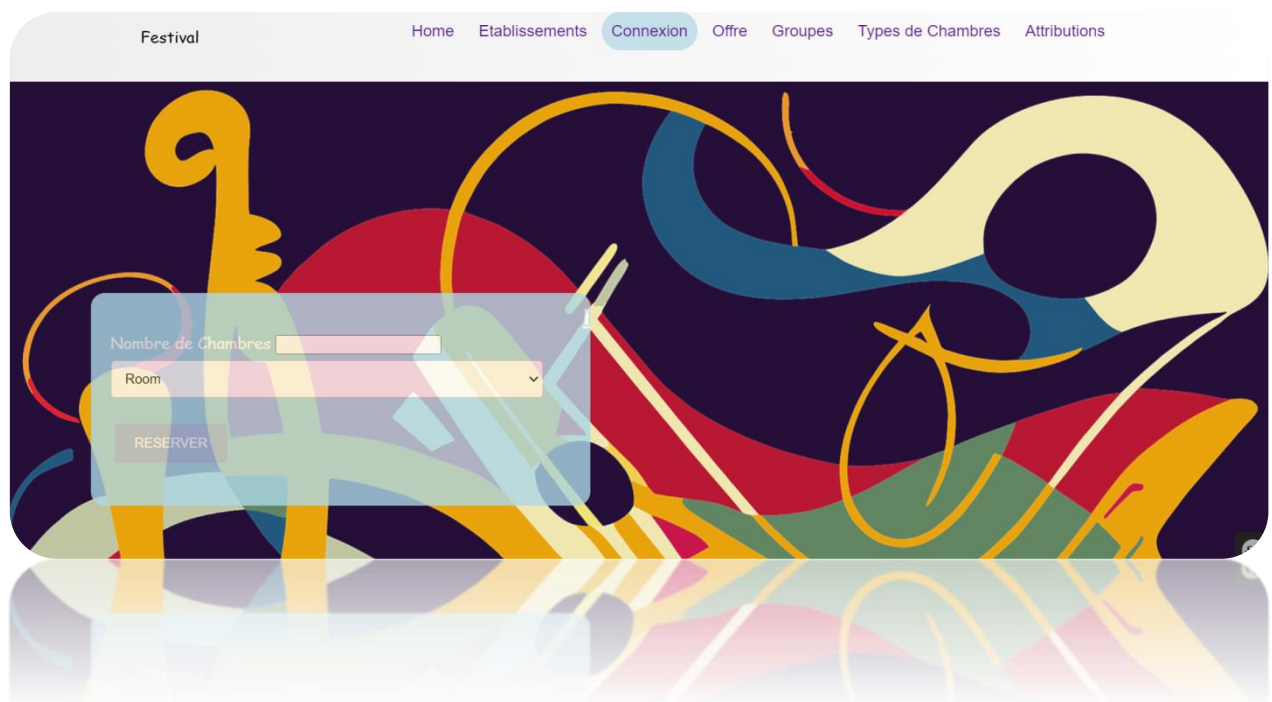


# SITE WEB FESTIVAL

---

## DOCUMENTATION TECHNIQUE



Sahli Dounia  
BTS SIO

# Table des matières

<b>DOCUMENTATION TECHNIQUE .....</b>	<b>1</b>
Sahli Dounia .....	1
BTS SIO .....	1
CONTEXTE .....	3
OUTILS UTILISE. ....	4
CREATION DU PROJET. ....	5
SECURITER USER ET AUTHENTIFICATION.....	6
CREATION DES ENTITES.....	9
UTILISATION DU CRUD. ....	11

## CONTEXTE .

La maison des Ligues (M2L) est une structure financée à 100% par le Conseil Régional de Lorraine qui fournit des espaces et des services aux différentes ligues sportives régionales. Celle-ci a fait appel à notre entreprise DJDEVOPS (entreprise fictive) pour la mise en place d'une solution applicative pour l'organisation du festival Folklores de la Maison des Ligues de Saint Malo.

Le festival de musique Folklores du Monde est un festival de musique se déroulant tous les ans durant la première semaine de juillet dans la ville bretonne de Saint-Malo. Il permet de réunir différents groupes venant de France mais aussi du reste du monde. La ville s'attache à bien recevoir les groupes et prend notamment en charge l'hébergement durant leur séjour. Les groupes sont invités pour la semaine entière ou pour une partie de la semaine. L'hébergement des groupes est assuré par divers établissements d'accueil. Ces entités sont sollicitées pour mettre à disposition les chambres de leur établissement. La Maison des Ligues, service municipal dépendant de la direction du développement culturel se charge de la préparation du festival .On lui demande de développer un site web permettant à l'administrateur d'organiser les hébergements et de les gérer. L'utilisateur, lui pourra s'inscrire sur le site et réserver son hébergement.

## OUTILS UTILISE.

### XAMP



Xamp est en mesure de lancer des sites web dynamiques qu'il soit connecté à Internet ou en local. Il peut être utilisé avec de simples navigateurs et contient un serveur Apache permettant de transférer des fichiers en ligne. Il intègre PhpMyAdmin, SGBD fonctionnant sur le langage Mysql.

### SYMFONY



Symfony est un ensemble de composants PHP ainsi qu'un framework MVC libre écrit en PHP. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d'accélérer le développement d'un site web. Je l'ai choisi car il possède une grande documentation.

### VISUAL STUDIO CODE



Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré.

## CREATION DU PROJET.

Dans l'invite de commande (cmd), nous allons créer l'application Symfony grâce à la commande :

```
composer create projet symfony/website skeleton siteFestival
```

Elle va créer un nouveau siteFestival/répertoire, y téléchargera les dépendances et générera les répertoires et fichiers de base dont nous auront besoin pour commencer.

## CONFIGURATION DU .env:

Configuration initiale :



```
# DATABASE_URL="mysql://db_user:db_password@127.0.0.1:3306/db_name?serverVersion=5.7"
```

Après modification :

```
DATABASE_URL="mysql://root@127.0.0.1:3306/symfony?serverVersion=mariadb-10.4.21"
```

Symfony (nom de la base de données) db\_password(aucun mot de passe).  
127.0.0.1:8000 (adresse IP de la base de données en local). Festival (nom du projet créé au préalable avec phpmyadmin)

## Bases de données

 Création d'une base de données 

<input type="text" value="Symfony"/>	<input type="text" value="utf8mb4_general_ci"/>	<input type="button" value="Créer"/>
--------------------------------------	---	--------------------------------------

Une fois la configuration terminée nous pouvons nous connecter avec la commande en console :

```
Php bin/console doctrine :database :create
```

## SECURITER USER ET AUTHENTIFICATION.

### Commande en console :

```
php bin/console make:user
```

Cela va créer l'Entity **User**, l'**UserRepository** et modifier le **config/packages/security.yaml**

```
class UserRepository extends ServiceEntityRepository implements PasswordUpgraderInterface
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, User::class);
    }

    public function save(User $entity, bool $flush = false): void
    {
        $this->getEntityManager()->persist($entity);

        if ($flush) {
            $this->getEntityManager()->flush();
        }
    }

    public function remove(User $entity, bool $flush = false): void
    {
        $this->getEntityManager()->remove($entity);

        if ($flush) {
            $this->getEntityManager()->flush();
        }
    }

    /**
     * Used to upgrade (rehash) the user's password automatically over time.
     */
    public function upgradePassword(PasswordAuthenticatedUserInterface $user, string $newHashedPassword): void
    {
        if (!$user instanceof User) {
            throw new UnsupportedUserException(sprintf('Instances of "%s" are not supported.', \get_class($user)));
        }

        $user->setPassword($newHashedPassword);

        $this->save($user, true);
    }
}
```

La le mot de passe de l'User sera encodé en base de données. Ensuite il faudra une authentification de l'User et Symfony nous le permet avec la commande en console :

```
php bin/console make :auth
```

Modification : **config/packages/security.yaml**

Création : **UsersAuthenticator.php ; SecurityController.php ; login.html.twig**

L'User s'authentifiera avec son Email et son mot de passe.

```

config > packages > ! security.yaml
1  security:
2      # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
3      password_hashers:
4          Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
5      # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
6      providers:
7          # used to reload user from session & other features (e.g. switch_user)
8          app_user_provider:
9              entity:
10                 class: App\Entity\User
11                 property: email
12      firewalls:
13          dev:
14              pattern: ^/(_(profiler|wdt)|css|images|js)/
15              security: false
16          main:
17              lazy: true
18              provider: app_user_provider
19              custom_authenticator: App\Security>LoginAuthenticator
20              logout:
21                  path: app_logout
22                  # where to redirect after logout
23                  # target: app_any_route
24
25              # activate different ways to authenticate
26              # https://symfony.com/doc/current/security.html#the-firewall
27
28              # https://symfony.com/doc/current/security/impersonating_user.html
29              # switch_user: true
30
31      # Easy way to control access for large sections of your site
32      # Note: Only the *first* access control that matches will be used
33      access_control:
34          - { path: ^/admin, roles: ROLE_ADMIN }
35          - { path: ^/user, roles: ROLE_USER }
36

```

login.html.twig :

```

88 {% block body %}
89 <body>
90 <nav class="navigation" id="navigation">
91 <div class="container">
92 <div class="logo">Festival</div>
93 <ul class="navigation-links">
94 <li>
95 <a href="{{ path('app_home') }}">Home</a>
96 </li>
97 <li>
98 <a href="{{ path('app_etablissement_index') }}">Etablissements</a>
99 </li>
100 <li class="active"><a href="{{ path('app_login') }}">Connexion</a></li>
101 <li>
102 <a href="{{ path('app_offre_index') }}">Offres</a>
103 </li>
104 <li>
105 <a href="{{ path('app_groupes_index') }}">Groupes</a>
106 </li>
107 <li>
108 <a href="{{ path('app_type_chambre_index') }}">Types de Chambres</a>
109 </li>
110 <li>
111 <a href="{{ path('app_attributions_index') }}">Attributions</a>
112 </li>
113 {% if app.user %}
114 <li>
115 <a href="{{ path('app_logout') }}">Se déconnecter</a>
116 </li>
117 {% endif %}
118 <li>
119 <a href="javascript:void(0);" class="icon" onclick="responsiveMenu()">
120 <i class="fa fa-bars"></i>
121 </a> </li>
122 </ul>
123 </div>
124 </nav>
125 <form method="post">
126 {% if error %}
127 <div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
128 {% endif %}
129 <h1 class="h3 mb-3 font-weight-normal">Merci de vous connecter!</h1>
130 <div class="connexion">
131 <label for="inputEmail">Email</label>
132 <input type="email" value="{{ last_username }}" name="email" id="inputEmail" class="form-control" autocomplete="email" required autofocus>
133 <label for="inputPassword">Password</label>
134 <input type="password" name="password" id="inputPassword" class="form-control" autocomplete="current-password" required>
135 <input type="hidden" name="_csrf_token"
136 value="{{ csrf_token('authenticate') }}"
137 >
138 {#
139 Uncomment this section and add a remember_me option below your firewall to activate remember me functionality.
140 See https://symfony.com/doc/current/security/remember_me.html
141 <div class="checkbox mb-3">
142 <label>
143 <input type="checkbox" name="_remember_me"> Se Souvenir de moi
144 </label>
145 </div>
146 #>
147 <button class="btn btn-lg btn-primary" type="submit">
148 Se Connecter
149 </button>
150 <button class="btn btn-lg btn-primary" type="submit">
151 <a href="{{ path('app_register') }}">S'enregistrer
152 </button>
153
264 <button class="btn btn-lg btn-primary" type="submit">
265 <a href="{{ path('app_forgot_password_request') }}">
266 Changer le mot de passe
267 </button>
268 </form>
269 </div>
270 {% endblock %}

```

## Reset Password :

Il est important pour un User de pouvoir réinitialiser son mot de passe en cas d'oubli.  
La commande dans la console:

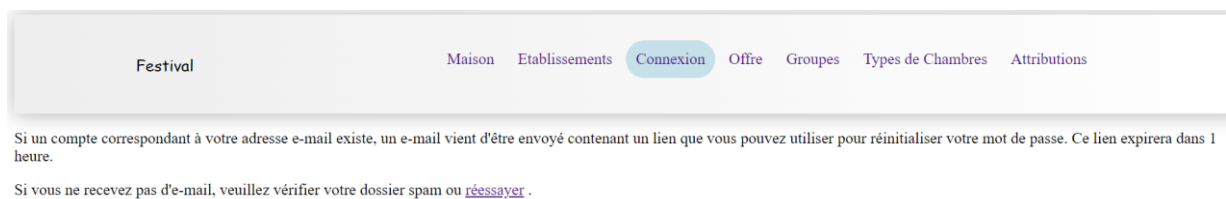
```
php bin/console make :reset-password
```

Va nous générer tous les fichiers dont nous avons besoin : le controller, l'entity, le form, le repository et toutes les vues dans le template.



```
✓ reset_password
  ✓ check_email.html.twig
  ✓ email.html.twig
  ✓ request.html.twig
  ✓ reset.html.twig
```

La vérification de l’User par son email, l’envoi du message de réinitialisation sur la boîte email avec le lien qui nous redirigera vers la page pour entrer pour changer de mot de passe.



Pour régénérer le schéma de la base de données avec tout le travail réalisé sur Symfony, nous utiliserons la commande en console :

```
Php bin/console make :migration
```

Les migrations sont des scripts qui permettent de gérer les changements du modèle de données à appliquer au SGBD. Pour cela nous suivons la recommandation de Symfony et nous exécutons la commande suivante :

```
Php bin/console doctrine :migrations :migrate
```

## CREATION DES ENTITES.

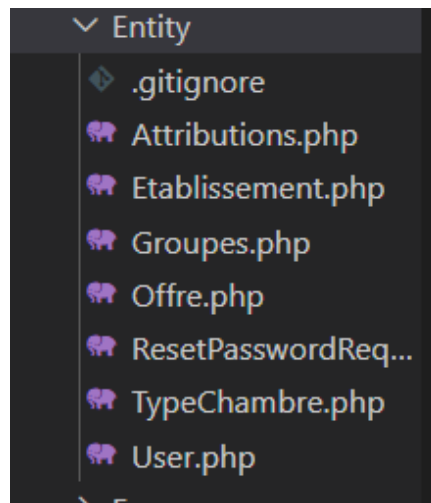
Après avoir réalisé avoir connecté la base de données, sécuriser et authentifier l’User, nous pouvons créer nos Entités. L’assistant générera le code pour le rajouter au modèle de données d’une nouvelle entité Doctrine.

### Dans la console :

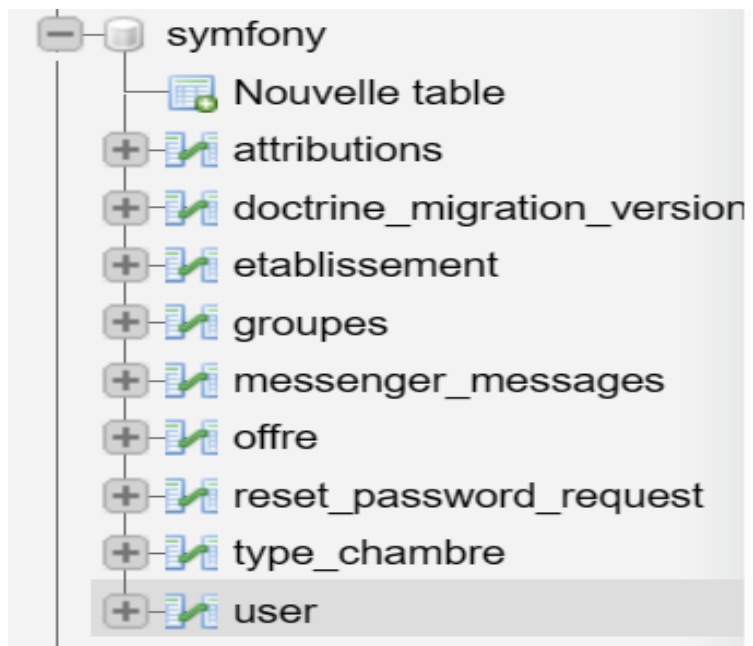
```
php bin/console make :entity
```

Entités à créer :

- **Attributions**
- **Etablissement**
- **Groupes**
- **Offres**
- **TypeChambre**
- **User**
- **ResetPassword**



### LA BASE DE DONNEES :



**La création des entités nous permettra par la suite de générer le CRUD.**

UTILISATION DU CRUD.

COMMANDE EN CONSOLE :

```
Php bin/console doctrine :generate :crud
```

Cela va générer, pour une entité du modèle de l'application, **un contrôleur, un jeu de formulaires, des templates** et méthodes associées à chacune des opérations CRUD :

- Création d'une nouvelle entité (**Create/new**)
- Consultation (**Read**), déclinée en deux sous-ensembles :
  - D'une collection d'entités (**index**)
  - D'une entité (**show**)
- Modification d'une entité (**Update/edit**)
- Suppression d'une entité (**Delete**)

```

  Controller
  ├── .gitignore
  ├── AttributionsController.php
  ├── EtablissementController.php
  ├── GroupesController.php
  ├── HomeController.php
  ├── OffreController.php
  ├── RegistrationController.php
  ├── ResetPasswordController.php
  ├── SecurityController.php
  ├── TypeChambreController.php
  └── UserController.php

```

```

  Entity
  ├── .gitignore
  ├── Attributions.php
  ├── Etablissement.php
  ├── Groupes.php
  ├── Offre.php
  ├── ResetPasswordRequest.php
  ├── TypeChambre.php
  └── User.php

  Form
  ├── AttributionsType.php
  ├── ChangePasswordFormType.php
  ├── EtablissementType.php
  ├── GroupesType.php
  ├── OffreType.php
  ├── RegistrationFormType.php
  ├── ResetPasswordRequestFormType.php
  ├── TypeChambreType.php
  └── UserType.php

  Repository
  ├── .gitignore
  ├── AttributionsRepository.php
  ├── EtablissementRepository.php
  ├── GroupesRepository.php
  ├── OffreRepository.php
  ├── ResetPasswordRequestRepository.php
  ├── TypeChambreRepository.php
  └── UserRepository.php

```

```

  templates
  ├── attributions
  │   ├── _delete_form.html.twig
  │   ├── _form.html.twig
  │   ├── edit.html.twig
  │   ├── etapeGroup.html.twig
  │   ├── index.html.twig
  │   ├── new.html.twig
  │   └── show.html.twig
  ├── base_elements
  │   ├── footer.html.twig
  │   └── nav.html.twig
  ├── etablisement
  │   ├── _delete_form.html.twig
  │   ├── _form.html.twig
  │   ├── edit.html.twig
  │   ├── index.html.twig
  │   ├── new.html.twig
  │   └── show.html.twig
  ├── groupes
  │   ├── _delete_form.html.twig
  │   ├── _form.html.twig
  │   ├── edit.html.twig
  │   ├── index.html.twig
  │   ├── new.html.twig
  │   └── show.html.twig
  ├── home
  │   ├── index.html.twig
  │   └── mail.html
  ├── offre
  │   ├── _delete_form.html.twig
  │   ├── _form.html.twig
  │   ├── edit.html.twig
  │   ├── index.html.twig
  │   ├── new.html.twig
  │   └── show.html.twig
  ├── registration
  │   ├── confirmation_email.html.twig
  │   └── register.html.twig

```

```

  reset_password
  ├── check_email.html.twig
  ├── email.html.twig
  ├── request.html.twig
  └── reset.html.twig

  security
  ├── login.html.twig

  type_chambre
  ├── _delete_form.html.twig
  ├── _form.html.twig
  ├── edit.html.twig
  ├── index.html.twig
  ├── new.html.twig
  └── show.html.twig

  user
  ├── _delete_form.html.twig
  ├── _form.html.twig
  ├── edit.html.twig
  ├── index.html.twig
  ├── new.html.twig
  ├── show.html.twig
  └── base.html.twig

```

