

PROJET TECHTONITE

PROJET TECHTONITE

Une application pour qui et pourquoi ?

- A qui on s'adresse, but du projet
- Déroulement en 3 concerts par jours
- Connexion sécurisée
- Permet de réserver des places



PROJET TECHTONITE

- Un développement au pas de course avec plusieurs sprints !
- Une équipe performante soucieuse du consommateur du service !
- Un socle technique réputé et sécurisé fonctionnant sur PC, tablette et smartphone !
- Des fonctionnalités simples d'utilisation :
 - ▷ Liste artistes → détail de l'artiste
 - ▷ Agenda des concerts -> réservation
 - ▷ Impression ou snapshot -> à présenter au concert



PROJET TECHTONITE

MCD

Uses Cases


Uses Stories



MCD

Modèle conceptuel des données




artiste		
 id		AUTO_INCREMENT
name		VARCHAR (50)
slug		VARCHAR (50)
description		TEXT (255)
concert		INT

0,1

inclure

0,n

category		
 idCategory		AUTO_INCREMENT
label		VARCHAR (50)
slug		VARCHAR (50)

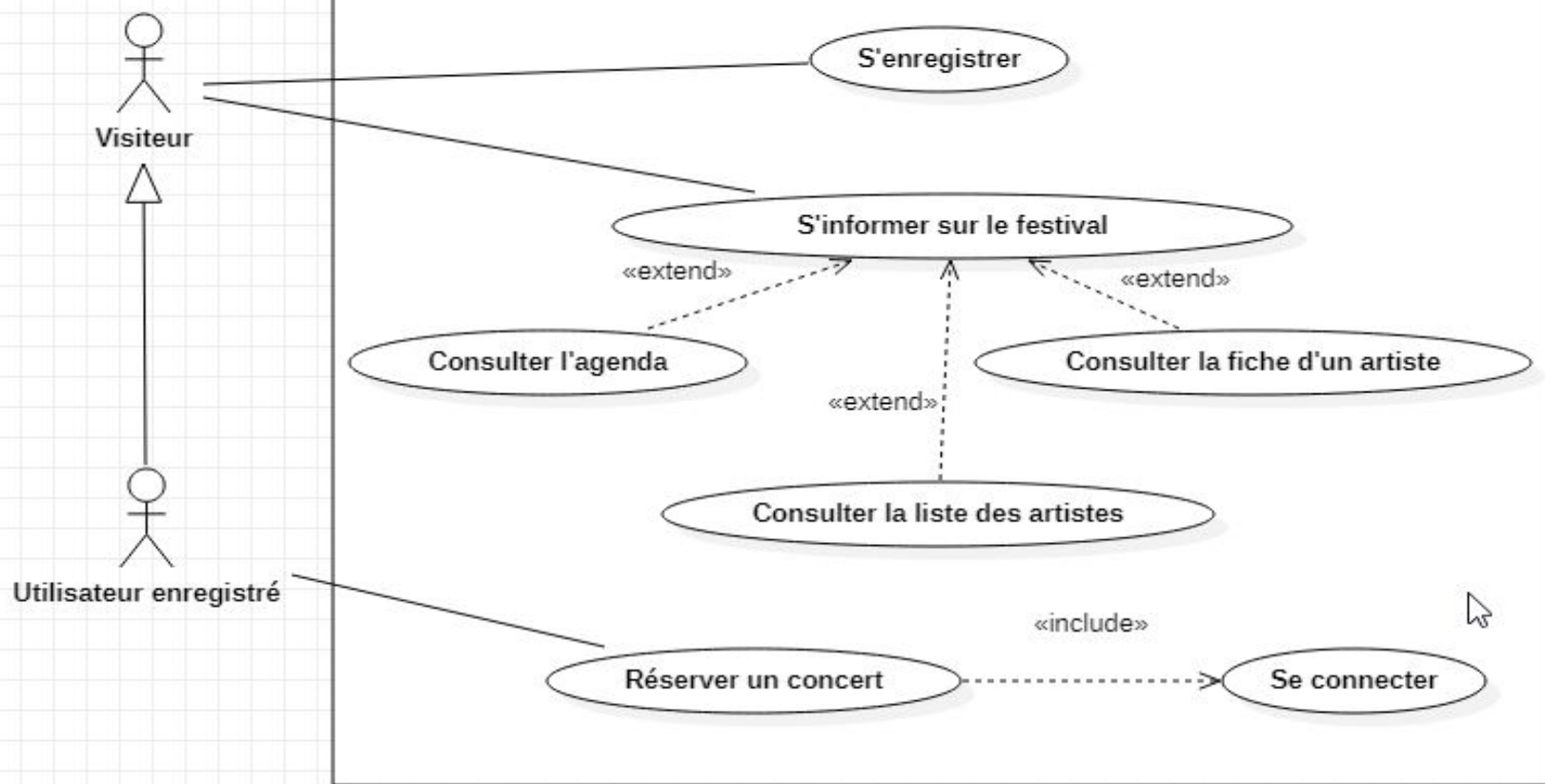
utilisateur		
 idUtilisateur		AUTO_INCREMENT
nomUtilisateur		VARCHAR (50)
prenomUtilisateur		VARCHAR (50)
numeroTelephone		NUMERIC
emailUtilisateur		VARCHAR (50)

USES CASES

Diagramme UML



Application web Festival Technonite



USES STORIES



US-1- Vue Accueil :

- En tant que visiteur je peux **avoir une présentation du festival.(Lien Accueil)**

- En tant que visiteur je peux **m'enregistrer via la page d'accueil :**

Description

Cliquer sur un lien Enregistrement qui m'envoie vers la page d'enregistrement via la barre de navigation

- En tant que visiteur je peux **me renseigner sur les dates des concerts et les artistes qui y participeront.**

Description

Cliquer sur un lien Agenda qui m'envoie vers une page qui contient toutes les informations nécessaires.

- En tant que visiteur je peux **consulter la liste des artistes présents pendant le festival.**

Description

Clique sur un lien Artistes qui m'envoie vers une page qui contient la liste de tous les artistes

- En tant qu'utilisateur je peux me **connecter via la page d'accueil**

Description

Cliquer sur un lien Connexion qui m'envoie vers la page de connexion via la barre de navigation

US-2- Vue Connexion:

- En tant qu'utilisateur je peux **me connecter via la page de connexion**

Description

Créer un formulaire de connexion avec les champs : Email et Mot de passe

Créer un bouton se connecter



US-3- Vue Enregistrement:

- En tant qu'utilisateur je peux **m'inscrire via la page d'enregistrement**

Description

Créer un formulaire d'enregistrement avec les champs : Email et Mot de passe

Créer un bouton s'enregistrer



US-4- Vue Artistes :

- En tant qu'utilisateur je peux **afficher la liste de tous les artistes**
- En tant qu'utilisateur je peux **afficher la liste des artistes par catégorie**

Description

Créer un bouton pour chaque catégorie de musique

En cliquant sur le bouton, seulement la liste des artistes de la catégorie sélectionnée sera affichée

- En tant qu'utilisateur je peux **avoir plus de détails sur un artiste.**

Description

Créer un lien qui envoie vers une nouvelle page contenant les détails d'un artiste



US-5- Vue Agenda :

- En tant que visiteur/utilisateur **je peux consulter les dates, les horaires des concerts + Artistes participants**

Description

Créer un tableau qui contient les dates, les horaires et les artistes.



- En tant qu'utilisateur **je peux aller vers la page de réservation pour réserver le concert sélectionné :**

Description

Créer un bouton réservation pour chaque concert

Cliquer sur le lien pour aller vers la page de réservation



US-6- Vue Billetterie:

- En tant qu'utilisateur je peux **réserver une ou plusieurs places, une ou plusieurs concerts**

Description

Créer un input pour chaque ligne du tableau qui contient les concerts

- En tant qu'utilisateur **je peux réserver via un formulaire**

Description

Créer un formulaire qui contient les champs : nom prénom et numéro de téléphone

Créer un bouton de validation

- En tant qu'utilisateur **je peux imprimer ma réservation**

Description

Créer un bouton imprimer qui permet d'imprimer la réservation du client

Choix du frameworks

Pourquoi Symfony ?

1. Sécurité de l'application
2. Facilité d'utilisation
3. Architecture MVC Flexible
4. Gestion automatisée de la base des données

DÉMONSTRATION DU PROJET

DEMONSTRATION DU PROJET



Création De

```
class CategoryFixtures extends Fixture
{
    private array $categories = [
        "Mélodique",
        "Industrielle",
        "Groovy",
        "Deep",
        "Détroit"
    ];
}
```

Données Fictives

```
public function load(ObjectManager $manager)
{
    $faker = Factory::create('fr_FR');
    $concert = 1;

    foreach($this->categories as $name) {
        $category = new Category();
        $category->setLabel($name);

        $manager->persist($category);

        for($i = 1; $i < rand(3,15); $i++) {
            $artist = new Artist();
            $artist->setName('DJ ' . $faker->firstName());
            $artist->setDescription($faker->paragraphs(3, true));
            $artist->setCategory($category);

            if($concert <= 9 && rand(0,8) <= 2) {
                $artist->setConcert($concert);
                $concert++;
            }

            $manager->persist($artist);
        }
    }

    $manager->flush();
}
```

Optim...

```
// Service/CategoryHandler.php
```

```
class CategoryHandler
```

```
{
```

```
    public function setColors(array $categories): array
```

```
{
```

```
    $colors = [
```

```
        "Mélodique" => "danger",
```

```
        "Industrielle" => "info",
```

```
        "Groovy" => "secondary",
```

```
        "Deep" => "warning",
```

```
        "Détroit" => "success"
```

```
    ];
```

```
    foreach($categories as $category) {
```

```
        $category->setColor($colors[$category->getLabel()]);
```

```
    }
```

```
    return $categories;
```

```
// Controller/ArtistController.php
```

```
class ArtistController extends AbstractController
```

```
{
```

```
    private array $categories;
```

```
    private ArtistRepository $artistRepository;
```

```
    public function __construct(ArtistRepository $artistRepository, CategoryRepository  
                                $categoryRepository, CategoryHandler $categoryHandler)
```

```
{
```

```
        $this->categories = $categoryHandler->setColors($categoryRepository->findAll());
```

```
        $this->artistRepository = $artistRepository;
```

```
}
```

```
// Controller/ArtistController.php
```

```
public function index(): Response
```

```
{
```

```
    return $this->render('artist/index.html.twig', [
```

```
        'categories' => $this->categories,
```

```
        'artists' => $this->artistRepository->findAll()
```

```
    ]);
```

```
}
```

...iZation

```
// artist/index.html.twig
<a href="{{ path('artist_category', {id: category.id}) }}">
    {{ category.label }}
</a>
```

```
// ArtistController.php
#[Route('/category/{id<\d+>}', name: 'category')]
public function category(int $id): Response
{
    return $this->render('artist/index.html.twig', [
        'artists' => $this->artistRepository->findBy(['category' => $id]),
        'categories' => $this->categories
    ]);
}
```

```
// ArtistRepository.php
public function findByConcert()
{
    return $this->createQueryBuilder('a')
        ->select('a.id', 'a.concert', 'a.name')
        ->where('a.concert IS NOT NULL')
        ->orderBy('a.id', 'ASC')
        ->getQuery()
        ->getResult();
}
```

CUSTOM QUERY

Récapitulatif

```
// templates/techno/ticket.html.twig
```

```
// Récapitulatif des réservations effectuées
```

```
{% if app.request.get('firstname') %}
```

```
  <div class="alert alert-dismissible alert-dark" style="margin-bottom: 1em">
```

```
    <div class="fs-2 text-light mb-2">
```

```
      <strong>Bravo !</strong> Vous êtes maintenant inscrit au(x) concert(s) suivant(s):
```

```
    </div>
```

```
  <div style="margin-left: 2rem">
```

```
    {% for num in 0..9 %}
```

```
      {% if app.request.get('concert_' ~ num) %}
```

```
        <div class='text-info'>
```

```
          Concert {{ num + 1 }} - {{ artists[num].name }} - Nombre de place(s) réservée(s):
```

```
          <strong class="text-white">{{ app.request.get('concert_' ~ num) }}</strong>
```

```
        </div>
```

```
      {% endif %}
```

```
    {% endfor %}
```

```
  </div>
```

```
  <div class="w-100 text-end">
```

```
    <input type="button" class="btn btn-sm btn-outline-success" value="Imprimer" onclick="window.print()" />
```

```
  </div>
```

```
</div>
```

```
{% endif %}
```

MERCI !

WEBCI i

Nous sommes à
votre disposition !

