

# Documentation

# Hackat'Web

**Quentin Couzinet**



# Sommaire

|    |   |      |
|----|---|------|
| 01 | Contexte                                      | p 3  |
| 02 | Environnement de travail                      | p 4  |
| 03 | Architecture & configuration de l'application | p 5  |
| 04 | Base de données                               | p 8  |
| 05 | Page d'accueil                                | p 10 |
| 06 | Liste des hackathons                          | p 11 |
| 07 | Compte de participant                         | p 14 |
| 08 | Inscription aux hackathons                    | p 19 |
| 09 | Gestion des favoris                           | p 21 |
| 10 | Onglet « Mes hackathons »                     | p 24 |
| 11 | Conclusion                                    | p 26 |

# Contexte

L'entreprise Hackat'Agence a pour but de simplifier la gestion de l'organisation d'évènements rassemblant des personnes autour d'un projet informatique, appelés hackathons.

Pour cela plusieurs applications vont être développées dont notamment une application web Hackat'Web, qui devra permettre de consulter les informations sur les hackathons et de s'inscrire aux hackathons.

Plus précisément, les fonctionnalités attendues sont :

- Lister les hackathons
- Pouvoir rechercher un hackathon
- Système de compte avec connexion/déconnexion et inscription
- Pouvoir s'inscrire aux hackathons et visualiser les inscriptions
- Ajouter/retirer un hackathon des favoris et lister ceux-ci

L'objet de ce document est de traiter de la réalisation, et du fonctionnement de cette application.

# Environnement de travail

Cette application a été développée sur une machine virtuelle Linux Debian 11, créée sur le réseau du BTS grâce à la plateforme VMware vSphere mise à disposition.

L'adresse IP de cette machine est 192.168.51.98

L'application, développée sur le framework Symfony (v 6.3.7), respecte l'architecture MVC (modèle vue contrôleur). Ainsi, les langages utilisés sont principalement du PHP et du HTML, avec également du JS pour les fonctionnalités front-end. L'environnement de développement utilisé était Visual Studio Code, et une connexion SSH permettait de coder directement sur la VM.

Le lien du site est :

<http://192.168.51.98/hackatQuentin/hackatWeb/public/index.php/home>

La gestion des versions a été faite grâce à Git et à la création d'un dépôt distant sur Github accessible à cet adresse :

<https://github.com/soniiix/hackathon/tree/master/hackatWeb> et permettant de retrouver l'historique des modifications globales de la solution.

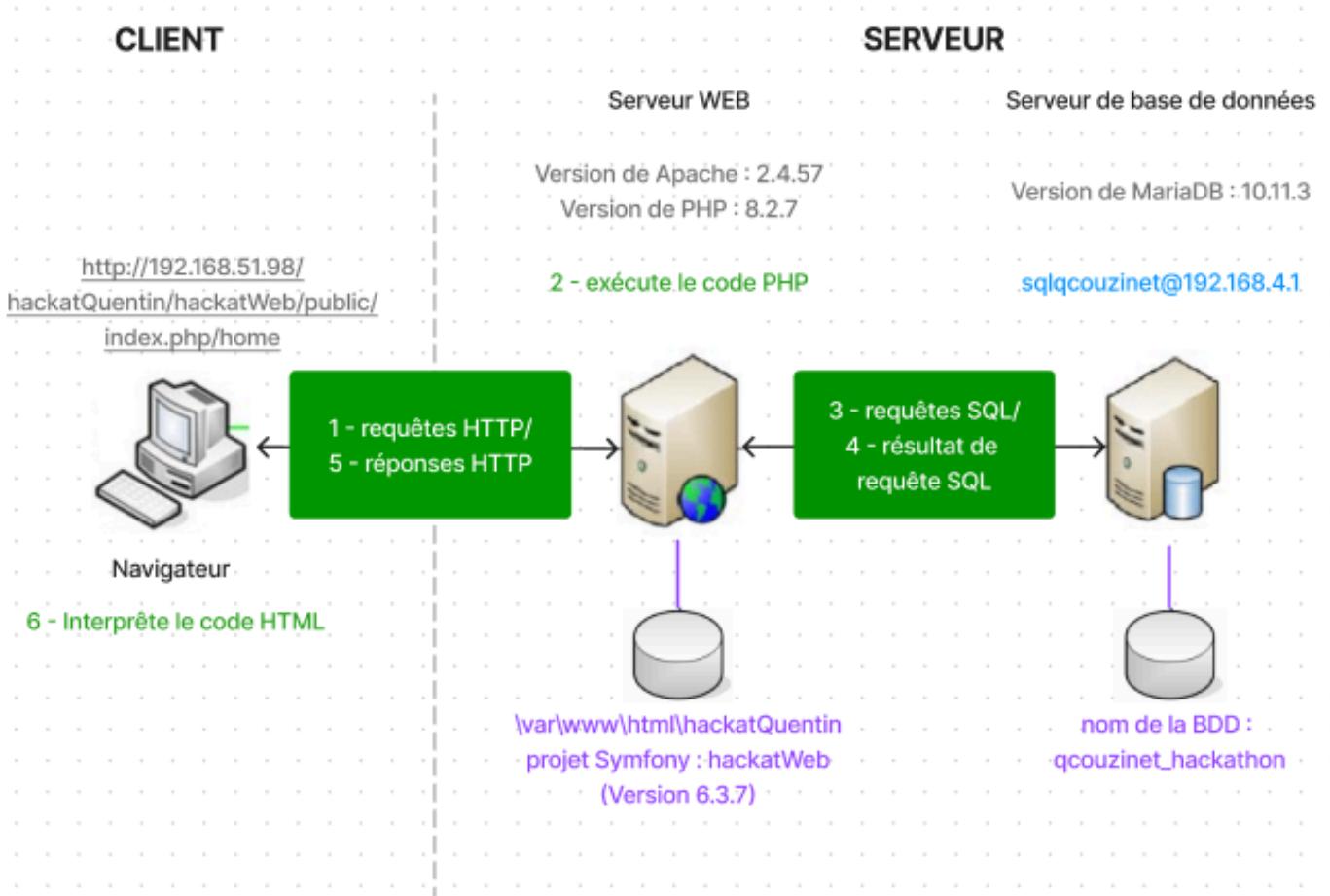
La gestion de ce projet a été faite sur Trello, et vous pouvez consulter la roadmap ici :

<https://trello.com/invite/b/mG4L34cv/ATTI6caa16096a647af1c45be7fc466c357c9DA474CB/hackatweb>

# Architecture de l'application

## A. Schéma d'architecture technique

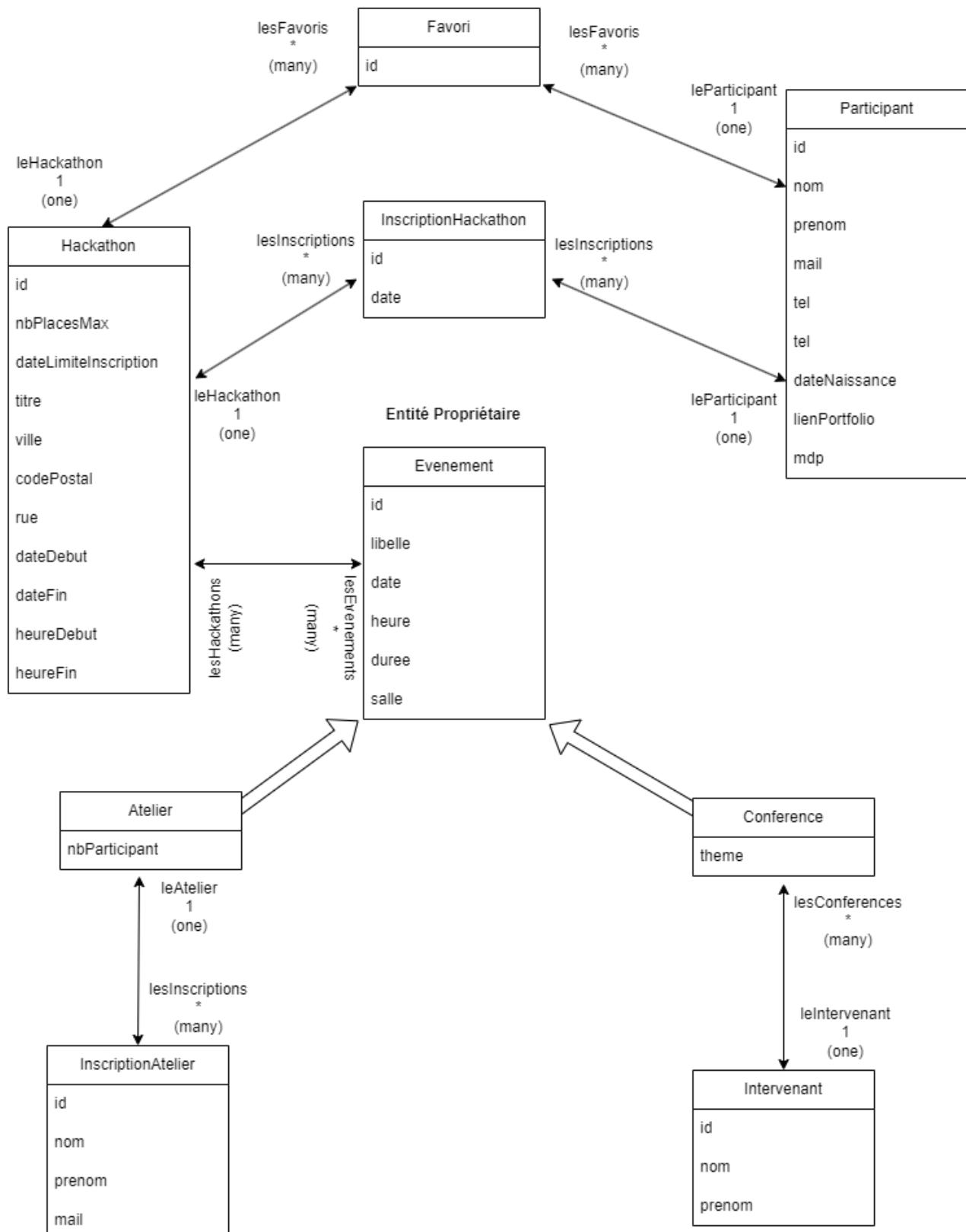
Voici le schéma de l'architecture technique de l'application Hackat'Web :



Il présente de manière détaillée la disposition, l'interaction, et les versions des différents composants logiciels et matériels.

## B. Diagramme des classes

Le diagramme des classes ci-dessous représente graphiquement l'organisation des classes et leurs relations dans l'application. Il offre une vue structurée de l'architecture de celle-ci.



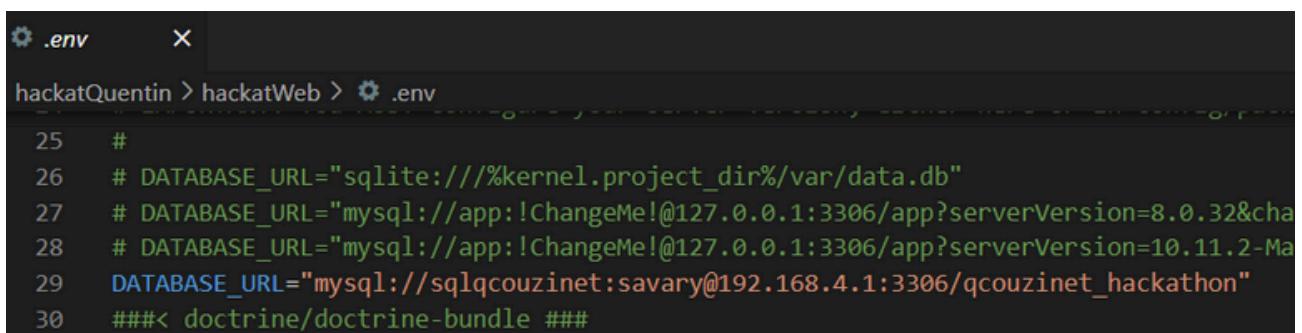
## C. Fichier de configuration

Voici le fichier .env de la solution, qui est le fichier de configuration principal de l'application, essentiel au fonctionnement de celle-ci.

La ligne la plus importante est la ligne 29 ci-dessous, qui spécifie l'URL de la base de données utilisée par l'application.

Cette URL, stockée dans la variable d'environnement 'DATABASE\_URL', est essentielle car elle indique l'emplacement et les paramètres de connexion à la base de données.

Ici, elle pointe vers la base de données MySQL "qcouzinet\_hackathon" hébergée à l'adresse IP 192.168.4.1 sur le port 3306.



```
hackatQuentin > hackatWeb > .env
25  #
26  # DATABASE_URL="sqlite:///kernel.project_dir/var/data.db"
27  # DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=8.0.32&cha
28  # DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=10.11.2-Ma
29  DATABASE_URL="mysql://sqlqcouzinet:savary@192.168.4.1:3306/qcouzinet_hackathon"
30  ###< doctrine/doctrine-bundle ###
```

Cette base de données sera expliquée dans la suite du document.

# Base de données

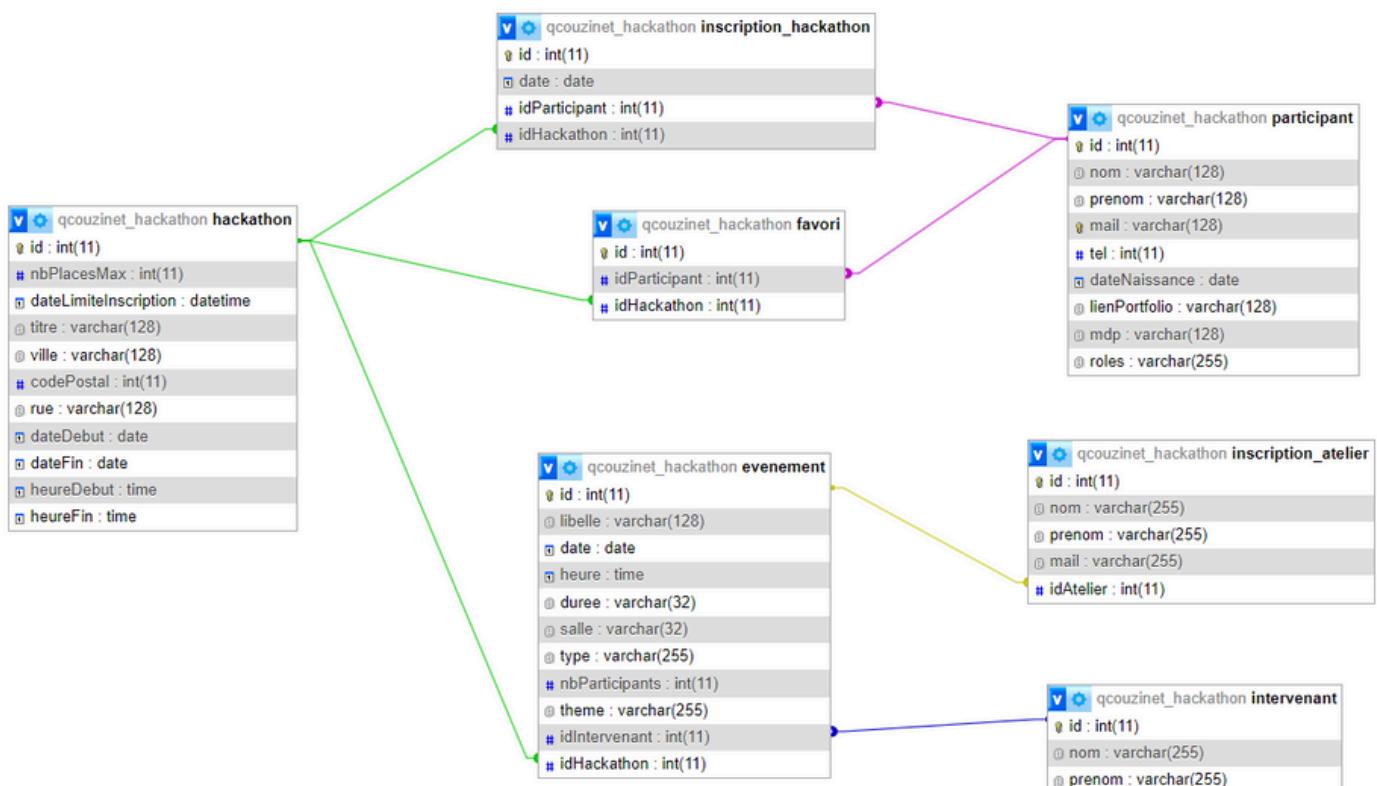
La base de données est essentielle au fonctionnement de l'application Hackat'Web.

Elle sert pour stocker et gérer toutes les informations relatives aux hackathons, aux utilisateurs et à leurs interactions au sein du site.

Retrouvez ci-après le schéma de la base de données, les différentes tables qui la composent, ainsi que les relations entre ces tables.

## A. Schéma conceptuel

Voici un schéma de la base de données :



Les tables utilisés pour l'application web sont :  
hackathon, inscription\_hackathon, favori, et participant

Dans la table hackathon, l'adresse a été séparée en plusieurs champs (ville, code postal, rue) pour des soucis de clarté et de lisibilité et pour simplifier les modifications éventuelles.

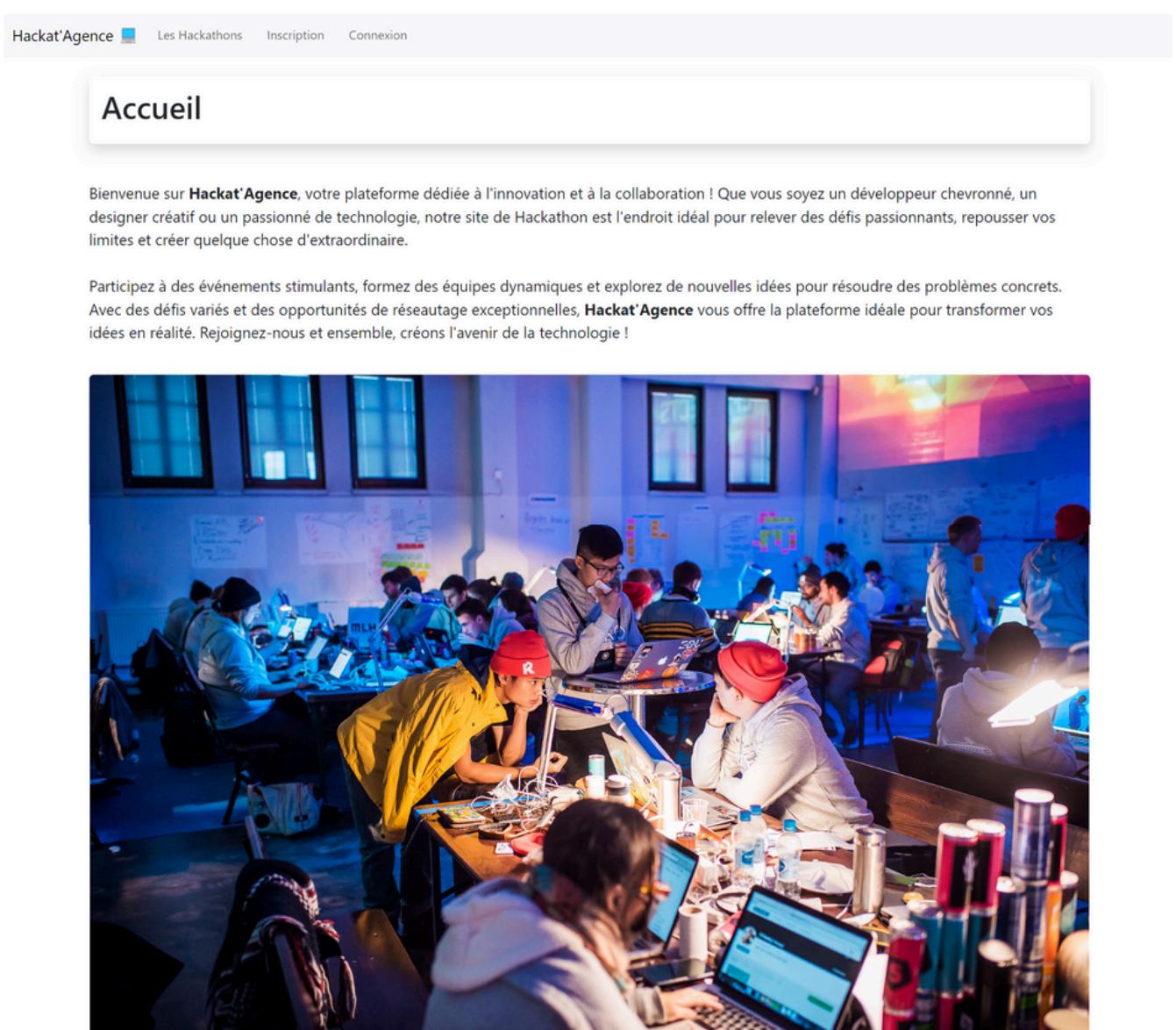
Dans la table inscription\_hackathon, le champ date correspond à la date de l'inscription (date et heure) à un hackathon émise par un participant.

Enfin, dans la table participant, le mot de passe est hashé pour des raisons de sécurité. En effet, cela garantit la protection contre les attaques de récupération de mot de passe. Même si la base de données est compromise, les mots de passe hashés sont difficiles à décrypter, car il est extrêmement difficile voire impossible de retrouver le mot de passe d'origine à partir du hash.

# Page d'accueil

La première page du site qui a été créée est la page d'accueil. Celle-ci présente dans une courte description ce qu'est Hackat'Agence.

Elle permet ainsi aux visiteurs d'avoir directement une idée de ce qu'est l'entreprise, et contient également une image d'illustration.



Hackat'Agence  Les Hackathons Inscription Connexion

## Accueil

Bienvenue sur **Hackat'Agence**, votre plateforme dédiée à l'innovation et à la collaboration ! Que vous soyez un développeur chevronné, un designer créatif ou un passionné de technologie, notre site de Hackathon est l'endroit idéal pour relever des défis passionnants, repousser vos limites et créer quelque chose d'extraordinaire.

Participez à des événements stimulants, formez des équipes dynamiques et explorez de nouvelles idées pour résoudre des problèmes concrets. Avec des défis variés et des opportunités de réseautage exceptionnelles, **Hackat'Agence** vous offre la plateforme idéale pour transformer vos idées en réalité. Rejoignez-nous et ensemble, créons l'avenir de la technologie !

A photograph showing a large room filled with people working at long tables during a hackathon. The room is dimly lit with blue and purple lights, and many laptops are open on the desks. There are also several Red Bull energy drink cans visible on the tables.

# Liste des hackathons

La page suivante est celle qui liste les hackathons de la base de données. Elle présente chacun sous forme de carte avec d'une part les informations les plus importantes : le titre, la date, et le nombre de places restantes;

The screenshot shows a web application interface for managing hackathons. At the top, there is a navigation bar with links for 'Hackat'Agence', 'Les Hackathons', 'Inscription', and 'Connexion'. Below the navigation bar, the main title 'Liste des hackathons' is displayed. To the right of the title are two buttons: 'Rechercher' (Search) and 'OK'. The main content area contains a grid of eight cards, each representing a different hackathon:

- CodeXplosion : Développez l'avenir du code**  
Date : 02/01/2025  
20 places restantes  
[Détails](#)
- QuantumQuest : Explorez les frontières de la programmation quantique**  
Date : 25/12/2024  
60 places restantes  
[Détails](#)
- ByteBurst Challenge : Explosiez les octets, créez le futur**  
Date : 01/12/2024  
0 places restantes  
[Détails](#)
- Défi CodePen**  
Date : 20/06/2024  
30 places restantes  
[Détails](#)
- TechTitans Hack : Affrontez les titans de la technologie**  
Date : 22/12/2023
- CyberForge : Forgez la sécurité numérique de demain**
- FutureFusion : Fusionnez les idées, forgez le futur du tech**  
Date : 15/12/2023
- RoboRush : Construisez des bots, dominez l'arène**  
Date : 15/12/2023

Et d'une autre part, en cliquant sur le bouton « Détails », les informations complémentaires, à savoir l'adresse, les dates et heures de début et de fin ainsi que la date limite d'inscription.

This screenshot shows a detailed view of the 'QuantumQuest' hackathon card from the previous list. The card displays the following information:

**QuantumQuest : Explorez les frontières de la programmation quantique**  
Date : 25/12/2024  
60 places restantes

A large button labeled 'Détails' is visible, which, when clicked, reveals additional details about the event:

- Adresse :** 75 Rue Louis Dansard, 69000, Lyon
- Dates précises :** Du 25/12/2024 à 14:00  
Au 27/12/2024 à 22:00
- Date limite d'inscription :** 20/12/2024

La liste des hackathons est récupérée dans la route `app_hackathon` du contrôleur `HackathonController`, en étant triée par date de début grâce au `findBy([ ], [ 'dateDebut' => 'DESC'])`, et est envoyé à la vue qui l'affiche ensuite sur la page comme décrit ci-dessus.

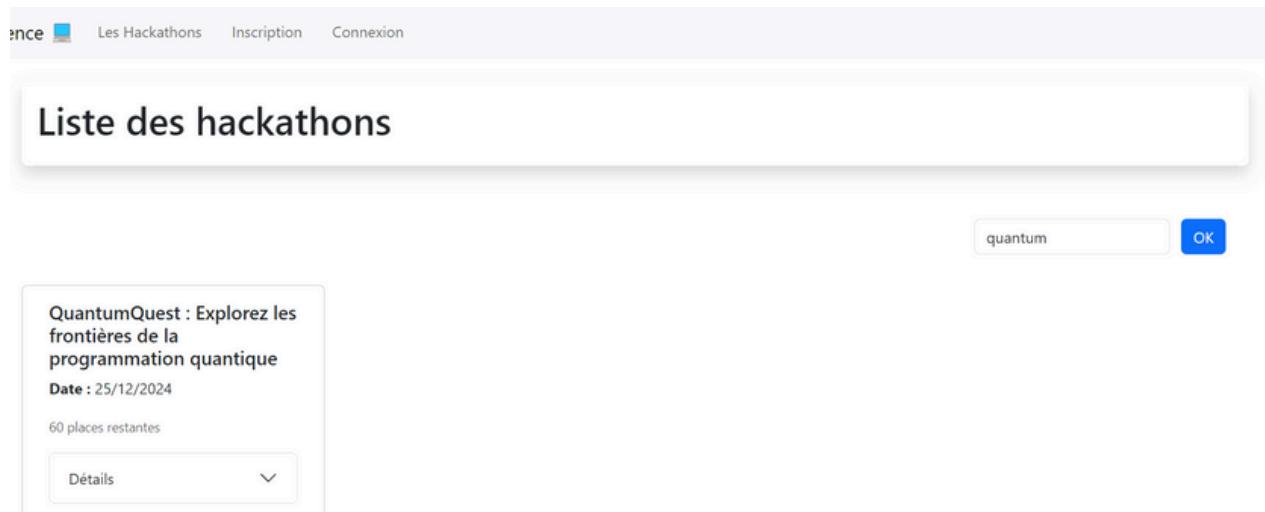
```
//===== Route pour voir tous les hackathons =====
#[Route('/hackathons', name: 'app_hackathon')]
public function index(ManagerRegistry $doctrine, Request $request): Response
{
    $hackathonRepository = $doctrine->getRepository(Hackathon::class);
    $favoriRepository = $doctrine->getRepository(Favori::class);

    $lesFavoris = $favoriRepository->findAll();
    $lesHackathons = $hackathonRepository->findBy([], [ 'dateDebut' => 'DESC']);

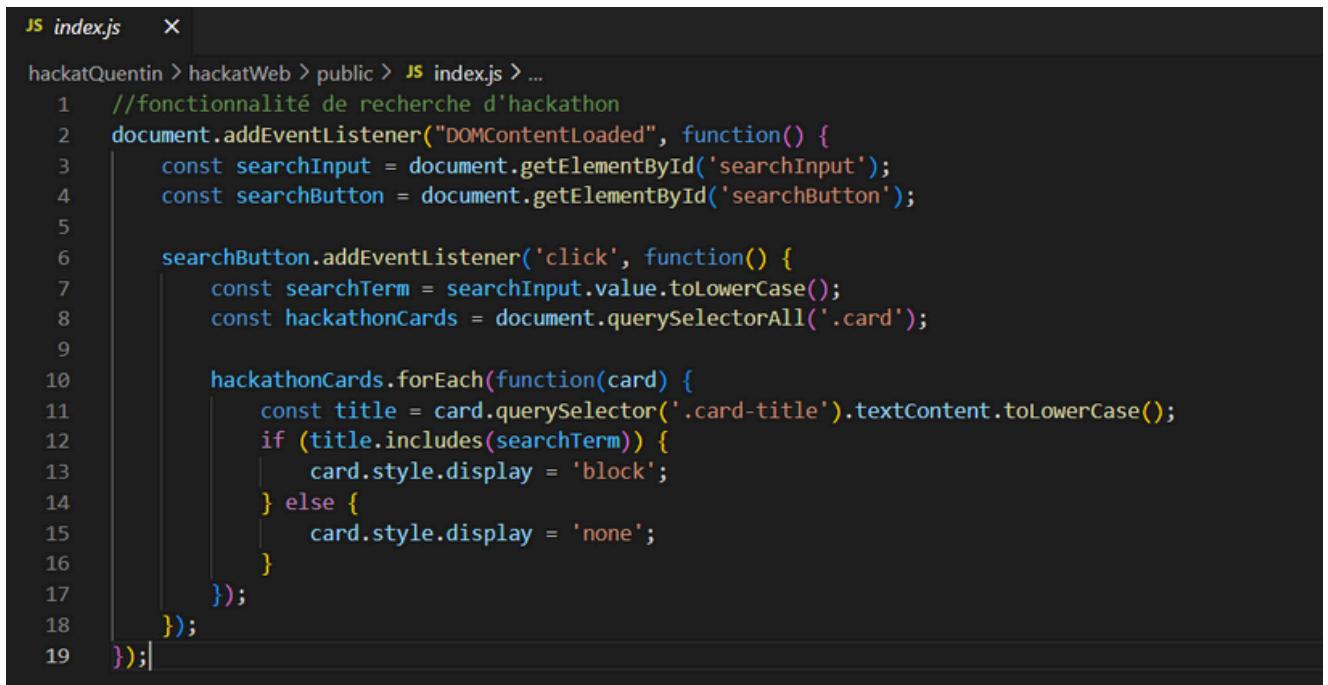
    return $this->render('hackathon/index.html.twig', ['lesHackathons' => $lesHackathons, 'lesFavoris' => $lesFavoris]);
}
```

## A. Fonctionnalité de recherche

Sur cette page, il est possible de rechercher un hackathon par titre. Il suffit de saisir un titre souhaité ou le début d'un titre puis de cliquer sur OK pour voir le hackathon recherché.



La recherche n'actualise pas la page car c'est une fonctionnalité côté client, intégrée en JavaScript. Voici son fonctionnement :



```
JS index.js X
hackatQuentin > hackatWeb > public > JS index.js > ...
1 //fonctionnalité de recherche d'hackathon
2 document.addEventListener("DOMContentLoaded", function() {
3     const searchInput = document.getElementById('searchInput');
4     const searchButton = document.getElementById('searchButton');
5
6     searchButton.addEventListener('click', function() {
7         const searchTerm = searchInput.value.toLowerCase();
8         const hackathonCards = document.querySelectorAll('.card');
9
10        hackathonCards.forEach(function(card) {
11            const title = card.querySelector('.card-title').textContent.toLowerCase();
12            if (title.includes(searchTerm)) {
13                card.style.display = 'block';
14            } else {
15                card.style.display = 'none';
16            }
17        });
18    });
19});
```

Le script JS est chargé dès le chargement de la page.

On récupère dans un premier temps les balises HTML concernés à savoir le champ de saisie et le bouton de validation.

Lorsqu'on clique sur le bouton, on récupère ce qui a été saisi dans le champ de texte, et on le met en minuscule. Chaque carte de la page correspondant à un hackathon est récupérée (grâce à sa classe 'card') et on parcours l'ensemble des hackathons afin de déterminer celui qui a le titre recherché. Si un hackathon correspond alors il est affiché, autrement il est masqué (style.display = 'none').

# Compte de participant

Grâce à Symfony et à son composant de sécurité (<https://symfony.com/doc/current/security.html>), il est possible de créer un système de comptes avec authentification et déconnexion.

Cela nous amène donc à la création de 2 nouvelles pages, une page de connexion, permettant à un participant de s'identifier :

Agence Les Hackathons Inscription Connexion

## Connexion

Adresse email :

Mot de passe :

**Connexion**

Et si le participant ne possède pas de compte, il peut en créer un via la page spécifique d'inscription :

Hackat'Agence Les Hackathons Inscription Connexion

## Inscription

Nom :

Prénom :

Adresse email :

Numéro de téléphone :

Date de naissance :  janv.  1904

Lien de votre portfolio :

Choisissez un mot de passe :

**S'inscrire**

Des vérifications sont effectives sur les champ de ce formulaire, empêchant la saisie d'informations erronées, ou le choix d'un mot de passe peut complexe. Exemples :

Adresse email :

0544

Veuillez saisir un email valide.

Lien de votre portfolio :

bonjour

Veuillez saisir un lien valide.

Choisissez un mot de passe :

Veuillez saisir un mot de passe complexe

Ce formulaire est créé en utilisant le composant de formulaires Symfony (<https://symfony.com/doc/current/forms.html>)

```
    IncriptionType.php ×
hackatQuentin > hackatWeb > src > Form >   IncriptionType.php > ...
14  use Symfony\Component\OptionsResolver\OptionsResolver;
15
16  class IncriptionType extends AbstractType
17  {
18      public function buildForm(FormBuilderInterface $builder, array $options): void
19      {
20          $builder
21              ->add('nom', TextType::class, ['label' => 'Nom :'])
22              ->add('prenom', TextType::class, ['label' => 'Prénom :'])
23              ->add('mail', TextType::class, ['label' => 'Adresse email :'])
24              ->add('tel', IntegerType::class, ['label' => 'Numéro de téléphone :'])
25              ->add('dateNaissance', BirthdayType::class, ['label' => 'Date de naissance :'])
26              ->add('lienPortfolio', TextType::class, ['label' => 'Lien de votre portfolio :'])
27              ->add('mdp', PasswordType::class, ['label' => 'Choisissez un mot de passe :'])
28          // ->add('roles')
29      ;
30  }
```

Ainsi, le formulaire est créé dans le contrôleur  
[InscriptionController](#) relatif à la page d'inscription (ligne 22)...

```
⌘ InscriptionController.php ×
hackatQuentin > hackatWeb > src > Controller > ⌘ InscriptionController.php > ...
14 class InscriptionController extends AbstractController
15 {
16     //===== Route pour enregistrer une nouvelle inscription ======
17     #[Route('/inscription', name: 'app_inscription')]
18     public function index(Request $request, ManagerRegistry $doctrine, UserPasswordHasherInterface $passwordHa
19     {
20         //création d'un objet participant et d'un form
21         $participant = new Participant();
22         $form=$this->createForm(InscriptionType::class, $participant);
23         $form->handleRequest($request);
24         $formInscription = $form->createView();
25
26         if ($form->isSubmitted() and $form->isValid()){
27             $entityManager = $doctrine->getManager();
28             $entityManager->persist($participant);
29
30             $mdpHash = password_hash($participant->getMdp(), PASSWORD_BCRYPT);
31             $participant->setMdp($mdpHash);
32
33             $entityManager->flush();
34
35             $this->addFlash('success', 'Votre compte a bien été créé. Veuillez vous connecter ci-dessous');
36             return $this->redirectToRoute('app_login');
37         }
38
39         return $this->render('inscription/index.html.twig', ['form' => $formInscription]);
40     }
41 }
42 }
```

...et est envoyé à la vue, qui l'affiche ensuite :

```
index.html.twig ×
hackatQuentin > hackatWeb > templates > inscription > index.html.twig
12
13     {{ form_start(form) }}
14     {{ form_row(form.nom) }}
15     {{ form_row(form.prenom) }}
16     {{ form_row(form.mail) }}
17     {{ form_row(form.tel) }}
18     {{ form_row(form.dateNaissance) }}
19     {{ form_row(form.lienPortfolio) }}
20     {{ form_row(form.mdp) }}
21     <button type="submit" method="POST" class="btn btn-primary">s'inscrire</button>
22     {{ form_end(form) }}
```

Si tous les champs sont remplis correctement, alors le compte du participant est bien créé (à noter que son mot de passe est hashé), il est redirigé vers la page de connexion et un message lui confirmant son inscription est affiché

The screenshot shows a web interface. At the top, there is a navigation bar with links: 'gence' (with a small logo), 'Les Hackathons', 'Inscription', and 'Connexion'. Below this, a green success message box contains the text: 'Votre compte a bien été créé. Veuillez vous connecter ci-dessous'. The main area is titled 'Connexion' and contains two input fields: 'Adresse email:' with the placeholder 'iharrigan0@furl.net' and 'Mot de passe:' with a masked input. A blue 'Connexion' button is located below the password field.

En saisissant correctement ses informations de connexion, le participant est redirigé vers la page d'accueil et est bien connecté (visible grâce au Profiler de Symfony) :

The screenshot shows a login form with fields for 'Adresse email:' containing 'iharrigan0@furl.net' and 'Mot de passe:' containing masked text. A blue 'Connexion' button is at the bottom. To the right, a dark sidebar displays the Symfony profiler with the following data:

|                 |                        |
|-----------------|------------------------|
| Logged in as    | Inessa Harrigan        |
| Authenticated   | Yes                    |
| Roles           | ROLE_USER              |
| Inherited Roles | none                   |
| Token class     | UsernamePasswordToken  |
| Firewall name   | main                   |
| Actions         | <a href="#">Logout</a> |

At the bottom of the sidebar, it says '4.0 MiB' for memory usage, '1 ms' for execution time, and '1 in 0.75' for the number of requests.

Une fois connecté, il a accès à de nouvelles fonctionnalités que nous verrons après.

La connexion s'effectue grâce au composant de sécurité de Symfony, et voici le contrôleur correspondant :

```
(LoginController.php) X
hackatQuentin > hackatWeb > src > Controller > LoginController.php > ...
9
10 class LoginController extends AbstractController
11 {
12     #[Route('/login', name: 'app_login')]
13     public function index(AuthenticationUtils $authenticationUtils): Response
14     {
15         // get the login error if there is one
16         $error = $authenticationUtils->getLastAuthenticationError();
17         // last username entered by the user
18         $lastUsername = $authenticationUtils->getLastUsername();
19         return $this->render('login/index.html.twig', ['last_username' => $lastUsername, 'errors' => $error]);
20     }
21
22     #[Route('/logout', name: 'app_logout')]
23     public function logout()
24     {
25     }
26 }
```

Il y a 2 routes, 1 pour la connexion et 1 pour la déconnexion.

La première est exécutée grâce au clic du bouton « Connexion » car le formulaire a un attribut action correspondant à cette route

```
<!-- formulaire de connexion -->
<form action="{{path('app_login')}}" method="post">
    <div class="form-group">
        <label for="login">Adresse email :</label>
        <input type="text" class="form-control" id="login" name="_username" value="{{last_username}}"/>
    </div>
    <br>
    <div class="form-group">
        <label for="password">Mot de passe :</label>
        <input type="password" class="form-control" id="password" name="_password"/>
    </div>
    <br>
    <button type="submit" class="btn btn-primary">Connexion</button>
</form>
```

Et la deuxième est exécutée grâce au bouton « Se déconnecter » disponible dans le menu de navigation.

Se déconnecter ➔

```
<li>
    <a class="nav-link" href="{{ path('app_logout') }}>Se déconnecter</a>
</li>
```

# Inscription aux hackathons

Si le participant est connecté, il a la possibilité de s'inscrire aux hackathons respectant ces critères :

- ceux ayant au moins 1 place restante
- ceux dont la date d'inscription n'est pas dépassé
- ceux pour lesquels le participant n'est pas déjà inscrit

The screenshot displays a grid of four hackathon cards:

- CodeXplosion : Développez l'avenir du code**  
Date : 02/01/2025  
20 places restantes  
Favori  
Détails S'inscrire
- QuantumQuest : Explorez les frontières de la programmation quantique**  
Date : 25/12/2024  
60 places restantes  
Favori  
Détails S'inscrire
- ByteBurst Challenge : Explorerez les octets, créez le futur**  
Date : 01/12/2024  
0 places restantes  
Favori  
Détails S'inscrire
- Défi CodePen**  
Date : 20/06/2024  
30 places restantes  
Favori  
Détails S'inscrire

Il lui suffit donc de cliquer sur le bouton "S'inscrire".

Un message s'affiche alors :

Inscription réussie !

Si le participant essaye de s'inscrire 2 fois, ça ne fonctionnera pas et un message d'erreur s'affiche :

## Liste des hackathons

You are already registered for this hackathon.

Rechercher

OK

CodeXplosion : Développez

QuantumQuest : Explorez les

ByteBurst Challenge :

Défi CodePen

Voici le code de la route "app\_inscription\_hackathon" du contrôleur **HackathonController**, qui gère l'inscription aux hackathons. Les commentaires expliquent globalement le fonctionnement, mais ce qu'il faut retenir c'est que la date de l'inscription est enregistrée automatiquement en fonction de la date et heure du jour, et qu'une vérification est faite empêchant un participant de s'inscrire plus d'une fois à un hackathon.

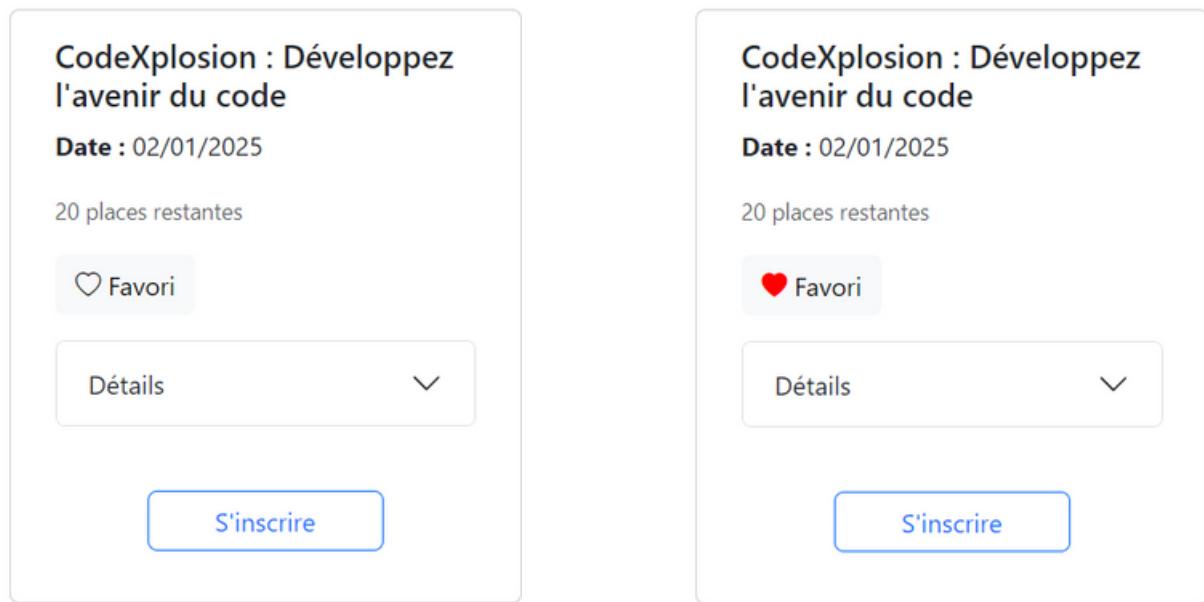
```
93     //===== Route permettant l'inscription à un hackathon =====
94     #[Route('/hackathon/{idHackathon}-{idParticipant}', name: 'app_inscription_hackathon')]
95     public function inscription(ManagerRegistry $doctrine, $idHackathon, $idParticipant): Response
96     {
97         $inscription = new IncriptionHackathon();
98
99         $hackathonRepository = $doctrine->getRepository(Hackathon::class);
100        $participantRepository = $doctrine->getRepository(Participant::class);
101        $inscriptionRepository = $doctrine->getRepository(IncriptionHackathon::class);
102
103        $dateInscription = new \DateTime();
104        $leHackathon = $hackathonRepository->find($idHackathon);
105        $leParticipant = $participantRepository->find($idParticipant);
106
107        //on cherche les inscriptions du participant
108        $exist = false;
109        $lesInscriptions = $inscriptionRepository->findBy(['leParticipant' => $leParticipant]);
110        foreach($lesInscriptions as $uneInscription){
111            if($uneInscription->getLeHackathon() == $leHackathon){
112                $exist = true;
113            }
114        }
115
116        //si le participant n'est pas déjà inscrit à cet hackathon
117        if ($exist == false){
118            $inscription->setDate($dateInscription);
119            $inscription->setLeHackathon($leHackathon);
120            $inscription->setLeParticipant($leParticipant);
121
122            $entityManager = $doctrine->getManager();
123            $entityManager->persist($inscription);
124            $entityManager->flush();
125
126            $this->addFlash('success', 'Inscription réussie !');
127            return $this->redirectToRoute('app_hackathon');
128        }
129        elseif ($exist == true){
130            $this->addFlash('failure', 'Vous êtes déjà inscrit à cet hackathon.');
131        }
132
133        return $this->redirectToRoute('app_hackathon');
134    }
135 }
```

# Gestion des favoris

Une autre fonctionnalité que j'ai intégrée est la gestion des favoris, c'est à dire que l'on peut ajouter ou retirer un hackathon des favoris.

Cette fonctionnalité est disponible côté client, ce qui veut dire qu'elle ne recharge pas la page.

Un bouton est affiché sur la carte des hackathons, et lorsque l'on clique dessus le script JS est exécuté, changeant ainsi l'icône et en ajoutant correctement le hackathon en favori.



Les favoris sont propres à chaque participant. Un enregistrement est fait dans la table favori de la BDD, contenant l'id du participant et l'id du hackathon.

Pour ce faire, j'ai créé une route d'API dans le contrôleur [HackathonController](#), qui enregistre le favori ou le supprime s'il existe déjà

```

nController.php X
in > hackatWeb > src > Controller > HackathonController.php > ...
===== Route d'API pour créer un favori =====
#[Route('/favoris/hackathons/{idHackathon}/participants/{idParticipant}', name: 'app_newfavorite')]
public function getFavoris(ManagerRegistry $doctrine, $idHackathon, $idParticipant)
{
    $hackathonRepository = $doctrine->getRepository(Hackathon::class);
    $participantRepository = $doctrine->getRepository(Participant::class);
    $favoriRepository = $doctrine->getRepository(Favori::class);

    //récupération des objets
    $leHackathon = $hackathonRepository->find($idHackathon);
    $leParticipant = $participantRepository->find($idParticipant);

    //on vérifie si le favori existe déjà...
    $leFavori = $favoriRepository->findBy(['leParticipant' => $leParticipant, 'leHackathon' => $leHackathon]);

    //...si ce n'est pas le cas
    if($leFavori == []){
        //création d'un nouveau favori correspondant
        $newFavori = new Favori();
        $newFavori->setLeHackathon($leHackathon);
        $newFavori->setLeParticipant($leParticipant);

        $entityManager = $doctrine->getManager();
        $entityManager->persist($newFavori);
        $entityManager->flush();

        $data = [
            'idHackathon' => $newFavori->getLeHackathon()->getId(),
            'idParticipant' => $newFavori->getLeParticipant()->getId()
        ];
        return new JsonResponse($data);
    }
    //...si c'est le cas, on supprime le favori
    else{
        $entityManager = $doctrine->getManager();
        $entityManager->remove($leFavori[0]);
        $entityManager->flush();
        return new JsonResponse(null, 409);
    }
}

```

Dans la vue, et plus précisément dans le script JS, on va effectuer une requête Ajax, au clic du bouton favori, grâce à l'utilisation d'un objet XMLHttpRequest.

```

<!-- Script d'ajout aux favoris -->
<script>
    function addToFavorite(n, idHackathon) {
        //récupération des éléments nécessaires
        var iconFavorite = document.getElementById('icon-favorite_' + n);
        var url = iconFavorite.getAttribute('data-url');

        var xhr = new XMLHttpRequest();
        xhr.open("get", url);
        xhr.responseType = "json";
        xhr.send();

        xhr.onload = function () {
            console.log(xhr.status);
            //si le hackathon est déjà en favori, on le retire
            if (xhr.status == 409) {
                //changement d'icône
                iconFavorite.className = "bi bi-heart";
                iconFavorite.style.color = "black";
            }
        }
    }

```

```

//Si le statut HTTP est 200, on ajoute l'hackathon aux favoris
} else if (xhr.status == 200) {
    //changement d'icône
    iconFavorite.className = "bi bi-heart-fill";
    iconFavorite.style.color = "red";
}
};

//Si la requête n'a pas pu aboutir...
xhr.onerror = function () {
    alert("La requête a échoué");
}

```

Si jamais la page est rechargée, les favoris sont bien affichés.

The screenshot shows a web interface for managing favorite hackathons. At the top, there's a navigation bar with links for 'Accès', 'Les Hackathons', 'Mes hackathons', and 'Se déconnecter'. Below the navigation is a search bar with a placeholder 'Rechercher' and a blue 'OK' button. The main content area is titled 'Liste des hackathons'. It displays four cards, each representing a hackathon:

- CodeXplosion : Développez l'avenir du code**  
Date : 02/01/2025  
20 places restantes  
**Favori** button  
Détails dropdown  
S'inscrire button
- QuantumQuest : Explorez les frontières de la programmation quantique**  
Date : 25/12/2024  
60 places restantes  
**Favori** button  
Détails dropdown  
S'inscrire button
- ByteBurst Challenge : Explorerez les octets, créez le futur**  
Date : 01/12/2024  
0 places restantes  
**Favori** button  
Détails dropdown
- Défi CodePen**  
Date : 20/06/2024  
30 places restantes  
**Favori** button  
Détails dropdown  
S'inscrire button

# Onglet « Mes hackathons »

Dans le menu, on retrouve un onglet "Mes hackathons" disponible quand le participant est connecté.

Cet onglet rassemble :

- les hackathons auxquels l'utilisateur s'est inscrit sur la page listant les hackathons

The screenshot shows a navigation bar with 'Hackat'Agence' and 'Mes hackathons' selected. Below it, a heading 'Liste de vos hackathons' is displayed. Two cards are shown: 'CodeXplosion : Développez l'avenir du code' (Date: 02/01/2025) and 'QuantumQuest : Explorez les frontières de la programmation quantique' (Date: 25/12/2024), each with a 'Détails' button.

- les hackathons ayant été mis en favoris, affichés dans une rubrique "Favoris"

The screenshot shows a 'Favoris' section with three cards: 'DataDive : Plongez dans le monde des données' (Date: 05/12/2023), 'CodeXplosion : Développez l'avenir du code' (Date: 02/01/2025), and 'ByteBurst Challenge : Explorerez les octets, créez le futur' (Date: 01/12/2024), each with a 'Détails' button.

Cette page est évidemment mise à jour à chaque actualisation.

Dans la route "app\_meshackathons" qui s'occupe de cette fonctionnalité, on va récupérer, grâce aux informations du participant connecté (ligne 83) et plus précisément de son identifiant, les inscriptions et le favoris qui lui sont propres grâce à la méthode `findBy` (ligne 86 et 87).

Ensute ces deux listes sont envoyés à la vue qui les affiche.

```
75 //===== Route pour voir les hackathons du participant connecté, ainsi que ses favoris =====
76 #[Route('/meshackathons/{idParticipant}', name: 'app_meshackathons')]
77 public function meshackathons(ManagerRegistry $doctrine, Request $request, $idParticipant): Response
78 {
79     $participantRepository = $doctrine->getRepository(Participant::class);
80     $inscriptionRepository = $doctrine->getRepository(InscriptionHackathon::class);
81     $favoriRepository = $doctrine->getRepository(Favori::class);
82
83     $leParticipant = $participantRepository->find($idParticipant);
84
85     //récupération des inscriptions et des favoris liés au participant
86     $lesInscriptions = $inscriptionRepository->findBy(['leParticipant' => $leParticipant]);
87     $lesFavoris = $favoriRepository->findBy(['leParticipant' => $leParticipant]);
88
89     return $this->render('meshackathons/index.html.twig', ['lesInscriptions' => $lesInscriptions, 'lesFavoris' => $lesFavoris]);
90 }
```

# Conclusion

Ce projet, réalisé en groupe, a été effectué durant toute la 2ème année de BTS SIO (2023-2024) durant les ateliers de professionnalisation et à la maison.

Il nous a permis de mettre en pratique les connaissances théoriques acquises en classe et de les appliquer dans un contexte réel. Cela nous a donné une expérience précieuse qui nous servira dans le monde professionnel.