

## Développement d'une application pour de la gestion d'évènements

ACCUEIL

CHANSON

CHANTEUR

CLIENT

FESTIVAL

ASSOCIATIONS

INSCRIPTION FESTIVAL

INSCRIPTION RAPIDE

Festivals

Id	Nom festival	Date debut	Lieu	Genre	Nb places	Voir	Modifier
1	Francofolies	13-07-2022 19:00	La Rochelle, France	Pop	1500	<a href="#">Voir</a>	<a href="#">Modifier</a>
2	TomorrowLand	22-07-2022 20:15	Belgique	Fous Furieux	1750	<a href="#">Voir</a>	<a href="#">Modifier</a>
3	Festival Rock in Rio	18-06-2022 16:45	Lisbonne, Portugal	Rock	1386	<a href="#">Voir</a>	<a href="#">Modifier</a>
4	Rock en Seine 2022	25-08-2022 15:30	Paris, France	Rock	850	<a href="#">Voir</a>	<a href="#">Modifier</a>



Symfony

<b>DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE</b>		N° réalisation : 3
Nom, prénom : DOMAS-VASSEROT Jade		N° candidat : 02146788571
Épreuve ponctuelle <input checked="" type="checkbox"/>	Contrôle en cours de formation <input type="checkbox"/>	Date : 15 / 04 / 2022
<b>Contexte de la réalisation professionnelle</b> Une entreprise d'organisation d'événements souhaite une solution afin de préparer leur gestion de festivals. Nous avons donc créé une application web permettant de gérer les festivals, les chansons, les chanteurs et les clients avec le Framework Symfony		
<b>Intitulé de la réalisation professionnelle</b> Développement d'une application pour de la gestion d'événements		
Période de réalisation : 2eme année..... Lieu : Lyon ..... Modalité : <input type="checkbox"/> Seul(e) <input checked="" type="checkbox"/> En équipe		
<b>Compétences travaillées</b> <input checked="" type="checkbox"/> Concevoir et développer une solution applicative <input type="checkbox"/> Assurer la maintenance corrective ou évolutive d'une solution applicative <input checked="" type="checkbox"/> Gérer les données		
<b>Conditions de réalisation<sup>1</sup> (ressources fournies, résultats attendus)</b> Ressources fournies : - Sujet de projet, Cours de Symfony, accès à la documentation internet Résultats attendus : - Conception d'une base de données - Développement d'une application Symfony en MVC (CRUD)		
<b>Description des ressources documentaires, matérielles et logicielles utilisées<sup>2</sup></b> <b>Logiciels :</b> VS code WampServer (PhpMyAdmin, Apache) Accès internet Navigateur Chrome Installation de Composer <b>Langages :</b> Twig pour la vue SQL PHP avec le Framework Symfony		
<b>Modalités d'accès aux productions<sup>3</sup> et à leur documentation<sup>4</sup></b> GitHub: <a href="https://github.com/JadeDomasVasserot/festival">https://github.com/JadeDomasVasserot/festival</a>		

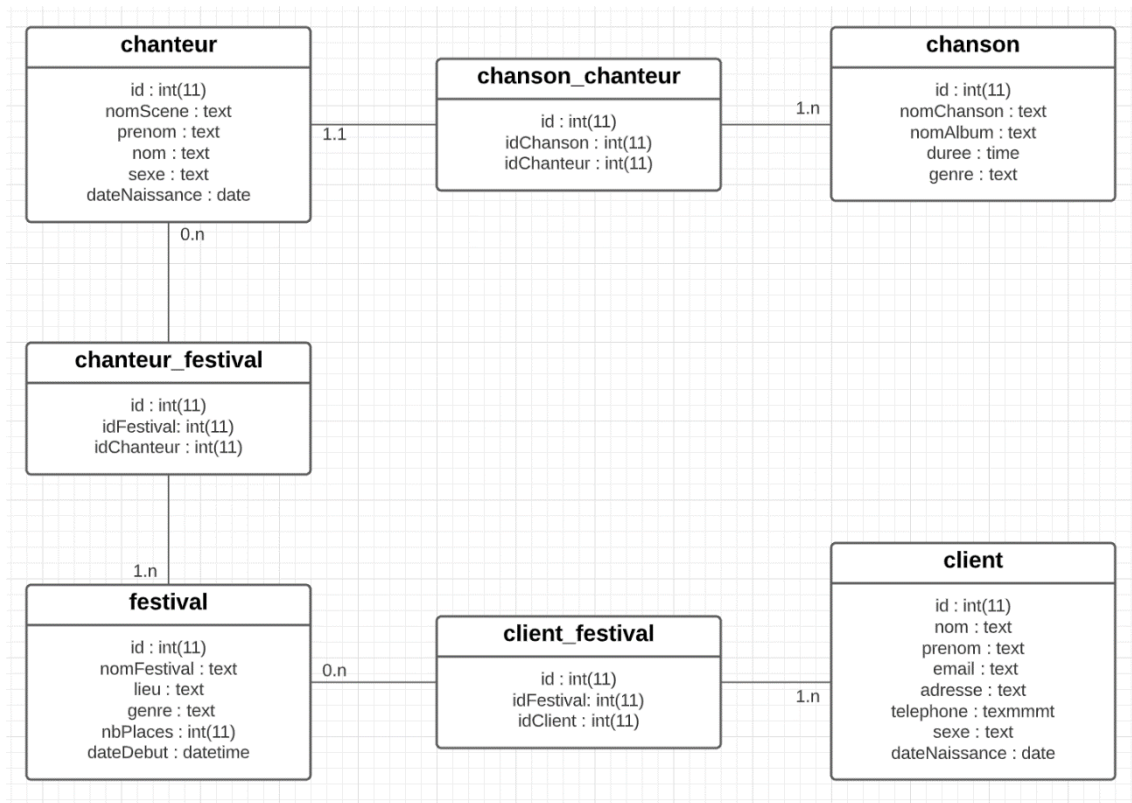
<sup>1</sup> En référence aux conditions de réalisation et ressources nécessaires du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO.

<sup>2</sup> Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

<sup>3</sup> Conformément au référentiel du BTS SIO « Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve. ». Les éléments peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.

<sup>4</sup> Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation professionnelle, par exemple service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données.

## Schéma de BDD :



## Description de l'activité :

- Conception d'une BDD
- Développement d'une application Symfony

```

    src > Controller > ChansonChanteurController.php
    1 <?php
    2
    3 namespace App\Controller;
    4
    5 use App\Entity\ChansonChanteur;
    6 use App\Form\ChansonChanteurType;
    7 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
    8 use Symfony\Component\HttpFoundation\Request;
    9 use Symfony\Component\HttpFoundation\Response;
    10 use Symfony\Component\Routing\Annotation\Route;
    11
    12 /**
    13  * @Route("/chanson/chanteur")
    14  */
    15 class ChansonChanteurController extends AbstractController
    16 {
    17     /**
    18      * @Route("/", name="chanson_chanteur_index", methods={"GET"})
    19      */
    20     public function index(): Response
    21     {
    22         $chansonChanteurs = $this->getDoctrine()
    23             ->getRepository(ChansonChanteur::class)
    24             ->findAll();
    25
    26         return $this->render('chanson_chanteur/index.html.twig', [
    27             'chanson_chanteurs' => $chansonChanteurs,
    28         ]);
    29     }
    30
    31     /**
    32      * @Route("/new", name="chanson_chanteur_new", methods={"GET", "POST"})
    33      */
    34     public function new(Request $request): Response
    35     {
    36         $chansonChanteur = new ChansonChanteur();
    37         $form = $this->createForm(ChansonChanteurType::class, $chansonChanteur);
    38         $form->handleRequest($request);
    39     }
    40 }
    
```

## Contexte :

Avec l'arrivée de l'été et la digitalisation de plus en plus commune des applications, une association d'organisations d'événements a voulu digitaliser son processus de gestion de festivals.

Il nous demande le fait de pouvoir ajouter/modifier/supprimer :

- Un festival
  - Un chanteur
  - Un client
  - Une chanson
- Et de lier chaque information.

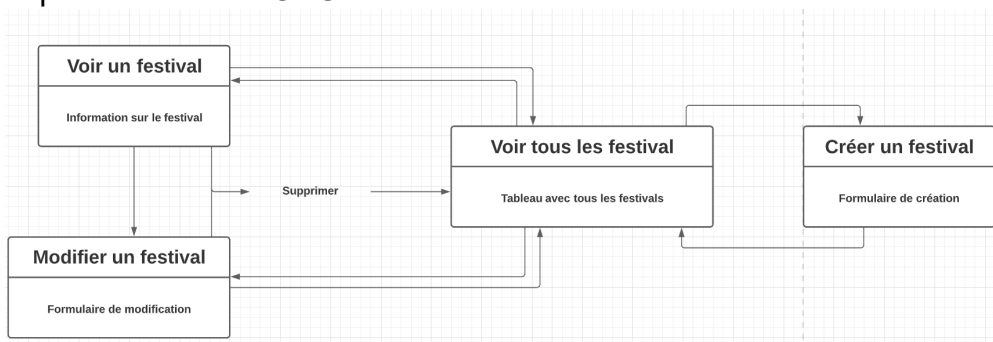
## Réalisation :

Pour cela, il nous a fallu dans un premier temps concevoir le processus de l'application afin de comprendre les besoins de l'association.

L'association avait besoin de gérer les clients, chansons, festivals, chanteurs.

On a donc conçu la base de données (voir plus haut). Puis, nous avons décidé d'utiliser la méthode CRUD (Create Read Update Delete) pour gérer les données.

Exemple de process avec le CRUD :



Pour chaque entité dans notre table (voir BDD).

Nous avons par la suite créé un projet Symfony.

composer create-project symfony/website-skeleton

Il a fallu connecter notre base de données dans .env qui a été créée au préalable.

La base de donnée a été créer avant.

```
.env
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
#
# DATABASE_URL="sqlite://%kernel.project_dir%/var/data.db"
27 DATABASE_URL="mysql://root:@127.0.0.1:3306/festival?serverVersion=5.7"
28
29 #DATABASE_URL="postgresql://db_user:db_password@127.0.0.1:5432/db_name?serverVersion=13&charset=utf8"
30
31 ###< doctrine/doctrine-bundle ###
32
33
```

Pour créer les entités à partir de la BDD puis les getters/setters puis les contrôleurs.

Commande utilisée :

**php bin/console doctrine:mapping:import "App\Entity" annotation --path=src/Entity**

**php bin/console make:entity --regenerate App**

**php bin/console make:controller**

## Exemple d'entité générée :

```
1 <?php
2 namespace App\Entity;
3
4 use Doctrine\ORM\Mapping as ORM;
5
6 /**
7  * Festival
8  *
9  * @ORM\Table(name="festival")
10  * @ORM\Entity
11  */
12 class Festival
13 {
14     /**
15      * @var int
16      *
17      * @ORM\Column(name="id", type="integer", nullable=false)
18      * @ORM\Id
19      * @ORM\GeneratedValue(strategy="IDENTITY")
20      */
21     private $id;
22
23     /**
24      * @var string
25      *
26      * @ORM\Column(name="nomFestival", type="text", length=65535, nullable=false)
27      */
28     private $nomFestival;
29
30     /**
31      * @var \DateTime
32      *
33      * @ORM\Column(name="dateDebut", type="datetime", nullable=false)
34      */
35     private $dateDebut;
36
37     /**
38      * @var string
39      *
40      * @ORM\Column(name="lieu", type="text", length=65535, nullable=false)
41      */
42     private $lieu;
43
44     /**
45      * @var string
46      *
47      * @ORM\Column(name="genre", type="text", length=65535, nullable=false)
48      */
49     private $genre;
50
51     /**
52      * @var int
53      *
54      * @ORM\Column(name="nbPlaces", type="integer", nullable=false)
55      */
56     private $nbPlaces;
57
58     public function getId(): ?int //Retourne l'id
59     {
60         return $this->id;
61     }
62     public function getNomFestival(): ?string //Retourne le nom du festival
63     {
64         return $this->nomFestival;
65     }
66     public function setNomFestival(string $nomFestival): self //Change le nom du festival
67     {
68         $this->nomFestival = $nomFestival;
69         return $this;
70     }
71     public function getDateDebut(): ?\DateTimeInterface //Retourne la date de début
72     {
73         return $this->dateDebut;
74     }
75     public function setDateDebut(\DateTimeInterface $dateDebut): self //Change la date de début
76     {
77         $this->dateDebut = $dateDebut;
78         return $this;
79     }
80     public function getLieu(): ?string //Retourne le lieu
81     {
82         return $this->lieu;
83     }
84     public function setLieu(string $lieu): self //Change le lieu
85     {
86         $this->lieu = $lieu;
87         return $this;
88     }
89     public function getGenre(): ?string //Retourne le genre
90     {
91         return $this->genre;
92     }
93     public function setGenre(string $genre): self //Change le genre
94     {
95         $this->genre = $genre;
96         return $this;
97     }
98     public function getNbPlaces(): ?int //Retourne le nombre de places
99     {
100         return $this->nbPlaces;
101     }
102     public function setNbPlaces(int $nbPlaces): self //Change le nombre de place
103     {
104         $this->nbPlaces = $nbPlaces;
105         return $this;
106     }
107     public function __toString()
108     {
109         return $this->id.'-'.$this->nomFestival;
110     }
111 }
```

Après la création des entités nous avons fait les contrôleurs avec la méthode du CRUD.  
Commande utilisée :

**php bin/console make:crud**

## Exemple de contrôleur :

```
1 <?php
2 namespace App\Controller;
3 use App\Entity\Festival;
4 use App\Form\FestivalType;
5 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
6 use Symfony\Component\HttpFoundation\Request;
7 use Symfony\Component\HttpFoundation\Response;
8 use Symfony\Component\Routing\Annotation\Route;
9 /**
10  * @Route("/festival")
11  */
12 class FestivalController extends AbstractController
13 {
14     /**
15      * @Route("/", name="festival_index", methods={"GET"})
16      */
17     public function index(): Response
18     {
19         $festivals = $this->getDoctrine()
20             ->getRepository(Festival::class)
21             ->findAll();
22
23         return $this->render('festival/index.html.twig', [
24             'festivals' => $festivals,
25         ]);
26     }
27
28     /**
29      * @Route("/new", name="festival_new", methods={"GET","POST"})
30      */
31     public function new(Request $request): Response
32     {
33         $festival = new Festival();
34         $form = $this->createForm(FestivalType::class, $festival);
35         $form->handleRequest($request);
36
37         if ($form->isSubmitted() && $form->isValid()) {
38             $entityManager = $this->getDoctrine()->getManager();
39             $entityManager->persist($festival);
40             $entityManager->flush();
41
42             return $this->redirectToRoute('festival_index', [], Response::HTTP_SEE_OTHER);
43         }
44
45         return $this->renderForm('festival/new.html.twig', [
46             'festival' => $festival,
47             'form' => $form,
48         ]);
49     }
50
51     /**
52      * @Route("/{id}/edit", name="festival_edit", methods={"GET","POST"})
53      */
54     public function edit(Request $request, Festival $festival): Response
55     {
56         $form = $this->createForm(FestivalType::class, $festival);
57         $form->handleRequest($request);
58
59         if ($form->isSubmitted() && $form->isValid()) {
60             $this->getDoctrine()->getManager()->flush();
61
62             return $this->redirectToRoute('festival_index', [], Response::HTTP_SEE_OTHER);
63         }
64
65         return $this->renderForm('festival/edit.html.twig', [
66             'festival' => $festival,
67             'form' => $form,
68         ]);
69     }
70
71     /**
72      * @Route("/{id}", name="festival_delete", methods={"POST"})
73      */
74     public function delete(Request $request, Festival $festival): Response
75     {
76         if ($this->isCsrfTokenValid('delete', $festival->getId(), $request->request->get('_token'))) {
77             $entityManager = $this->getDoctrine()->getManager();
78             $entityManager->remove($festival);
79             $entityManager->flush();
80
81             return $this->redirectToRoute('festival_index', [], Response::HTTP_SEE_OTHER);
82         }
83     }
84 }
```

Nous avons donc pu partir de cette base afin de faire les fonctionnalités voulues en modifiant les contrôleurs, routes, entités.  
Comme la page d'index où l'on affiche tous les festivals.  
En parallèle, nous avons fait les pages twig pour le visuel avec de l'HTML CSS.

Page d'index :

Festivals							
Id	Nom festival	Date debut	Lieu	Genre	Nb places	Voir	Modifier
1	Francofolies	13-07-2022 19:00	La Rochelle, France	Pop	1500	<a href="#">Voir</a>	<a href="#">Modifier</a>
2	TomorrowLand	22-07-2022 20:15	Belgique	Fous Furieux	1750	<a href="#">Voir</a>	<a href="#">Modifier</a>
3	Festival Rock in Rio	18-06-2022 16:45	Lisbonne, Portugal	Rock	1386	<a href="#">Voir</a>	<a href="#">Modifier</a>
4	Rock en Seine 2022	25-08-2022 15:30	Paris, France	Rock	850	<a href="#">Voir</a>	<a href="#">Modifier</a>

Présentation du projet :

[Téléchargement et aperçu de notre BDD](#)

Table	Action	Champs	Type	Interclassement	Taille	Partie
chanson	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	18	IntnoDB	utf8mb4_unicode_ci	16,0 Kio	-
chanson_chanteur	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	18	IntnoDB	utf8mb4_unicode_ci	48,0 Kio	-
chanteur	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	9	IntnoDB	utf8mb4_unicode_ci	16,0 Kio	-
chanteur_festival	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	2	IntnoDB	utf8mb4_unicode_ci	16,0 Kio	-
client	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	6	IntnoDB	utf8mb4_unicode_ci	16,0 Kio	-
client_festival	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	2	IntnoDB	utf8mb4_unicode_ci	16,0 Kio	-
festival	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	7	IntnoDB	utf8mb4_unicode_ci	16,0 Kio	-
7 tables	Somme	14	MySQL	utf8mb4_unicode_ci	208,0 Kio	100%

Page index d'un contrôleur :

Index de tous les festivals							
Id	Nomfestival	Datedebut	Lieu	Genre	Nbplaces	Voir	Modifier
1	Francofolies	13-07-2022 19:00	La Rochelle, France	Pop	1500	<a href="#">Voir</a>	<a href="#">Modifier</a>
2	TomorrowLand	22-07-2022 20:15	Belgique	Fous Furieux	1750	<a href="#">Voir</a>	<a href="#">Modifier</a>
3	Festival Rock in Rio	18-06-2022 16:45	Lisbonne, Portugal	Rock	1386	<a href="#">Voir</a>	<a href="#">Modifier</a>
4	Rock en Seine 2022	25-08-2022 15:30	Paris, France	Rock	850	<a href="#">Voir</a>	<a href="#">Modifier</a>

[Créer un festival](#)

Page de modification :

Editer votre festival

Nom Du Festival

Francfofolies

Date Et Heure

Jul

13

2022

19

00

Lieu

La Rochelle, France

Genre

Pop

Nombre De Places

1500

Enregistrer

Supprimer

< Retour à la liste

Page de création :

Créer un nouveau Festival

Nom Du Festival

Date Et Heure

Jan

1

2017

00

00

Lieu

Genre

Nombre De Places

Sauvegarder

< Retour à la liste



Page pour voir une entité :

[ACCUEIL](#) [CHANSON](#) [CHANTEUR](#) [CLIENT](#) [FESTIVAL](#) [ASSOCIATIONS](#) [INSCRIPTION FESTIVAL](#) [INSCRIPTION RAPIDE](#)

### Festival : Francofolies

<b>Id</b>	1
<b>Nomfestival</b>	Francofolies
<b>Datedebut</b>	13-07-2022 19:00
<b>Lieu</b>	La Rochelle, France
<b>Genre</b>	Pop
<b>Nbplaces</b>	1500

[Retour à l'index](#)[Editer](#)[Supprimer](#)