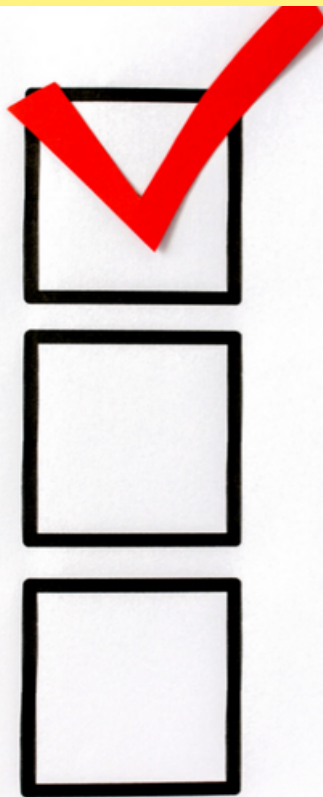




DOCUMENTATION TECHNIQUE

TO DO LIST

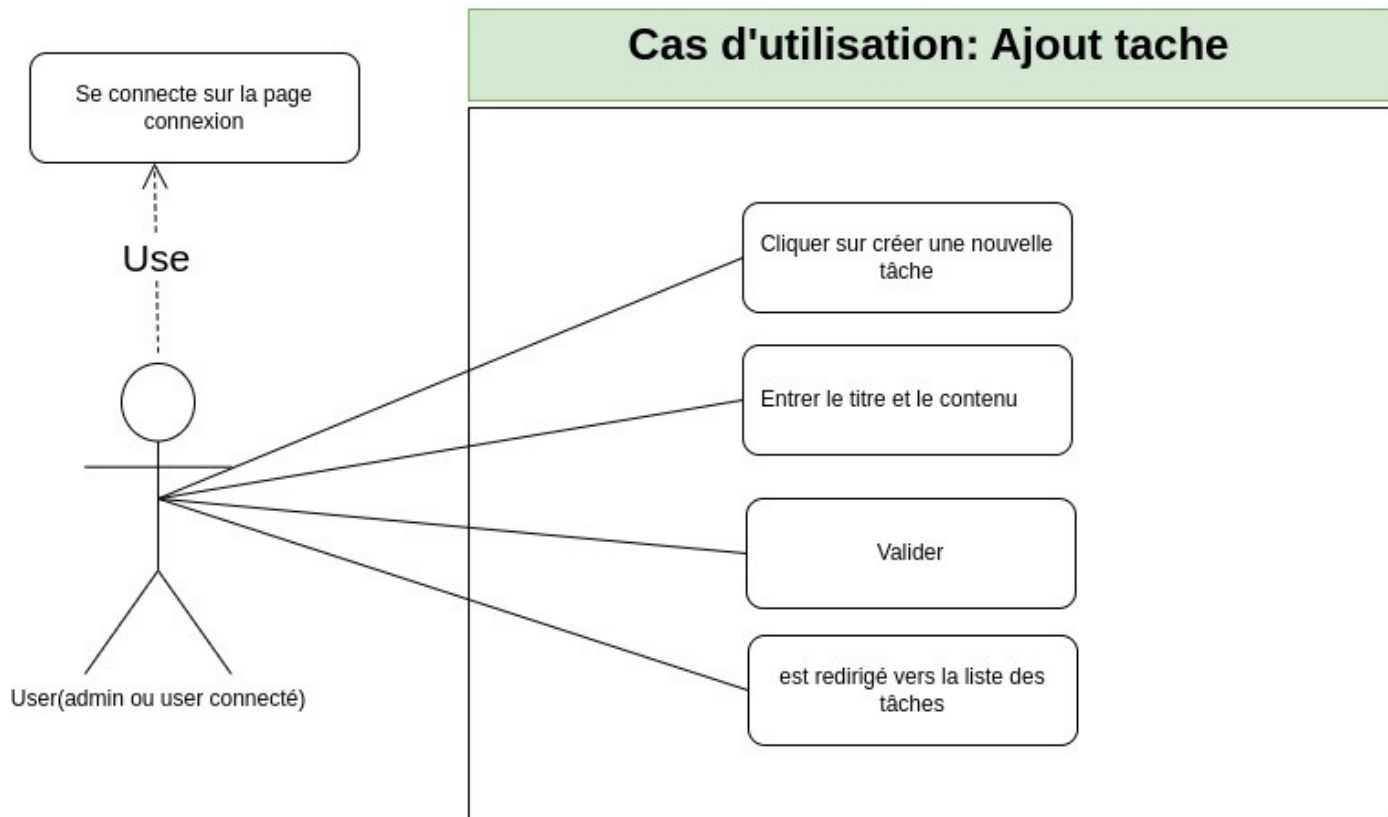


INTRODUCTION

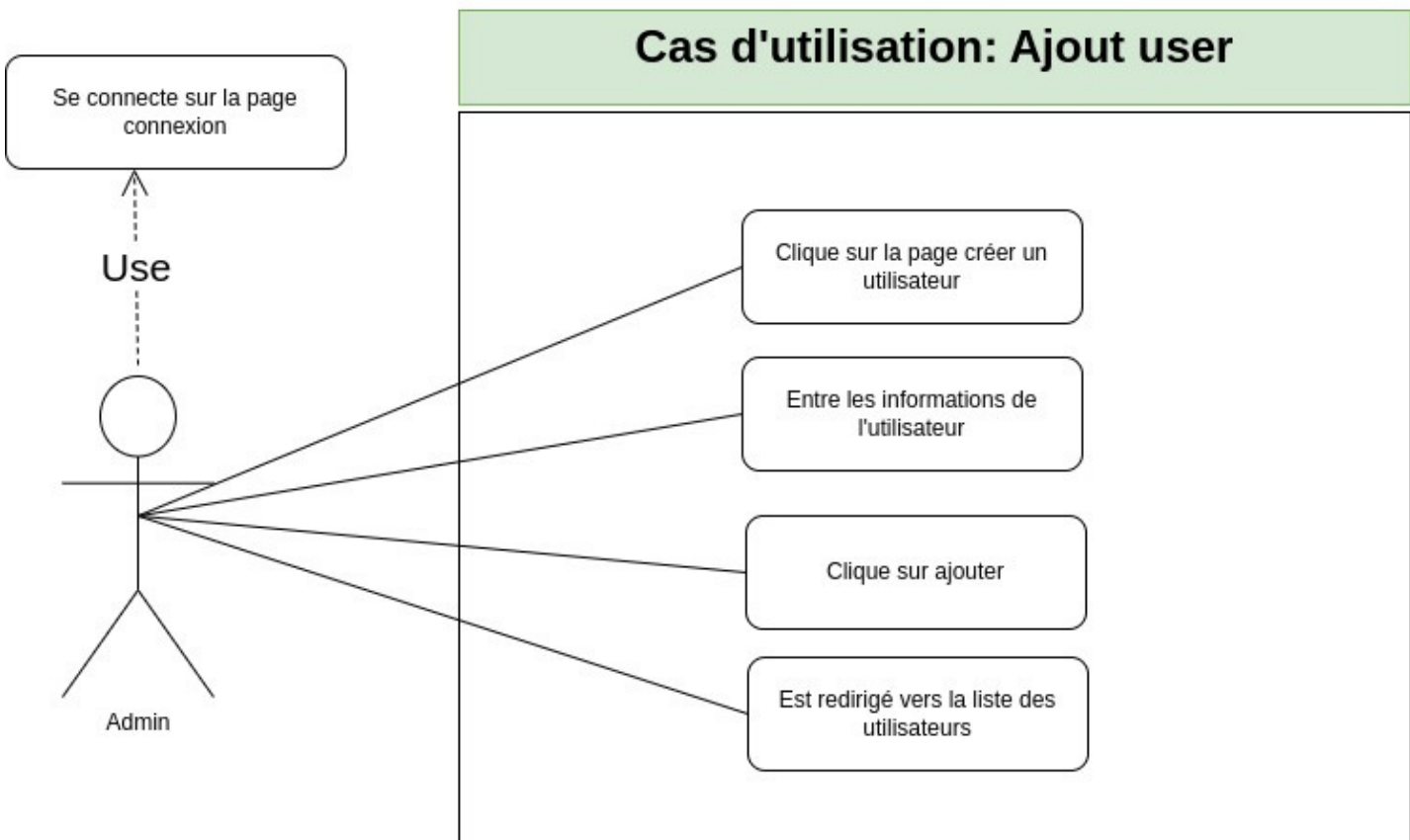


01.LES CAS D'UTILISATION

To do List



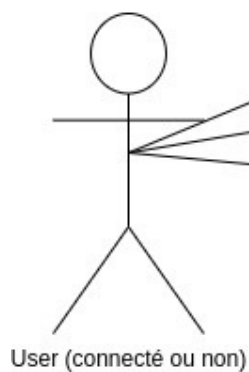
To do List



LES CAS D'UTILISATION

To do List

Cas d'utilisation: Done



Accéder à la liste des tâches

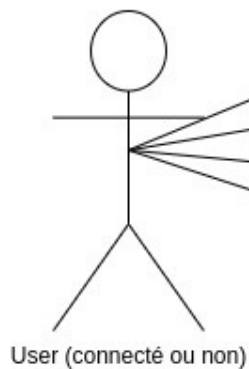
Cibler la tâche effectuée

Cliquer sur "marquer comme faite"

To do List

Cas d'utilisation: suppression de la tâche

Connecté en tant qu'admin si la tâche est anonyme ou en tant que user ayant créé la tâche.



Use

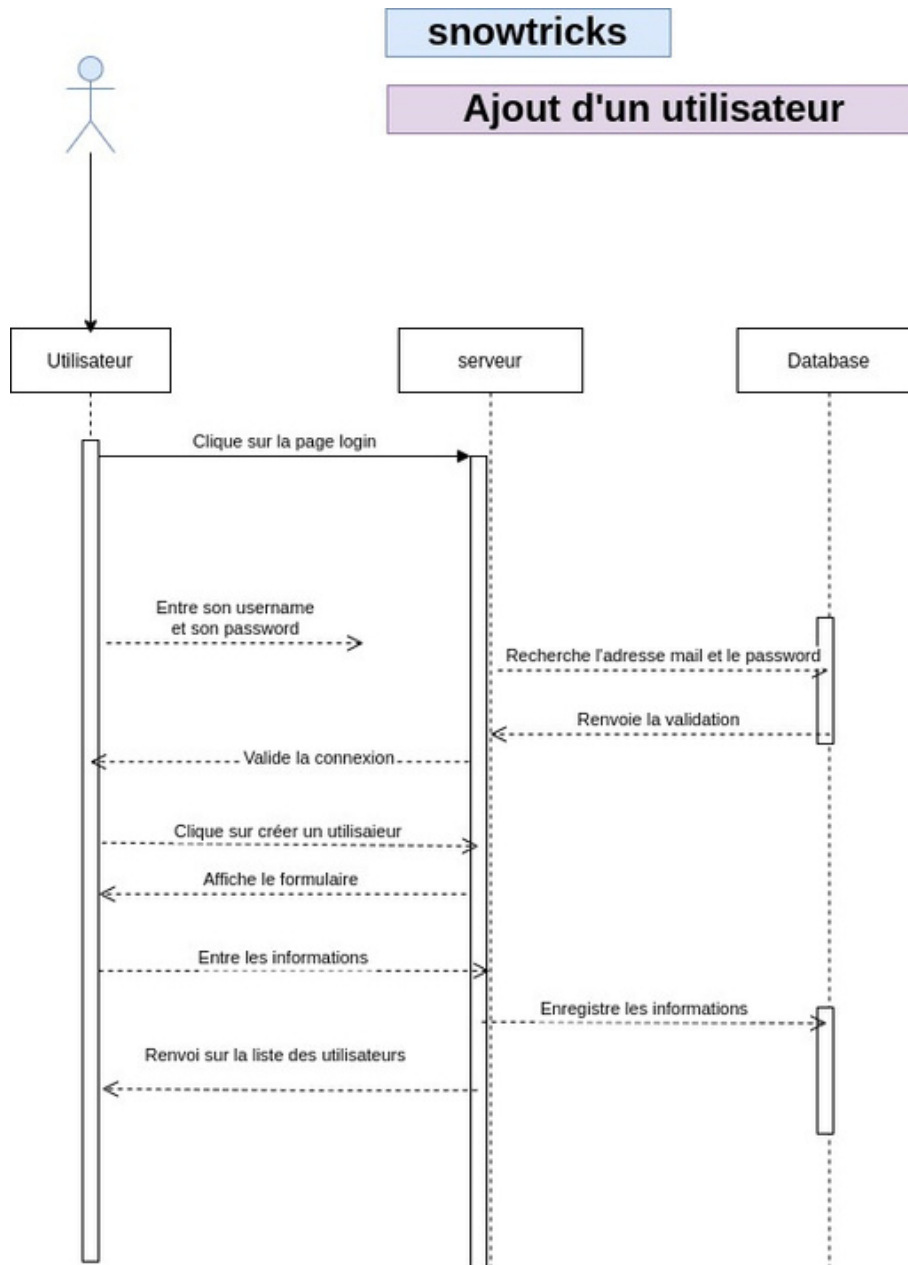
Accéder à la liste des tâches

Cibler la tâche à supprimer

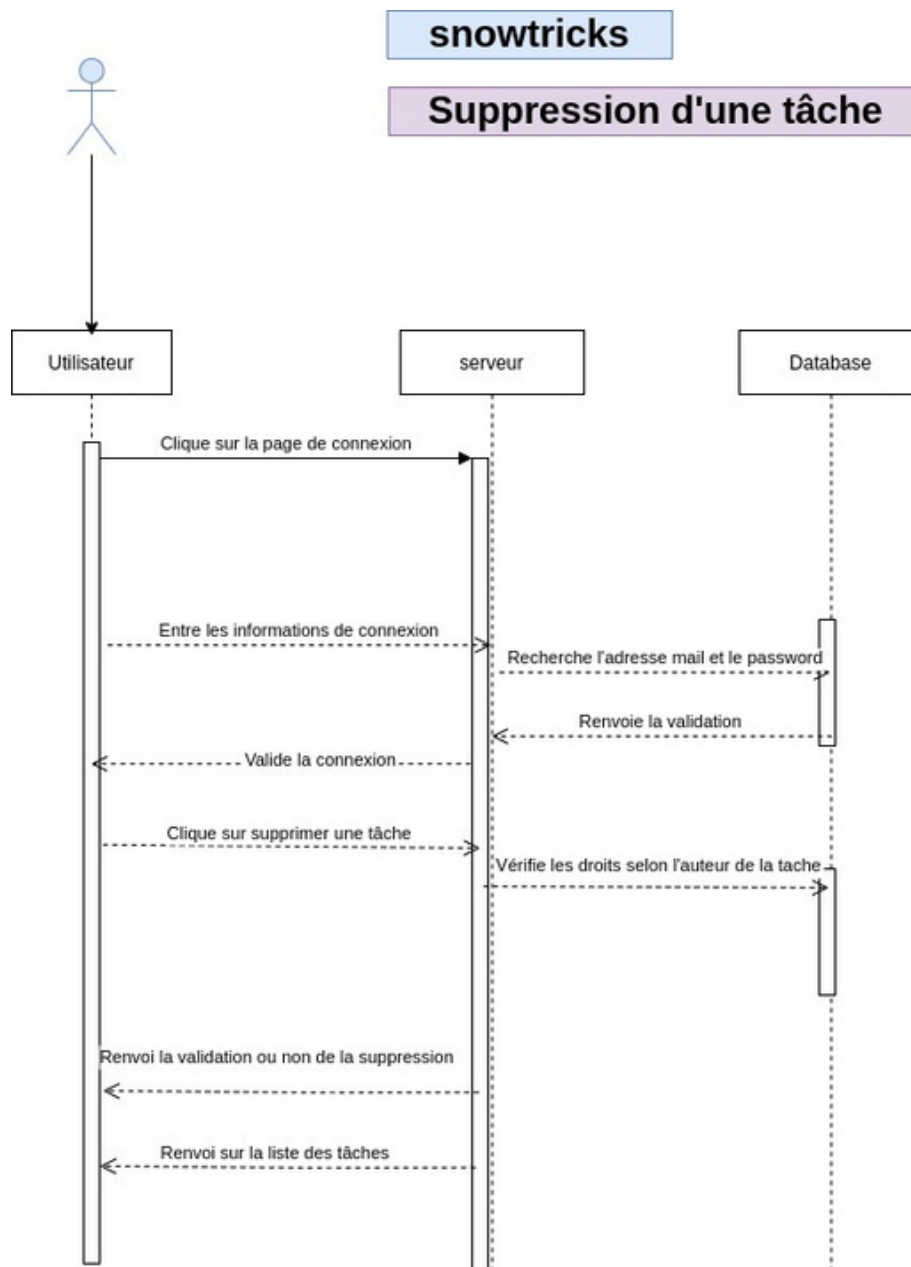
Cliquer sur supprimer

Est supprimée

02.LES DIAGRAMMES DE SÉQUENCE

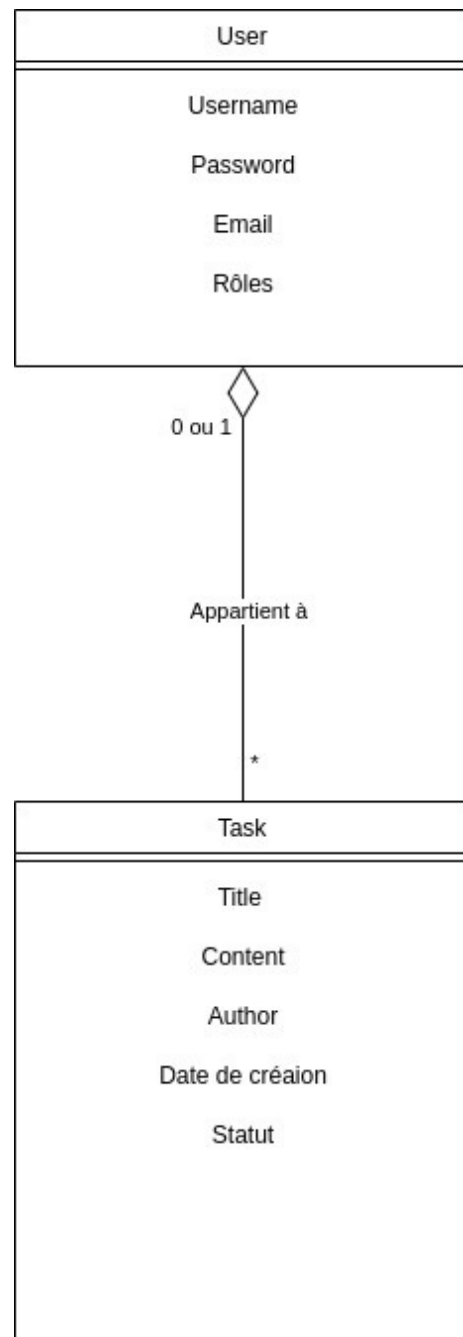


LES DIAGRAMMES DE SÉQUENCE



03.Les modèles de données

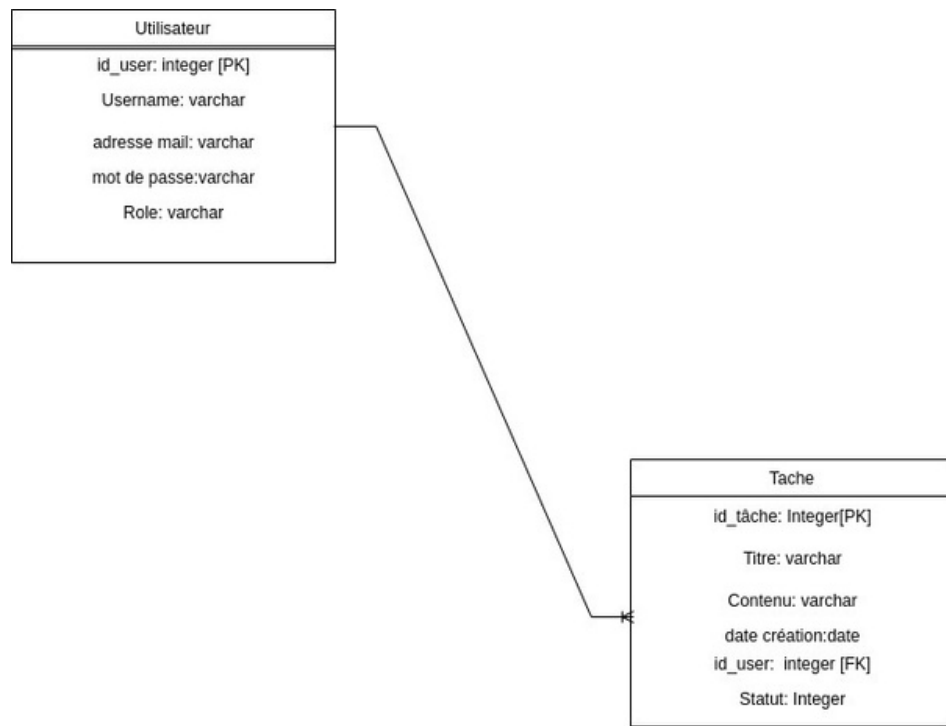
Le modèle conceptuel de données



Le modèle conceptuel de données montre un lien entre les tâches et les utilisateurs. Un utilisateur peut avoir plusieurs tâches et une tâche peut être rattachée à 0 ou 1 utilisateur. Si elle n'est pas rattachée à un utilisateur celui ci sera anonyme.

Les modèles de données

Le modèle physique de données



Le modèle physique de données montre une référence de chaque utilisateur et tâche par sa clé unique (id). précision: l'auteur anonyme n'est pas référencé en base de données donc n'apparaît pas en tant que user en modèle de données.

04-LA MISE EN PLACE DU PROJET

Le projet TodoList est une application permettant de gérer ses tâches quotidiennes. Ce projet a été développé à ce jour dans l'idée d'un futur développement grâce à une levée de fonds. Il utilise le framework symfony.



A- L'initialisation du projet

Description de l'initialisation du projet- Les débuts du projet et les potentiels d'évolution.

B- L'organisation du projet

Organisation globale du projet Comment gérer les modifications et évolutions du projet.

C- Les tests unitaires et fonctionnels

A- L'initialisation du projet

Clonez le projet ici:

`git@github.com:marinejourdan/PROJET8-Todolist.git`

- Realisation technique du blog

Visual studio code-éditeur de texte (html/php/css)

phpmyadmin: serveur de gestion de base de données

Git/Github: Gestionnaire de versions

- Installation du projet

Installez git et récupérez le projet de Github

`git@github.com:marinejourdan/PROJET8-Todolist.git`

Paramêtrer le projet avec vos propres données (.env).

-exécutez les migrations (3 versions à exécuter avec chargement du jeu de données dans les migrations)

- composer.json

Au sein de votre composer.json à la racine du projet, vous aurez accès aux versions actuelles des bundles et services de votre projet.

Il vous faudra faire évoluer le projet à partir des mises à jour de ce composer.

- console

Accès à la console au sein du dossier bin au sein du projet.

- security.yml

Accès à la partie sécurité du projet : connexions et rôles définis.
access_control:

- { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/logout, roles: ROLE_USER }
- { path: ^/tasks/.*, roles: ROLE_USER }
- { path: ^/tasks, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/users, roles: ROLE_ADMIN }

role_hierarchy:

ROLE_ADMIN: ROLE_USER

LES POINTS PRINCIPAUX DU PROJET

Au sein du projet vous aurez accès à plusieurs dossiers:

- **Cas utilisation** : Cas pratique pour l'utilisateur du projet
- **les diagrammes de séquences** : échanges client/serveur et base de données
- **modèles de données**: modèle conceptuel et physique de données/organisation des relations basées sur le modèle UML.
- **config**: Outils de configuration du projet (security)
affichage des pages en twig
- **bin**: console
- **src**: data fixtures- controller - entités- formulaires
- **var**: cache-logs-sessions
- **vendor**: bundles
- **web**:css, font, images...
- **tests**: à effectuer après chaque modification
- **migrations**: à effectuer lors de l'initialisation du projet

LES ETAPES DE GESTION DU PROJET

Notre organisation de travail se déroule en étape, chaque modification entraîne des points obligatoires pour permettre une évolution sécurisée du projet et un travail en équipe.

1- Demande de modification du projet

2- réflexion et (re)écriture des tests unitaires et fonctionnels en adéquation avec la modification à apporter (attention, certains tests fonctionnels seront à modifier si les fonctions évoluent)

3- modification à apporter dans la fonction

4- passage en revue en ligne de commande des tests complets unitaires et fonctionnels

5- correction des éventuels bugs lors des tests

6- enregistrement de la couverture de code

7- Validation avec le chef de projet

Attention ! Chaque modification doit être suivie pas à pas sur github à l'aide des commits détaillant les étapes et d'une pull request accessible au chef de projet.

B- Les tests unitaires et fonctionnels

Trouver les données des tests unitaires et fonctionnels dans le dossier tests.
Séparation en deux dossiers:

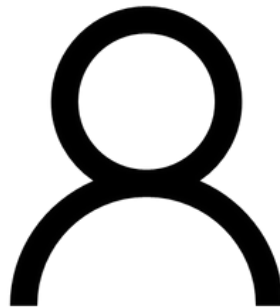
- tests unitaires: concernent les tests effectués des entités et de leur retour répondant aux critères donnés.
- tests fonctionnels: ces test concernent plus strictement les tests correspndant au retour des fonctions

05. L'authentification et le ROLE USER

Nous allons présenter ici la mise en place de l'authentification et des rôles au sein du projet: de l'enregistrement de l'utilisateur avec un rôle attribué aux accès suivant le rôle enregistré (ou anonymous).

Voici les rôles de chaque user et la manière dont ils sont organisés dans le fichier security.yml.

Les rôles



Anonyme	Rôle user	Rôle admin
Ses accès: <ul style="list-style-type: none">• liste des tâches• page login• Validation ou non d'une tâche	Ses accès: <ul style="list-style-type: none">• liste des tâches• page login• Validation ou non d'une tâche• création d'une tâche• suppression de ses propres tâches• modification d'une tâche	Ses accès: <ul style="list-style-type: none">• liste des tâches• page login• Validation ou non d'une tâche• création d'une tâche• suppression de ses propres tâches• modification d'une tâche• gestion utilisateurs (ajout/liste/modification)
Security.yml		
IS_AUTHENTICATED _ANONYMOUSLY	ROLE_USER	ROLE_ADMIN

Pour administrer vos utilisateurs, commencez par créer un utilisateur dans le formulaire et lui donner un rôle (rôle multiple possible). Le rôle est enregistré en base de données pour chaque utilisateur. Cette administration préalable est déjà implémentée lors de vos migrations.

Pour vérifier les accès de chaque type d'utilisateurs, se rendre dans le fichier security.yml / acces_control et configurer les paths(accès). Vous pouvez ajouter plusieurs rôles.

Ici, 3 rôles sont définis avec la configuration des accès définis ci-dessus.

Seul l'administrateur peut modifier cet accès pour un utilisateur

L'authentification est définie dans le security.yml/form_login: login_path: login
check_path: login_check

IL faudra donc se loguer pour être authentifié (sinon considéré comme anonymously).



**MARINE JOURDAN-
DEVELOPPEUR PHP-SYMFONY**

AUDIT DE CODE ET DE PERFORMANCE



ANALYSE SYMFONY INSIGHT

Analyse effectuée via github et l'outil d'analyse en ligne symfony insight.

Effectué durant les différentes évolutions du code pour correspondre aux normes symfony et permettre un partage du projet efficace avec un socle de normes communes

Évolutions symfonyInsight

- *Evolution de symfony : actuellement en 3.4 évolué en 5.4*
- *Supprimer les commentaires de code*
- *Ajouter le favicon personnalisé*
- *Typier les entités*
- *effectuer des migrations*

Utilisation de PHP CS-FIXER

PHP CS Fixer est un outil permettant de vérifier et corriger le formatage du code PHP selon le code style défini dans la configuration du projet.

Le style code est une convention qui définit comment doit être écrit le code. Par exemple l'emplacement des accolades des structures de contrôle, des classes, des fonctions, les espaces autour d'un signe égal, etc.

Cela permet de faciliter la lecture du code pour tout le monde.

Utilisation de symfony profiler

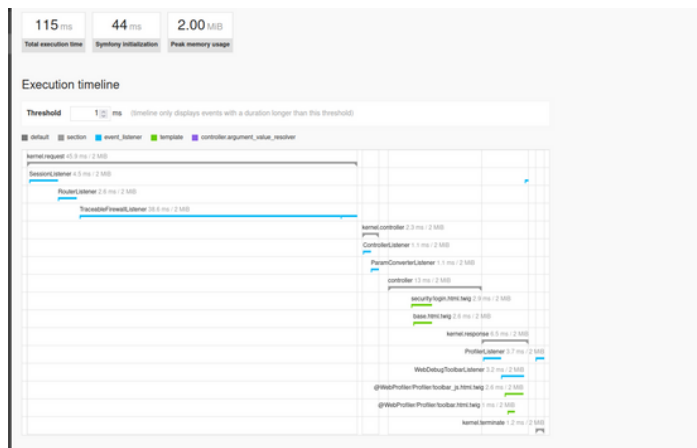
- Faire un point de l'existant
- Obtenir des solutions pérennes à l'amélioration de la performance
- Corriger les anomalies

Audit de code vise à améliorer la qualité de l'application, augmenter ses performances, accélérer l'expérience utilisateur, augmenter la sécurité de l'application, identifier et anticiper les problèmes de sécurité et de performance, permettre au code d'être réutilisable pour tous selon les normes en vigueur.

Il permet de collecter de nombreuses informations sur votre application durant son exécution : pour chaque requête utilisateur, le Web Profiler va créer un profil et l'application web fournit une interface soignée pour parcourir les informations de ce profil. Il permet de collecter de nombreuses informations sur votre application durant son exécution : pour chaque requête utilisateur, le Web Profiler va créer un profil et l'application web fournit une interface soignée pour parcourir les informations de ce profil.

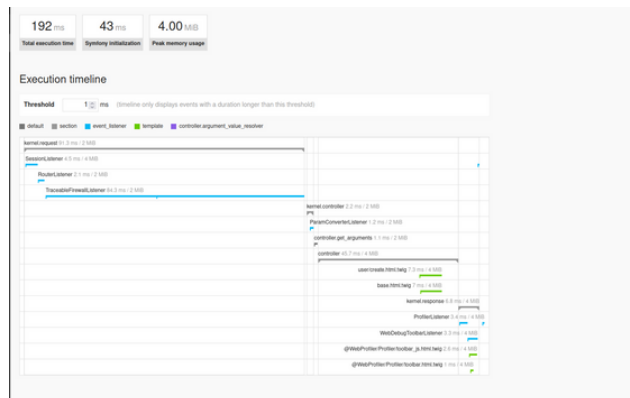
- LE TEMPS DE RENDU DE LA PAGE;
- LA CONSOMMATION DE MÉMOIRE ;
- LE NOMBRE D'APPELS AU CACHE ;
- LE TEMPS DE RENDU DES BLOCS TWIG...

Page login (symfony 5.4)



réponse du kernel divisée par 2-
temps du controller et des blocs twig inchangés malgré les ajouts de certaines propriétés des entités.
Request kernel: temps plus long (chargement de plus de bundles et de dépendances)
Plus de ressources malgré une évolution de l'application-
la mémoire utilisée est elle énormément amoindrie entre les deux versions de symfony. Ce qui permettra de supporter plus de charge et plus d'utilisateurs.

Page create (symfony 5.4)

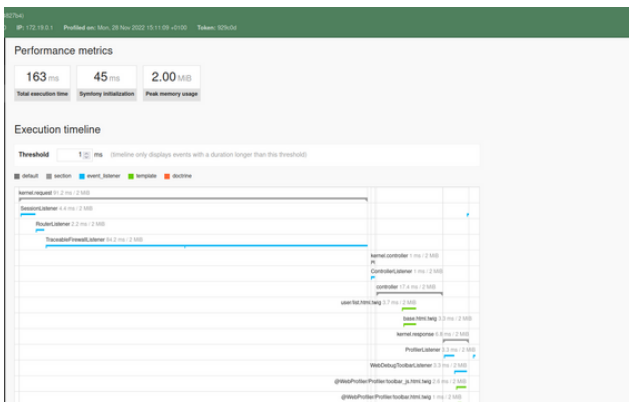


Réponse du kernel réduite
la mémoire utilisée est elle énormément amoindrie entre les deux versions de symfony.
On constate aussi que l'affichage twig prend autant de temps mais beaucoup moins de mémoire tout comme le controleur.

Page users (symfony 3.4)

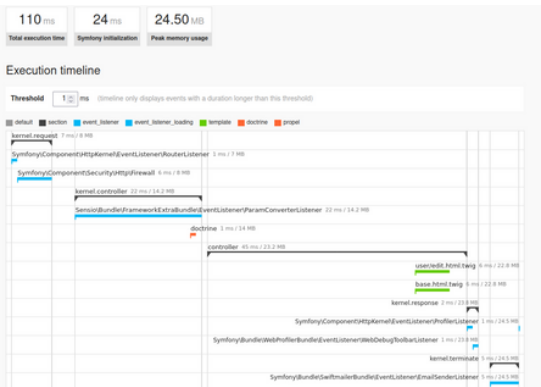


Page users (symfony 5.4)

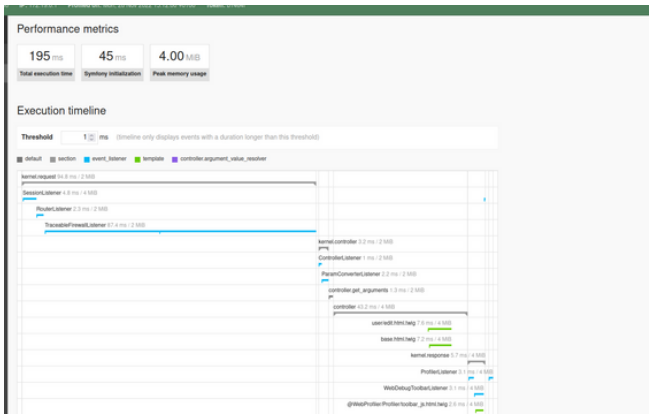


Réponse du kernel réduite
la mémoire utilisée est elle énormément amoindrie entre les deux versions de symfony.
Cependant, ici le controller en charge de l'affichage des users est plus rapide

Page edit/user (symfony 3.4)



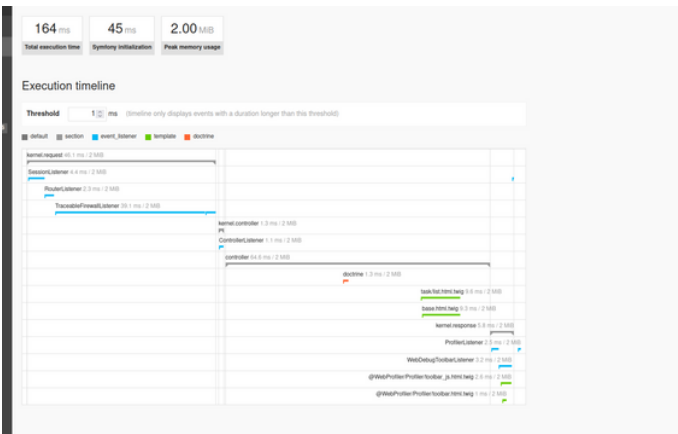
Page edit/user (symfony 5.4)



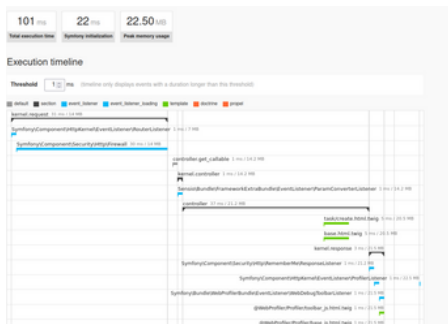
Page tasks (symfony 3.4)



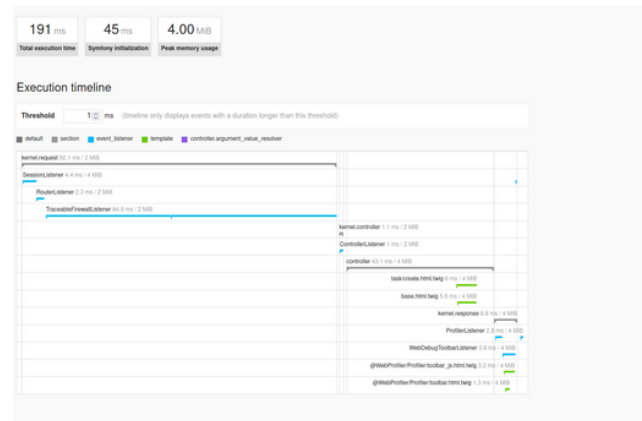
Page tasks (symfony 5.4)



Page task/create (symfony 3.4)



Page task/create (symfony 5.4)



Conclusion du passage de symfony 3.4 à 5.4 : Une évolution notable dans la kernel response, la mémoire utilisée et le chargement des données doctrine.





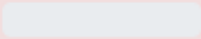

La version 5 et les bundles installés suite à la migration ont ralenti le temps de chargement du kernel est des dépendances.

L'application a grossi, le système de gestion des utilisateurs a été mis en place ce qui donne globalement un temps de chargement identique ou plus long de l'affichage plus précisément.






Cependant, la mémoire a été énormément réduite ce qui permet d'avoir un site beaucoup plus léger.

DÉVELOPPEMENT POSSIBLE:
OPTIMISER LE CHARGEUR AUTOMATIQUE DE COMPOSER EN MODE PROD

RAPPORT DE COUVERTURE DE CODE SYMFONY 5.4 (APRÈS MIGRATION)



		Lines
Total		76.13%
📁 Controller		76.92%
📁 Entity		90.20%
📁 Form		100.00%
📁 Migrations		0.00%
📁 Security		64.00%
📄 AppBundle.php		n/a

TAUX DE COUVERTURE GLOBALE:
76,13%



		Lines	
Total		87.69%	57 / 65
📄 DefaultController.php		100.00%	1 / 1
📄 SecurityController.php		66.67%	4 / 6
📄 TaskController.php		82.86%	29 / 35
📄 UserController.php		100.00%	23 / 23

Legend

LES CONTROLLEURS


		Lines	
Total	<div><div></div></div>	90.20%	46 / 51
 Task.php	<div><div></div></div>	85.00%	17 / 20
 User.php	<div><div></div></div>	93.55%	29 / 31

LES ENTITÉS

		Lines	
Total	<div><div></div></div>	100.00%	6 / 6
 TaskType.php	<div><div></div></div>	100.00%	2 / 2
 UserType.php	<div><div></div></div>	100.00%	4 / 4

Legend

LES FORMULAIRES

		Lines	
Total	<div><div></div></div>	64.00%	16 / 25
 LoginFormAuthenticator.php	<div><div></div></div>	64.00%	16 / 25

Legend

LES POINTS DE SECURITÉ (LOGIN)