResolver\OptionsResolver;
15 use Symfony\Component\Validator\Constraints\NotBlank;
16
17 class AdminUserType extends AbstractType
18 {
19     public function buildForm(FormBuilderInterface $builder, array $options)
20     {
21         $builder
22             ->add('email', EmailType::class, [
23                 'label' => 'Email *',
24                 'constraints' => [
25                     new NotBlank(),
26                 ]
27             ])
28             ->add('roles', ChoiceType::class, [
29                 'label' => 'Rôle *',
30                 'choices' => [
31                     'Administrateur' => 'ROLE_ADMIN',
32                     'Modérateur' => 'ROLE_MODERATOR',
33                     'Membre' => 'ROLE_MEMBER',
34                     'Utilisateur' => 'ROLE_USER',
35                 ],
36                 'multiple' => false,
37                 'expanded' => true,
38             ])
39             ->add('firstname', TextType::class, [
40                 'label' => 'Prénom *',
41                 'constraints' => [
42                     new NotBlank(),
43                 ]
44             ])
45             ->add('lastname', TextType::class, [
46                 'label' => 'Nom *',
47                 'constraints' => [
48                     new NotBlank(),
49                 ]
50             ])
51             ->add('username', TextType::class, [
52                 'label' => 'Pseudo',
53                 'required' => false,
54             ])
55         ]
56         ->add('password', RepeatedType::class, [
57             'mapped' => false,
58             'type' => PasswordType::class,
59             'required' => false,
60             'first_options' => ['label' => 'Mot de passe *'],
61             'second_options' => ['label' => 'Répète le mot de passe *'],
62             'invalid_message' => 'Les mots de passe doivent être identiques.'
63         ])
64
65         ->get('roles')->addModelTransformer(new CallbackTransformer(
66             // Transforme la données de l'entité pour le formulaire
67             function (array $rolesArray): string {
68                 // retourne le premier role dans le tableau
69                 // Si le tableau est vide, l'index 0 n'existe et on retourne une string vide
70                 return $rolesArray[0] ?? '';
71             },
72             // Transformer la donnée du formulaire pour l'entité
73             function (string $roleString): array {
74                 // retourne la string dans un tableau
75                 return [$roleString];
76             }
77         ))
78     ;
79 }
80
81     public function configureOptions(OptionsResolver $resolver)
82     {
83         $resolver->setDefaults([
84             'data_class' => User::class,
85         ]);
86     }
87 }
```

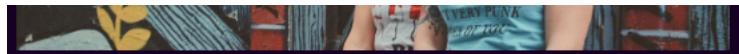
## G/ Jeu d'essai

L'une des fonctionnalités centrales de ce site est de pouvoir s'inscrire à un atelier. Je vous ai montré la création d'un atelier dans la partie "mes réalisations personnelles" "les formulaire". Pour ce jeu d'essai je vais donc simuler une utilisatrice qui découvre le site et souhaite s'inscrire à un atelier, puis la validation ou non de cette inscription par une admin.

Sur la page d'accueil du site je découvre parmi les événements proposés, les ateliers de musique.



Je clique sur le bouton "s'inscrire" qui me redirige vers la page Atelier du site. J'y découvre une description détaillée des ateliers ainsi qu'un atelier qui est ouvert aux inscriptions !



## LES ATELIERS DE MUSIQUE



Les ateliers se déroulent sur 2 jours, en général sur un week-end, de 11h à 19h. Les participantes (6 maximum) sont des femmes, trans ou personnes non-binaires qui n'ont jamais ou très peu pratiqué la musique en groupe, mais souhaiteraient le faire. Chaque atelier est composé de plusieurs temps forts, notamment des temps de parole (échanges et partages, expression libre des expériences que les participantes souhaiteraient partager) et des temps de pratique des instruments mis à disposition : guitare, basse, batterie, clavier et chant.

personnes non-binaires qui n'ont jamais ou très peu pratiqué la musique en groupe, mais souhaiteraient le faire. Chaque atelier est composé de plusieurs temps forts, notamment des temps de parole (échanges et partages, expression libre des expériences que les participantes souhaiteraient partager) et des temps de pratique des instruments mis à disposition : guitare, basse, batterie, clavier et chant. L'objectif est d'amener les participantes à découvrir les instruments proposés, à se sentir à l'aise de pratiquer la musique en groupe, puis à composer des morceaux dans un groupe à géométrie variable. Aussi, de constituer et de pérenniser une communauté de femmes musiciennes et un réseau d'entraide.

## S'inscrire

### Super atelier

Date limite d'inscription: **11-10-2021**

Cet atelier aura lieu à **Paris Du 13 octobre au 15 octobre 2021**

ça va être super chouette!

[Se connecter pour s'inscrire](#)



[Mentions Légales](#)

©2021 par Salut les Zikettes!

N'étant pas connecté au site, le bouton me demande de me connecter pour m'inscrire.



Retour

## CONNEXION

E-mail

Mot de passe

[Mot de passe oublié?](#)

[Connexion](#)

[Créer un Compte](#)



[Mentions Légales](#)

N'ayant pas de compte, j'en crée un.

[Retour](#)

## CRÉER UN COMPTE

Prénom \*

Nom \*

E-mail \*

Mot de passe \*

Répète le mot de passe \*

Le mot de passe doit faire 8 caractères minimum,  
pense à utiliser des majuscules, minuscules,  
chiffres et caractères spéciaux pour plus de  
sécurité

Conditions générales

Accepter les conditions générales d'utilisation \*

[Créer un compte](#)

Une fois mon compte créé, je me connecte et peut maintenant m'inscrire directement.

[S'inscrire](#)

### Super atelier

Date limite d'inscription: **11-10-2021**

Cet atelier aura lieu à **Paris Du 13 octobre au 15 octobre 2021**

ça va être super chouette!

[S'inscrire](#)

J'accède alors au formulaire d'inscription de cet atelier

The screenshot shows a mobile application interface. At the top, there is a header with the text "SALUT LES ZIKETTES!" and a menu icon. Below the header is a photograph of two women standing in front of a wall covered in graffiti. A green overlay bar at the bottom of the screen displays the text "Bonjour Maoui" on the left and a close button "X" on the right. The main content area has a dark background and features the following text:  
**ATELIER SUPER ATELIER**  
Date limite d'inscription: **11-10-2021**  
Cet atelier aura lieu à Paris Du **13 octobre au 15 octobre 2021**  
ça va être super chouette!  
\* Champs obligatoires  
Date de Naissance \*  
Jan 1 1901  
Ville de résidence \*  
Par quel prénom souhaitez-tu que l'on s'adresse à toi?  
[Empty input field]

Jan ▾ 1 ▾ 1901 ▾

Ville de résidence \*

Par quel prénom souhaites-tu que l'on s'adresse à toi?

As-tu déjà pratiqué un instrument (la voix étant aussi un instrument)? Si oui, depuis combien de temps et dans quel cadre? \*

Peux-tu nous dire, en quelques mots, ta motivation et/ou tes attentes concernant l'atelier? \*

Comment as-tu connu "Salut Les Zikettes!"?

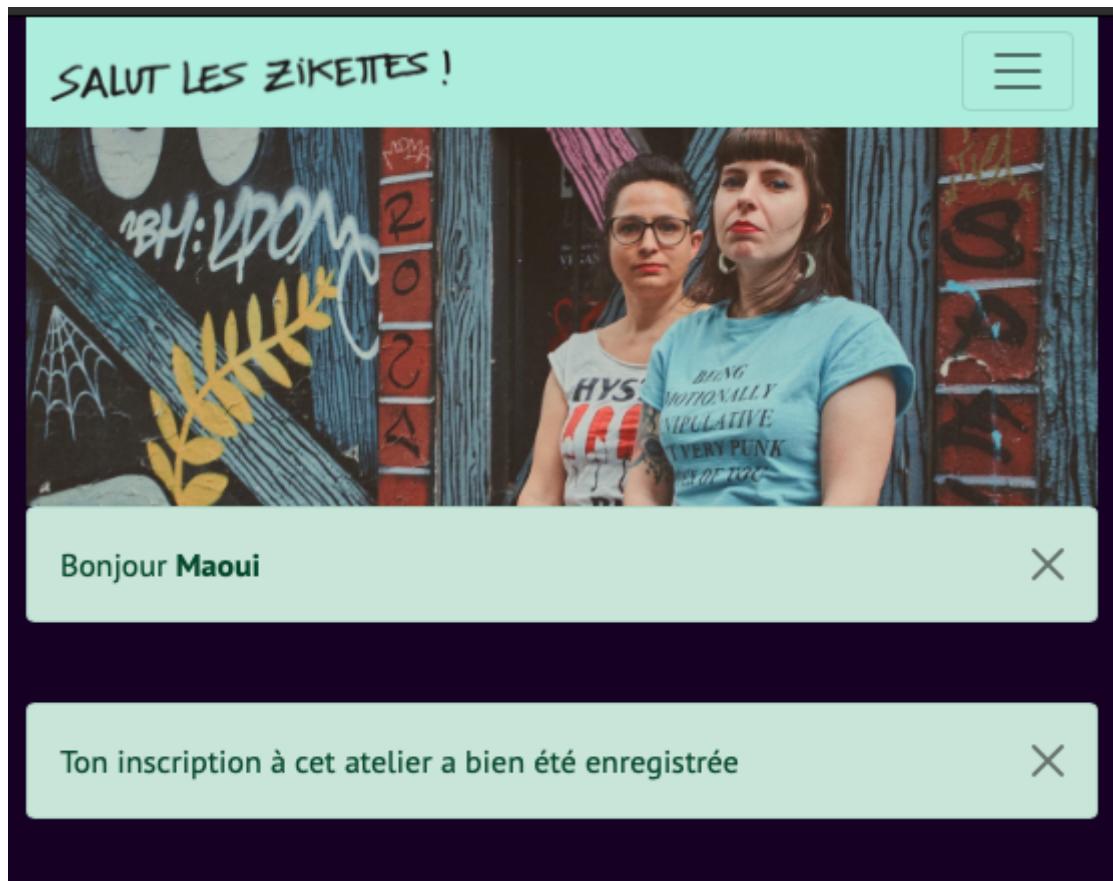
[Je m'inscris](#)

[Mentions Légales](#)

©2021 par Salut les Zikettes!

Une fois le formulaire validé, j'ai un message pop-up qui m'avertit que mon inscription a bien été enregistrée.



Maintenant en tant qu'admin si je veux accéder aux inscriptions, j'ai deux possibilités, je peux soit consulter toutes les inscriptions dans la page "Inscriptions aux ateliers" comme ceci :

SALUT LES ZIKETTES!

Bonjour **Manon**

[Retour à l'accueil](#)

[Retour](#)

# Les Inscriptions aux ateliers

## Inscriptions à l'atelier Super atelier qui à lieu à Paris

**Candidate:**

Prénom:  
Maoui

Nom:  
Manon

Date de Naissance:  
01-01-1992

Ville de résidence:  
Paris

Par quel prénom souhaites-tu que l'on s'adresse à toi ?:

As-tu déjà pratiqué un instrument (la voix étant aussi un

Par quel prénom souhaites-tu que l'on s'adresse à toi ?:

As-tu déjà pratiqué un instrument (la voix étant aussi un instrument)? Si oui, depuis combien de temps et dans quel cadre?:

Oui mais je chante faux

Peux-tu nous dire, en quelques mots, ta motivation et/ou tes attentes concernant l'atelier?:

Je suis super motivée!!

Comment as tu connu "Salut Les Zikettes!":

Date d'inscription:

01-10-2021

État de l'inscription:

En attente de validation

**Valider**

**Supprimer**

Soit, deuxième possibilité : Je peux choisir de me rendre dans la page “Ateliers” de l’admin pour cibler les inscriptions à un ateliers en particulier ...

The screenshot shows a mobile application interface. At the top, a blue header bar displays the text "SALUT LES ZIKETTES !". To the right of the header is a menu icon consisting of three horizontal lines and a close button (an 'X'). Below the header, a dark navigation bar contains the text "Bonjour Manon" and a link "Retour à l'accueil". On the left side of the main content area is a blue "Retour" button. The main content area features a large white title "Les ateliers". Below this is a green button labeled "Ajouter un atelier". A red-bordered box contains detailed information about a workshop titled "Super atelier":  
**Description de l'atelier:**  
ça va être super chouette!  
**Cet atelier à lieu à:**  
Paris  
**Aux dates:**  
Du 13 octobre au 15 octobre 2021  
**La date limite d'inscription est le:**  
11-10-2021  
**Atelier crée le:**  
30-09-2021  
At the bottom of this red-bordered box are three buttons: "Accéder aux inscriptions" (green), "Modifier" (orange), and "Supprimer" (red).

... Et en cliquant sur "Accéder aux inscriptions", je peux consulter l'ensemble des informations du dit atelier, suivi de toutes les inscriptions associées.

Bonjour Manon  
[Retour à l'accueil](#)

X

# Atelier Super atelier

## Informations relatives à l'atelier:

Description de l'atelier:

ça va être super chouette!

Cet atelier a lieu à:

Paris

Aux dates:

Du 13 octobre au 15 octobre 2021

La date limite d'inscription est le:

11-10-2021

Atelier créé le:

30-09-2021

## Les Inscriptions:

Candidate:

## Candidate:

Prénom:

Maoui

Nom:

Manon

Date de Naissance:

01-01-1992

Ville de résidence:

Paris

Par quel prénom souhaites-tu que l'on s'adresse à toi ?:

As-tu déjà pratiqué un instrument (la voix étant aussi un instrument!)? Si oui, depuis combien de temps et dans quel cadre?:

Oui mais je chante faux

Peux-tu nous dire, en quelques mots, ta motivation et/ou tes attentes concernant l'atelier?:

Je suis super motivée!!

Comment as tu connu "Salut Les Zikettes!":

Date d'inscription:

01-10-2021

État de l'inscription:

En attente de validation

**Valider**

**Supprimer**

Je peux modifier l'état de l'inscription en cliquant sur le bouton “valider”

# Veuillez une inscription à un atelier

Nom et prénom

Maoui Manon

Veut s'inscrire à l'atelier:

Super atelier

Qui a lieu à:

Paris

Date de Naissance:

01-01-1992

Ville de résidence:

Paris

Par quel prénom souhaitez-tu que l'on s'adresse à toi ?:

As-tu déjà pratiqué un instrument (la voix étant aussi un instrument!)? Si oui, depuis combien de temps et dans quel cadre?:

Oui mais je chante faux

Peux-tu nous dire, en quelques mots, ta motivation et/ou tes attentes concernant l'atelier?:

Je suis super motivée!!

Comment as tu connu "Salut Les Zikettes!"

Validation de l'inscription

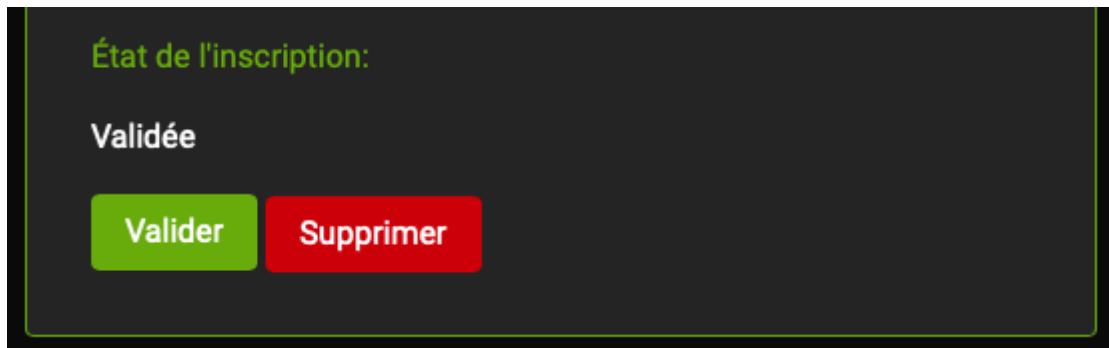
- Valider
- Refuser

**Valider**

Une fois le formulaire validé j'ai un message pop-up qui me confirme que l'inscription a bien été modifiée.



Je peux vérifier l'inscription :



Maoui est inscrite à "l'atelier Super" !

# H/ Veilles

## Description de la veille

Tout au long du projet nous avons effectué une veille informatique.

Cela nous a servi pour mettre de côté des liens de recherches qui nous ont été utiles plus tard dans l'avancement du projet. Par exemple, lors d'une discussion au début du sprint 0 nous avons évoqué les mauvaises expériences utilisateurs pour sélectionner une date dans un formulaire. Nous avons donc fait une recherche et gardé le lien pour le moment où nous avons développé les formulaires :

- <https://symfony.com/doc/current/reference/forms/types/birthday.html>

Nous avons également fait une recherche sur la mise en place des notifications avec Symfony :

- <https://symfony.com/doc/current/notifier.html>
- , <https://symfony.com/doc/current/the-fast-track/fr/25-notifier.html>

Cette veille nous a également permis de mettre de côté des liens utiles tout au long du projet, par exemple un récapitulatif des commandes git:

- <https://kourou.oclock.io/ressources/fiche-recap/branches/>

ou bien une convention de nommage pour les branches et commit sur laquelle nous nous sommes mises d'accord:

- [Git : comment nommer ses branches et ses commits ?](#)

Cela nous a également permis de garder des liens de documentation sur des grosses fonctionnalités qui prenaient plusieurs jours afin de pouvoir revenir dessus facilement et d'être accessible à toute l'équipe. Par exemple pour la mise en place des formulaires :

- <https://symfony.com/doc/current/reference/constraints.html>
- <https://symfony.com/doc/current/reference/forms/types.html>
- [https://symfony.com/doc/current/best\\_practices.html#forms](https://symfony.com/doc/current/best_practices.html#forms)
- [https://symfony.com/doc/current/form/form\\_customization.html#reference-forms-twig-form](https://symfony.com/doc/current/form/form_customization.html#reference-forms-twig-form)
- <http://www.lisis.org/elmouelhia/courses/php/sf/coursSymfonyFormulaire.pdf>
- <https://www.kaherecode.com/tutorial/creer-un-blog-avec-symfony-les-formulaires>
- <https://symfony.com/doc/current/form/bootstrap5.html>

Enfin la veille a été essentielle pour faire notre choix pour savoir ou déployer le site, puis d'y arriver :

- <https://www.kimsufi.com/fr/>
- <https://www.netlify.com/products/>

- <https://www.ovh.com/fr/>
- <https://www.gandi.net/fr/>
- <https://www.namecheap.com/>
- <https://www.digitalocean.com/>
- <https://devcenter.heroku.com/articles/deploying-symfony4>
- <https://www.heroku.com/>
- <https://spawn.cc/>
- [https://www.youtube.com/watch?v=sxH\\_0uSft3M](https://www.youtube.com/watch?v=sxH_0uSft3M)
- <https://www.youtube.com/watch?v=mBCH9OTVaGw>
- <https://www.youtube.com/watch?v=XEdFtzq0RYo>
- [watch?v=aEm0BN493sU](https://www.youtube.com/watch?v=aEm0BN493sU)
- <https://devcenter.heroku.com/articles/custom-domains>

## Veille significative

Une partie importante de la veille effectuée a été relative à la mise en place d'envoi d'email. Il était essentiel au vu des fonctionnalités proposées que l'envoi de courriel fonctionne. Notamment, pour le formulaire de contact, les admins doivent absolument recevoir les messages qui leur sont envoyés, mais également pour la partie forum, il faut que les utilisatrices puissent être tenu au courant des réponses à leur post et qu'elles puissent signaler des posts ou commentaires aux admins.

Pour cela nous avons utilisé en local Swift Mailer (librairie orientée objet qui permet l'envoi d'email) : <https://symfony.com/doc/current/email.html> et nous sommes servis de Mailtrap pour faire nos tests, c'est un outil qui permet de récupérer sur une interface web, le contenu de l'email expédié depuis le site : <https://mailtrap.io/blog/send-emails-in-symfony/>.

En production, l'envoi d'email ne fonctionnant plus, nous nous sommes tourné vers le SMTP de Gmail :

<https://dev-yakuza.posstree.com/en/django/gmail-smtp/?fbclid=IwAR06e2VQgbSVIzbN3ZhF27leGGF6vzv74goRYgCDfagOKxWnRdrdnYD4-So.>

Enfin, pour des raisons de sécurités nous avons mis en place la vérification des emails lors des créations de compte :

[https://symfony.com/doc/current/security/user\\_checkers.html](https://symfony.com/doc/current/security/user_checkers.html)

Cela permet d'envoyer un email lors de la création de compte, une fois que l'utilisateur clique sur le lien reçu dans l'e-mail, il obtient le statut "is verified" en base de données qui lui permet ensuite de se connecter au site.

Toujours dans un souci de sécurité, nous avons ajouté la possibilité de réinitialiser son mot de passe en cas d'oubli : <https://blog.dayo.fr/2020/05/symfony-makeuser-auth-registration-form-et-reset-password/>.

## Situation de recherche

Comme vu précédemment la mise en place de la vérification d'email a été une recherche clé, voici l'article provenant du site [https://symfony.com/doc/current/security/user\\_checkers.html](https://symfony.com/doc/current/security/user_checkers.html)

## “How to Create and Enable Custom User Checkers

During the authentication of a user, additional checks might be required to verify if the identified user is allowed to log in. By defining a custom user checker, you can define per firewall which checker should be used.

### Creating a Custom User Checker

User checkers are classes that must implement the `Symfony\Component\Security\Core\User\UserCheckerInterface`. This interface defines two methods called `checkPreAuth()` and `checkPostAuth()` to perform checks before and after user authentication. If one or more conditions are not met, throw an exception which extends the `Symfony\Component\Security\Core\Exception\AccountStatusException` class. Consider using `Symfony\Component\Security\Core\Exception\CustomUserMessageAccountStatusException`, which extends `AccountStatusException` and allows to customize the error message displayed to the user:

```

1 <?php
2
3 namespace App\Security;
4
5 use App\Entity\User as AppUser;
6 use Symfony\Component\Security\Core\Exception\AccountExpiredException;
7 use Symfony\Component\Security\Core\Exception\CustomUserMessageAccountStatusException;
8 use Symfony\Component\Security\Core\User\UserCheckerInterface;
9 use Symfony\Component\Security\Core\User\UserInterface;
10
11 class UserChecker implements UserCheckerInterface
12 {
13     public function checkPreAuth(UserInterface $user): void
14     {
15         if (!$user instanceof AppUser) {
16             return;
17         }
18
19         if ($user->isDeleted()) {
20             // the message passed to this exception is meant to be displayed to the user
21             throw new CustomUserMessageAccountStatusException('Your user account no longer exists.');
22         }
23     }
24
25     public function checkPostAuth(UserInterface $user): void
26     {
27         if (!$user instanceof AppUser) {
28             return;
29         }
30
31         // user account is expired, the user may be notified
32         if ($user->isExpired()) {
33             throw new AccountExpiredException('...');
34         }
35     }
36 }

```

## Enabling the Custom User Checker

Next, make sure your user checker is registered as a service. If you're using the default [services.yaml configuration](#), the service is registered automatically.

All that's left to do is add the checker to the desired firewall where the value is the service id of your user checker:"

```

1 # config/packages/security.yaml
2
3 # ...
4 security:
5     firewalls:
6         main:
7             pattern: ^/
8             user_checker: App\Security\UserChecker
9             # ...

```

Sa traduction en français

## « Comment créer et activer des vérificateurs d'utilisateurs personnalisés

Lors de l'authentification d'un utilisateur, des contrôles supplémentaires peuvent être mis en place pour vérifier si l'utilisateur identifié est autorisé à se connecter. En définissant un vérificateur d'utilisateur personnalisé, vous pouvez définir par quel pare-feu le vérificateur doit être utilisé.

### Créer un vérificateur d'utilisateur personnalisé

Les vérificateurs d'utilisateurs sont des classes qui doivent implémenter ceci : `Symfony\Component\Security\Core\User\UserCheckerInterface`. Cette interface définit deux méthodes appelées `checkPreAuth()` et `checkPostAuth()` pour effectuer des vérifications avant et après l'authentification de l'utilisateur. Si une ou plusieurs conditions ne sont pas remplies, lancez une exception qui étend la classe : `Symfony\Component\Security\Core\Exception\AccountStatusException`. Pensez à utiliser `Symfony\Component\Security\Core\Exception\CustomUserMessageAccountStatusException`, qui étend `AccountStatusException` et permet de personnaliser le message d'erreur affiché à l'utilisateur :

### Activer le vérificateur d'utilisateur personnalisé

Ensuite, assurez-vous que votre vérificateur d'utilisateurs soit défini en tant que service. Si vous utilisez par défaut la configuration de services.yaml, le service est défini automatiquement.

Enfin, il ne reste plus qu'à ajouter le vérificateur au pare-feu souhaité où la valeur est l'identifiant du service de votre vérificateur d'utilisateur :"

## Conclusion

Comme son nom l'indique, ce projet a été l'apothéose de ces 5 mois de formation intenses. Cela a été très enrichissant de pouvoir mettre en pratique pendant un mois tout ce que nous avons pu apprendre, puis de pousser pour découvrir de nouvelles fonctionnalités.

Lors de ce projet nous avons rencontré des difficultés, notamment au niveau du paramétrage en fonction des rôles (qui a accès à quoi) avec la mise en place des voters. L'association nous a demandé des couleurs spécifiques qu'il a fallu adapter pour obtenir un bon niveau de contraste pour une meilleure expérience utilisateur. Les relations entre les données ont également demandé beaucoup de réflexion, par exemple lier une inscription à un atelier et à un utilisateur. Enfin les formulaires nous ont demandé beaucoup de recherche, de test et de réflexion, notamment sur la réutilisation ou non du même formulaire pour le front et le back office en fonction des droits utilisateurs.

Ce projet a été très motivant, il nous a apporté de l'autonomie et plus de confiance en nos capacités nous donnant ainsi un regain de motivation en fin de formation. L'un des points que nous avons appréciés, est que nous avons vraiment pensé l'application pour les utilisateurs finaux et avons donc apporté plus de finitions que si nous avions travaillé sur un projet fictif par exemple. Il a également été très intéressant de travailler en collaboration avec l'association et selon ces besoins. Nous avons de plus eu la chance d'avoir le temps de découvrir de nouvelles fonctionnalités seules et de les mettre en place dans notre projet.

Cette application sera utilisée dans le futur par l'association, quand celle-ci aura les moyens financiers nécessaires pour assumer l'hébergement, et sera maintenue par moi-même.