

## **Projet : Jeux de concours**

**Réalisé par : Abdessamad Bouzekri**

**12012248**

**Outlies: PHP - Symfony –Twig - Bootstrap – xampp – postman – Visual Studio Code – GitHub**

**Lien github: <https://github.com/bouzekri1/Jeux-de-concours>**

## 1- Présentation du projet :

L'application des jeux de concours est une application qui permet les utilisateurs de participer dans des jeux de concours organisée par des compagnes et de gagner des lots.

Les compagnes doivent créer des concours, ajouter des lots et réaliser le tirage au sort afin de faire gagner un participant dans un concours.

Enfin l'utilisateur peut voir les concours dont il a gagné.

## 2- Point de vue admin (Organisateur) :

- Tout d'abord j'ai créé la page de login à l'aide de la commande

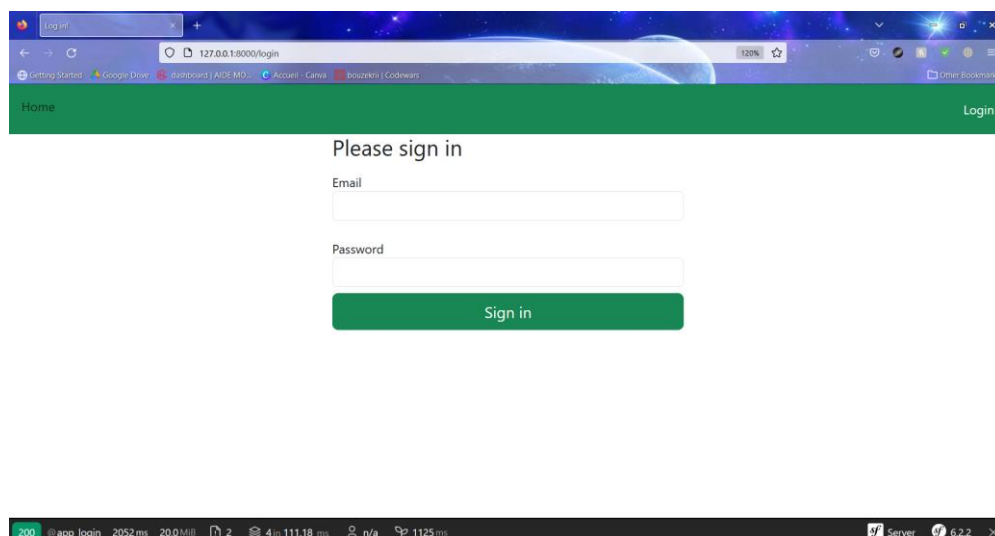
```
Stream the logs via symfony.exe server.log
PS C:\Users\ing\Desktop\Jeux-de-concours-master> php bin/console make:auth

What style of authentication do you want? [Empty authenticator]:
[0] Empty authenticator
[1] Login form authenticator
> 1
```

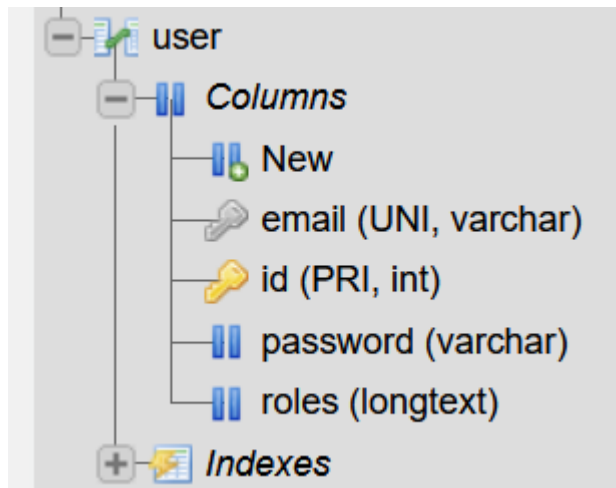
Ensuite symfony crée le répertoire Security qui contient la classe *AppCustomAuthenticator* responsable du process de l'authentification et la Template Security dont on trouve le fichier *login.html.twig* qui permet de réaliser la page d'authentification, pour cela j'ai ajouté aussi Bootstrap pour dans le fichier *Base.html.twig*

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
      rel="stylesheet" integrity="sha384-GlhlTQ8iRABdZL1603oVMwSkktQOp6b7In1Z13/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
{# Run 'composer require symfony/webpack-encore-bundle' to start using Symfony UX #}
```

pour avoir la page suivante :



- Ensuite j'ai créé l'entité user avec les attributs suivants :

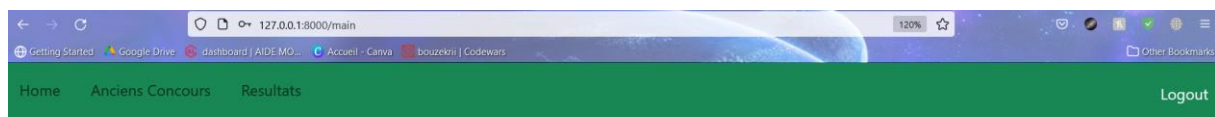


J'ai ajouté ensuite dans cette table un utilisateur admin avec un ROLE\_ADMIN

```
7 admin@admin.com ["ROLE_ADMIN"] $2y$13$dZcfs9Kd.feWFNhY1bp6eIG7lowSZ4figsra2iASM9...
```

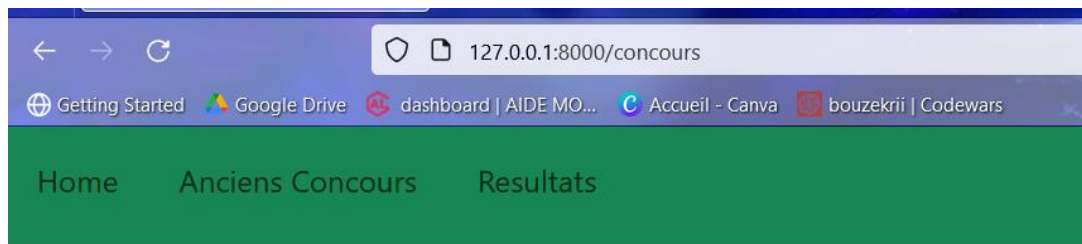
Son mot de passe est : 123456789

Dès qu'il se connecte il se dirige vers le contrôleur **main** où il trouve la liste des concours ajouté :



+ Tous les Concours				
Concours				
Id	Name	Description	rewards	Opertaions
2	gagner TV	tv	178	<button>Tirer Gagnant</button> <button>Participer</button>
3	gagner PC	HP	1458	<button>Tirer Gagnant</button> <button>Participer</button>
9	Noel	gagner un voyage	10.12	<button>Tirer Gagnant</button> <button>Participer</button>

Il peut ensuite à l'aide du bouton (+) qui lui fait diriger vers le controlleur **concours** et lui permet d'ajouter un nouveau concours



### Ajouter Nouveau Concours

Name

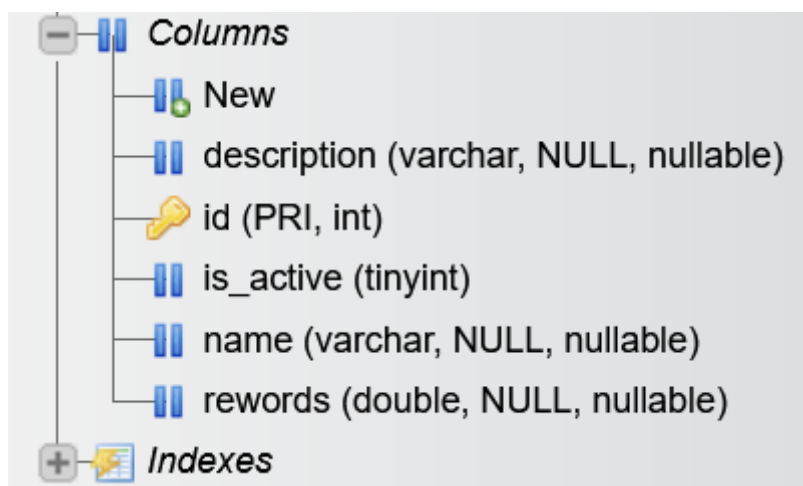
Description

Rewords

Register

Le champs Name est pour le nom du concours, Description lui permet d'ajouter de saisir les lots à gagner et Rewords pour le nombre de lots.

Pour créer cette page j'ai créé une entité **concours** avec les attributs suivants :



La colonne **is\_active** est pour spécifier si un concours est toujours disponible (il sera désactivé dès que l'admin réalise le tirage au sort).

Ensuite j'ai créé le formulaire de l'ajout du concours avec la commande Symfony :

```
PS C:\Users\ing\Desktop\Jeux-de-concours-master> php bin/console make:form

The name of the form class (e.g. AgreeablePopsicleType):
```

Dès que l'organisateur ajout le nouveau concours il sera ajouter dans la liste des concours :

11 Concours de Samsung

Galaxy J1

2

Tirer Gagnant

Participer

Ensuite l'admin peut réaliser le tirage du gagnant grâce au bouton **Tirer Gagnant**

Dès qui clique dessus il affiche les participants dans ce concours à l'aide de la fonction **index** et je récupère l'id du concours à l'aide de la fonction

```
public function index(Request $request, EntityManagerInterface $entityManager): Response
{
    $id_concr = $_GET['id_concr'];

    $repository = $entityManager->getRepository(Participants::class);
    $particippant= $repository->findBy(['id_concours' => $id_concr]);
    shuffle($particippant);
    // dd($particippant);

    return $this->render('tirage_gagnant/index.html.twig', [
        'controller_name' => 'TirageGagnantController', 'participants'=>$particippant,'id_concr'=>$id_concr,'errorMessage' => null
    ]);
}
```

lancer le tirage Tous les Participants

Participants

Id	Name	Lastname	phoneNumber
5	ffff	fff	ffff
7	ffff	fff	ffff
6	ffff	fff	ffff
3	ffff	fff	ffff
9	sss	sokdd	78798809
4	ffff	fff	ffff
12	user1	user1	064587512

Ensuite l'organisateur peut lancer le tirage au sort à l'aide du bouton lancer le tirage un onglet s'affiche ensuite pour spécifier le nombre de gagnants

# Tirage



Nombre ds gagnant



Annuler

lancer

Et lorsqu'il lance une fonction de shuffle s'exécute dans la fonction **result**

```
shuffle($particippant);  
  
$participants_gagnants = array();  
for ($i = 0; $i < $nb_gagnant;$i++)  
{  
    array_push($participants_gagnants, $particippant[$i]);  
    $particippant[$i]->setIsGagnant(true);  
    $entityManager->persist($particippant[$i]);  
    $entityManager->flush();  
}
```

Enfin l'organisateur aura les 3 gagnants qui ont gagné



les Participants gagnants

Concours

Id	Name	Lastname	phoneNumber
12	user1	user1	064587512
6	ffff	fff	ffff
4	ffff	fff	ffff

Et le concours s'ajoutera dans la liste des concours désactivés (Anciens Conours) :

Id	Name	Description	rewards
1	tombola	gagner TV	14.5
2	gagner TV	tv	178
6	Cheque	cheque	15
7	Noel	gagner un voyage	10.12
8	Noel	gagner un voyage	10.12
10	Concours Apple	Gagner AirPods3	2

NB : L'organisateur ne peut pas réaliser un tirage au sort si le nombre de gagnant est supérieur à celui des participants

Exemple :

lancer le tirage

Tous les Participants

Participants

Id	Name	Lastname	phoneNumber

Tirage

x

Nombre ds gagnant

1

Annuler

lancer

lancer le tirage

Tous les Participants

Participants

le nombre des gagnants est superieur aux participants

Id	Name	Lastname	phoneNumber
----	------	----------	-------------

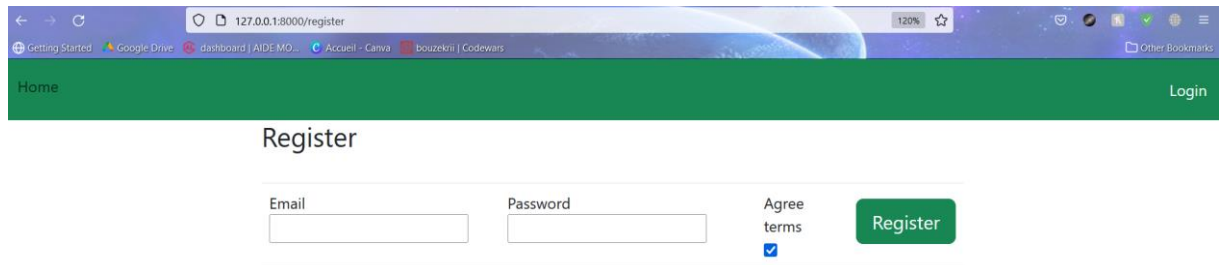
Grâce à l’instruction :

```
if($nb_gagnant>count($particippant))
{
    return $this->render('tirage_gagnant/index.html.twig', [
        'id_concr'=>$id_concr,
        'participants'=>$particippant,
        'errorMessage' => 'le nombre des gagnants est superieur aux participants'
    ]);
}
```



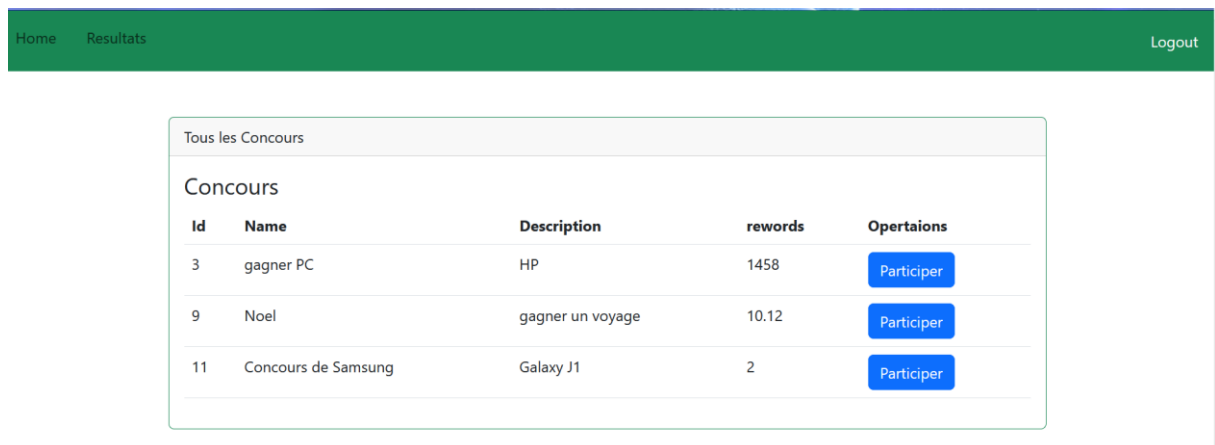
Point de vue utilisateur :

Un utilisateur peut créer un compte à l'aide du formulaire de l'enregistrement suivant :



The screenshot shows a web browser at the URL 127.0.0.1:8000/register. The page has a green header with 'Home' and 'Login' links. The main content area is titled 'Register' and contains a form with three input fields: 'Email', 'Password', and 'Agree terms' (with a checked checkbox). A green 'Register' button is positioned to the right of the 'Agree terms' checkbox.

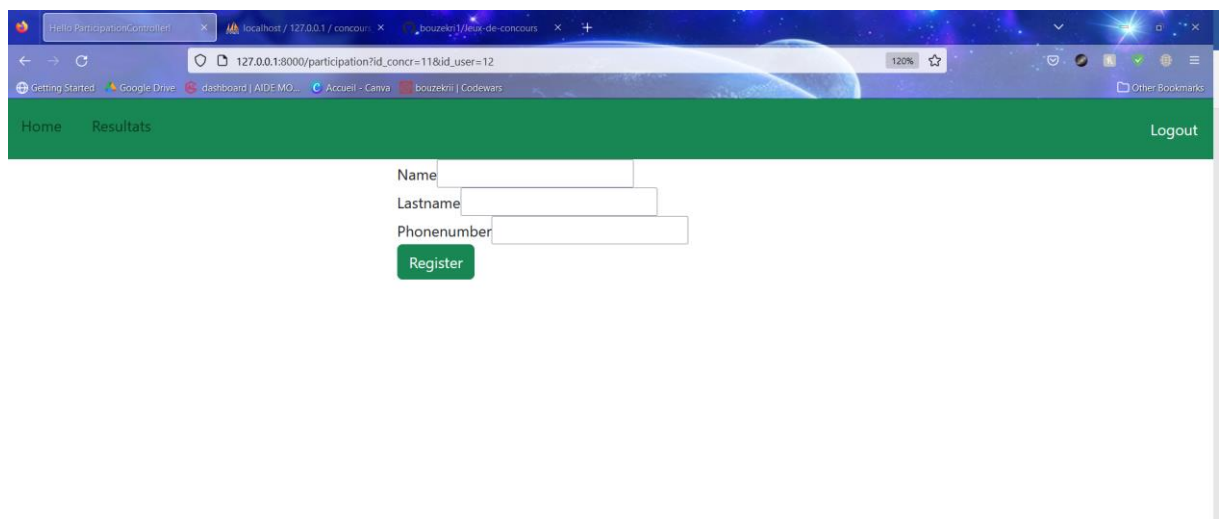
Dès qu'il sera enregistré il verra la liste des concours disponibles :



The screenshot shows a user dashboard with a green header containing 'Home', 'Resultats', and 'Logout' links. Below the header, there is a section titled 'Tous les Concours' which contains a table of competitions.

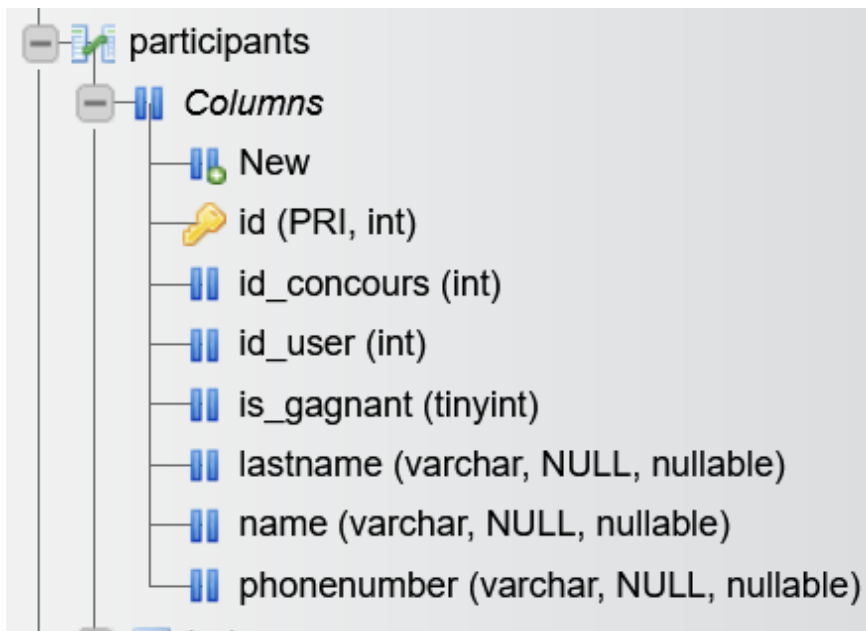
Concours				
Id	Name	Description	rewards	Opertaions
3	gagner PC	HP	1458	<a href="#">Participer</a>
9	Noel	gagner un voyage	10.12	<a href="#">Participer</a>
11	Concours de Samsung	Galaxy J1	2	<a href="#">Participer</a>

Pour participer dans un concours il doit cliquer sur le bouton **Participer** et remplir le formulaire



The screenshot shows a web browser at the URL 127.0.0.1:8000/participation?id\_conc=11&id\_user=12. The page has a green header with 'Home', 'Resultats', and 'Logout' links. Below the header, there is a form with three input fields: 'Name', 'Lastname', and 'Phonenumber'. A green 'Register' button is positioned below the 'Phonenumber' field.

Ces données seront insérées dans la table **participants** qui a les attributs suivants :

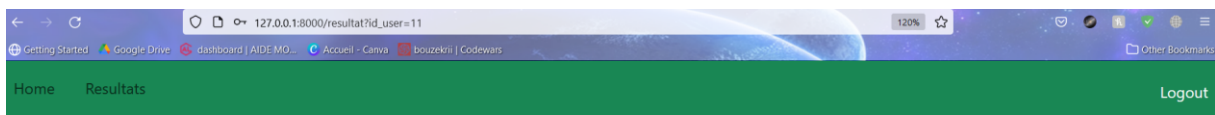


Ensuite il peut accéder aux résultats et voir s'il a gagné dans un concours :



Resultats				
Concours				
Id	Name	Description	rewards	Status
11	Concours de Samsung	Galaxy J1	2	Gagner

Et s'il n'a gagné aucun concours il aura l'affichage suivant :



Resultats				
Concours				
X Pas de chance !! vous n'avez pas gagner .				

### 3 – Fonctionnalité de la création de l'utilisateur à partir du ligne de commande :

J'ai créé la commande **create-user** à l'aide du ligne de commande suivant :

```
PS C:\Users\ing\Desktop\Jeux-de-concours-master> symfony console make:command
```

Cette commande a créé une classe *CreateUserCommand* dans le dossier *src/Command*, la commande prend en entrée deux arguments 'email' et 'password' si l'email existe déjà on aura le résultat suivant

```
PS C:\Users\ing\Desktop\Jeux-de-concours-master> php bin/console create-user test22@gmail.com 123456789
email already exists
```

En cherchant si cet email existe déjà dans la repository.

Sinon il va ajouter ces informations dans l'entité users.

```
PS C:\Users\ing\Desktop\Jeux-de-concours-master> php bin/console create-user test223@gmail.com 123456789
User is added successfully
PS C:\Users\ing\Desktop\Jeux-de-concours-master>
```

### 4-Fonctionnalité de l'extraction des données de l'utilisateur par l'organisateur en Json via API :

Pour cela j'ai ajouté la fonction *getAllPlayers* dans le contrôleur *Main* qui a une *JsonResponse* comme réponse et on utilise la fonction *findAll()* sur la repository des utilisateurs

```
#[Route('/api/v1/joueurs', name: 'app_joueurs')]
4 references | 0 overrides
public function getAllPlayers(EntityManagerInterface $em): JsonResponse
{
    $repository = $em->getRepository(User::class);
    $users = $repository->findAll();
    return $this->json($users);
}
```

Sur Postman on peut visualiser les données en spécifiant le chemin suivant :

<http://localhost:8000/api/v1/joueurs> dans le GET

Overview

GET http://localhost:8000/api/v1/joueurs

No E

http://localhost:8000/api/v1/joueurs

GET

http://localhost:8000/api/v1/joueurs

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE	DESCRIPTION
-----	-------	-------------

Body

Cookies

Headers (8)

Test Results

Status: 200 OK Time: 1304 ms Size:

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "id": 1,
3    "email": "email@email.com",
4    "roles": [
5      "ROLE_USER"
6    ],
7    "userIdentifier": "email@email.com"
8  },
9  {
10   "id": 2,
11   "email": "ff@gmail.com",
12   "roles": [
13     "ROLE_USER"
14   ],
15   "userIdentifier": "ff@gmail.com"
16 },
17 {
18   "id": 5,
19   "email": "email1@jkj.com",
20   "roles": [
21     "ROLE_USER"
22   ],
23   "userIdentifier": "email1@jkj.com"
24 },
25 {
26   "id": 6,
27   "email": "adminn@admin.com",
28   "roles": [
29     "ROLE_USER"
30   ],
31   "userIdentifier": "adminn@admin.com"
32 },
33 }
```

Activer Windows

Accédez aux paramètres

## **Conclusion :**

Ce projet m'a aidé pour appliquer et monter en compétence en Symfony d'une manière autonome, ainsi de développer ma connaissance au développement Back-end, même si que je n'ai pas pu donner mon maximum vu que j'ai passer par des problèmes de santé.