TodoList - Reprise du projet

DEFAULT CONTROLLER

indexAction() - Route : "/"

```
class DefaultController extends Controller
{
/**
* @Route("/", name="homepage")
*/
public function indexAction()
{
return $this->render('default/index.html.twig');
}
}
```

Fonction : Rend la vue "default/index.html.twig". Cette action est liée à la route "/" associée à la page d'accueil de l'application. Elle renvoie simplement la vue par défaut.

TASK CONTROLLER

listAction() - Route : "/tasks"

```
/**

@Route("/tasks", name="task_list")
*/
public function listAction()
{
return $this->render('task/list.html.twig', ['tasks' => $this-
>getDoctrine()->getRepository('AppBundle:Task')->findAll()]);
}
```

Fonction : Récupère toutes les tâches depuis la base de données à l'aide du gestionnaire d'entités Doctrine et les transmet à la vue "task/list.html.twig" pour les afficher.

createAction(Request $request) - Route : "/tasks/create"

```
/**

@Route("/tasks/create", name="task_create")
```

```
*/
public function createAction(Request $request)
{
$task = new Task();
$form = $this->createForm(TaskType::class, $task);

$form->handleRequest($request);

if ($form->isValid()) {
$em = $this->getDoctrine()->getManager();
$em->persist($task);
$em->flush();

$this->addFlash('success', 'La tâche a été bien été ajoutée.');

return $this->redirectToRoute('task_list');
}

return $this->render('task/create.html.twig', ['form' => $form-
>createView()]);
}
```

Fonction : Gère la création d'une nouvelle tâche. Crée un nouvel objet Task, crée un formulaire à partir du type de formulaire TaskType, et traite la requête HTTP. Si le formulaire est valide, la nouvelle tâche est persistée dans la base de données, et l'utilisateur est redirigé vers la liste des tâches avec un message flash de succès.

editAction(Task $task, Request $request) - Route : "/tasks/{id}/edit"

```
/**

@Route("/tasks/{id}/edit", name="task_edit")
*/
public function editAction(Task $task, Request $request)
{
$form = $this->createForm(TaskType::class, $task);

$form->handleRequest($request);

if ($form->isValid()) {
$this->getDoctrine()->getManager()->flush();

$this->addFlash('success', 'La tâche a bien été modifiée.');

return $this->redirectToRoute('task_list');
}

return $this->render('task/edit.html.twig', [
'form' => $form->createView(),
'task' => $task,
```

```
]);
}
```

Fonction : Gère la modification d'une tâche existante. Récupère la tâche existante à partir de l'ID passé en paramètre, crée un formulaire de modification à partir de TaskType, et traite la requête HTTP. Si le formulaire est valide, les modifications sont persistées dans la base de données, et l'utilisateur est redirigé vers la liste des tâches avec un message flash de succès.

toggleTaskAction(Task $task) - Route : "/tasks/{id}/toggle"

```
/**

@Route("/tasks/{id}/toggle", name="task_toggle")
*/
public function toggleTaskAction(Task $task)
{
$task->toggle(!$task->isDone());
$this->getDoctrine()->getManager()->flush();

$this->addFlash('success', sprintf('La tâche %s a bien été marquée comme
faite.', $task->getTitle()));

return $this->redirectToRoute('task_list');
}
```

Fonction : Bascule l'état "done" d'une tâche entre vrai et faux. Si la tâche était marquée comme terminée, elle sera marquée comme non terminée, et vice versa. Les modifications sont persistées dans la base de données, et l'utilisateur est redirigé vers la liste des tâches avec un message flash de succès.

deleteTaskAction(Task $task) - Route : "/tasks/{id}/delete"

```
/**

@Route("/tasks/{id}/delete", name="task_delete")
*/
public function deleteTaskAction(Task $task)
{
$em = $this->getDoctrine()->getManager();
$em->remove($task);
$em->flush();

$this->addFlash('success', 'La tâche a bien été supprimée.');

return $this->redirectToRoute('task_list');
}
```

Fonction : Supprime une tâche de la base de données. La tâche à supprimer est spécifiée par l'ID passé en paramètre. Une fois supprimée, l'utilisateur est redirigé vers la liste des tâches avec un message flash de succès.

SECURITY CONTROLLER

loginAction(Request $request) - Route : "/login"

```
/**

@Route("/login", name="login")
*/
public function loginAction(Request $request)
{
$authenticationUtils = $this->get('security.authentication_utils');

$error = $authenticationUtils->getLastAuthenticationError();
$lastUsername = $authenticationUtils->getLastUsername();

return $this->render('security/login.html.twig', array(
'last_username' => $lastUsername,
'error' => $error,
));
}
```

Fonction : Gère l'affichage du formulaire de connexion. Il utilise le service security.authentication_utils pour récupérer les erreurs d'authentification éventuelles et le dernier nom d'utilisateur saisi. Ces informations sont ensuite transmises à la vue "security/login.html.twig" pour affichage.

loginCheck() - Route : "/login_check"

```
/**

@Route("/login_check", name="login_check")
*/
public function loginCheck()
{
// This code is never executed.
}
```

Fonction : Une route utilisée par Symfony pour la vérification automatique des informations d'identification lors de la soumission du formulaire de connexion. L'authentification est gérée par le composant de sécurité de Symfony.

logoutCheck() - Route : "/logout"

```
/**

@Route("/logout", name="logout")
*/
public function logoutCheck()
{
// This code is never executed.
}
```

Fonction : Une route utilisée par Symfony pour gérer la déconnexion. Le composant de sécurité gère automatiquement le processus de déconnexion.

USER CONTROLLER

listAction() - Route : "/users"

```
/**
* @Route("/users", name="user_list")
*/
public function listAction()
{
return $this->render('user/list.html.twig', ['users' => $this-
>getDoctrine()->getRepository('AppBundle:User')->findAll()]);
}
```

Fonction : Récupère tous les utilisateurs depuis la base de données à l'aide de Doctrine et les transmet à la vue "user/list.html.twig" pour les afficher.

createAction(Request $request) - Route : "/users/create"

```
/**
* @Route("/users/create", name="user_create")
*/
public function createAction(Request $request)
{
$user = new User();
$form = $this->createForm(UserType::class, $user);

$form->handleRequest($request);

if ($form->isValid()) {
$em = $this->getDoctrine()->getManager();
$password = $this->get('security.password_encoder')->encodePassword($user,
$user->getPassword());
$user->setPassword($password);
```

```
$em->persist($user);
$em->flush();

$this->addFlash('success', "L'utilisateur a bien été ajouté.");

return $this->redirectToRoute('user_list');
}

return $this->render('user/create.html.twig', ['form' => $form-
>createView()]);
}
```

Fonction : Gère la création d'un nouvel utilisateur. Crée un nouvel objet User, crée un formulaire à partir du type de formulaire UserType, et traite la requête HTTP. Si le formulaire est valide, le mot de passe de l'utilisateur est encodé à l'aide du service security.password_encoder, l'utilisateur est persisté dans la base de données, et l'utilisateur est redirigé vers la liste des utilisateurs avec un message flash de succès.

editAction(User $user, Request $request) - Route : "/users/{id}/edit"

```
/**
* @Route("/users/{id}/edit", name="user_edit")
*/
public function editAction(User $user, Request $request)
{
$form = $this->createForm(UserType::class, $user);

$form->handleRequest($request);

if ($form->isValid()) {
$password = $this->get('security.password_encoder')->encodePassword($user,
$user->getPassword());
$user->setPassword($password);

$this->getDoctrine()->getManager()->flush();

$this->addFlash('success', "L'utilisateur a bien été modifié");

return $this->redirectToRoute('user_list');
}

return $this->render('user/edit.html.twig', ['form' => $form-
>createView(), 'user' => $user]);
}
```

Fonction : Gère la modification d'un utilisateur existant. Récupère l'utilisateur existant à partir de l'ID passé en paramètre, crée un formulaire de modification à partir de UserType, et traite la requête HTTP. Si le formulaire est valide, le mot de passe de l'utilisateur est encodé à l'aide du service security.password_encoder, les modifications sont persistées dans la base de données, et l'utilisateur est redirigé vers la liste des utilisateurs avec un message flash de succès.

ACTIONS A METTRE EN PLACE

- Mise à jour de Symfony et de ses dépendances
- Corrections d'anomalies : une tâche doit être attachée à un utilisateur
- Corrections d'anomalies : choisir un rôle pour un utilisateur
- Nouvelle fonctionnalité : autorisation (gestion users par ADMIN, suppression tâche par initiateur, user "anonyme")
- Nouvelle fonctionnalité : implémentation de tests automatisés
- Créer une documentation concernant l'authentification