AMÉLIOREZ UNE APPLICATION EXISTANTE DE TODO & CO

OpenClassRooms – Projet P8

PLAN DE LA PRÉSENTATION

- Première partie
  1. Présentation du contexte du projet
  2. Analyse du besoin
  3. Organisation du projet

- Deuxième partie
  1. Démonstration de l'application

- Troisième partie
  1. Versioning du projet
  2. Architecture du projet
  3. Librairies employées
  4. Exemple d'une pull request sur GitHub
  5. Gestion de la sécurité
  6. Tests unitaires et fonctionnels
  7. Documentation technique
  8. Audits

PREMIÈRE PARTIE

Présentation du contexte du projet, analyse du besoin & organisation du projet

PRÉSENTATION DU CONTEXTE DU PROJET

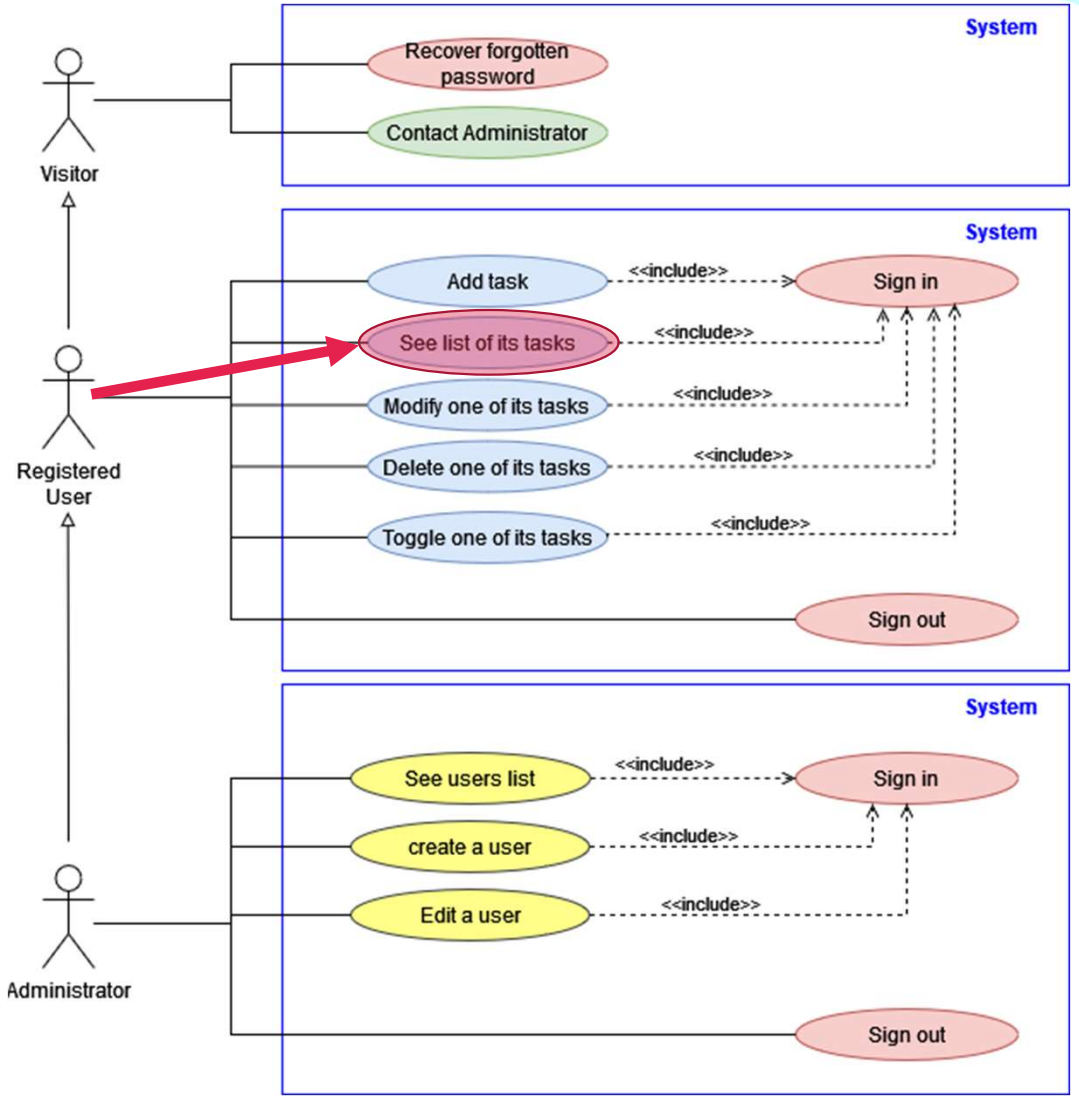
- Vous venez d'intégrer une startup dont le cœur de métier est une application permettant de gérer ses tâches quotidiennes.
- L'entreprise vient tout juste d'être montée, et l'application a dû être développée à toute vitesse pour permettre de montrer à de potentiels investisseurs que le concept est viable (on parle de Minimum Viable Product ou MVP).
- Le choix du développeur précédent a été d'utiliser le framework PHP Symfony.
- **ToDo & Co** a réussi à lever des fonds pour permettre le développement de l'entreprise et surtout de l'application.
- **But** : améliorer la qualité de l'application.

ANALYSE DU BESOIN

- **ToDo & Co** vous donne la charge des tâches suivantes :
  1. Faire un état des lieux de la dette technique de l'application.
  2. Corriger des anomalies :
     - Une tâche doit être attachée à un utilisateur
     - Choisir un rôle pour un utilisateur
  3. Implémenter de nouvelles fonctionnalités :
     - Gérer les autorisations
     - Implémenter des tests automatisés(Fournir un jeu de données et un rapport de couverture des tests.)
  4. Etablir une documentation technique
  5. Etablir un audit de qualité du code et des performance de l'application

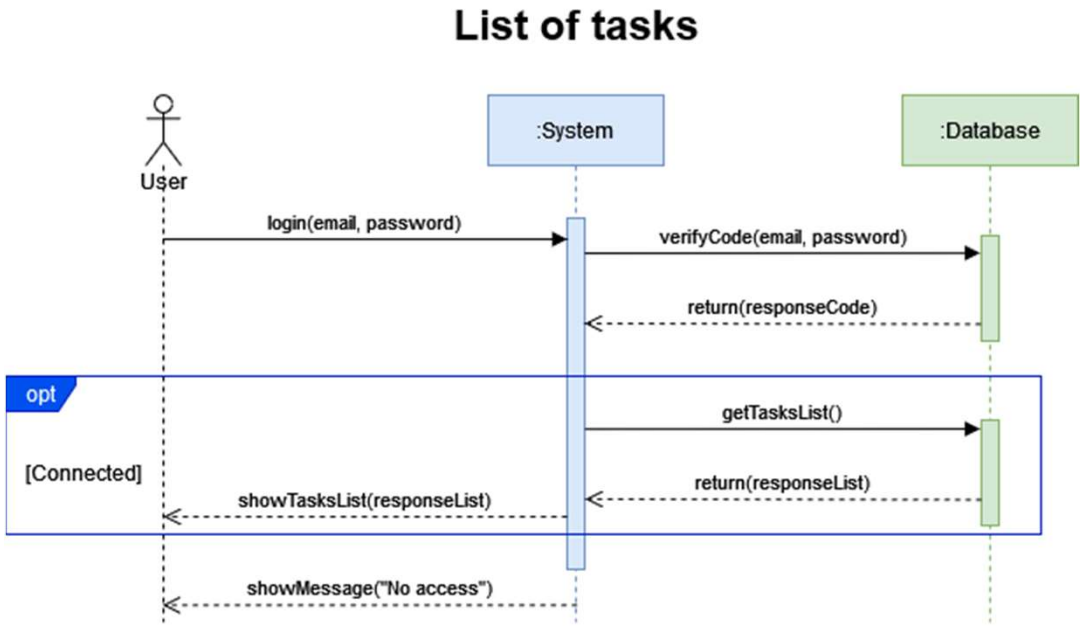
ANALYSE DU BESOIN CAS DE L'OBTENTION DE LA LISTE DES TACHES

- Diagramme d'utilisation



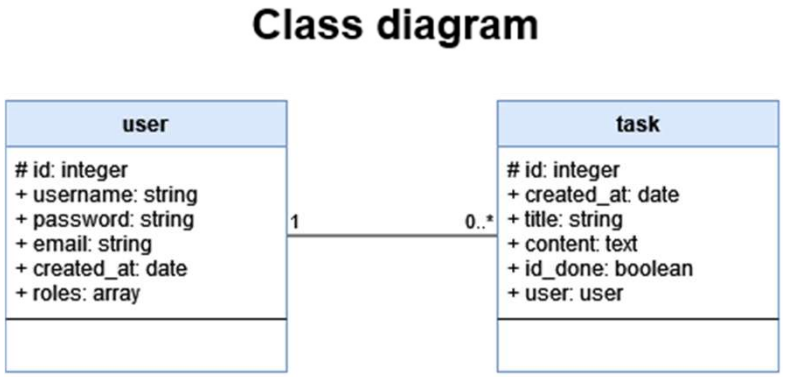
ANALYSE DU BESOIN
CAS DE L'OBTENTION DE LA LISTE DES TACHES

- Diagramme de séquence



ANALYSE DU BESOIN
CAS DE L'OBTENTION DE LA LISTE DES TACHES

- Diagramme de classe



ORGANISATION DU PROJET

- Création du dépôt sur GitHub : https://github.com/Urza45/todolist

- Création de label sur le dépôt GitHub afin de déterminer le poids des issues qui seront créées :

Estimation-1	Very light functionality (<0.5 day)
Estimation-2	Moderate functionality (> 0.5 day; <2 days)
Estimation-3	Difficult functionality (> 2 days; <5 days)

ORGANISATION DU PROJET

- Suivi du temps réel passé sur un fichier Excel : exemple à un instant t

DEUXIÈME PARTIE

Démonstration de l'application

DÉMONSTRATION DE L'APPLICATION

- Une version est disponible en ligne : https://p8.urza-web.fr

TROISIÈME PARTIE

Description technique du projet

VERSIONING DU PROJET

* Création depuis la branche « main »
** Pull request de cette branche sur la branche « main »

Lien du dépôt sur GitHub : https://github.com/Urza45/todolist

ARCHITECTURE DU PROJET

L'architecture du projet a évoluée de Symfony 3 à Symfony 5.

Le schéma ci-contre indique l 'évolution des répertoires entre les deux versions de Symfony.

LA sécurité (en dehors des fichiers de paramétrage) est gérée dans le répertoire **src/Security** (Non-indiqué sur le schéma)

Les tests se situent dans le répertoire **tests**.

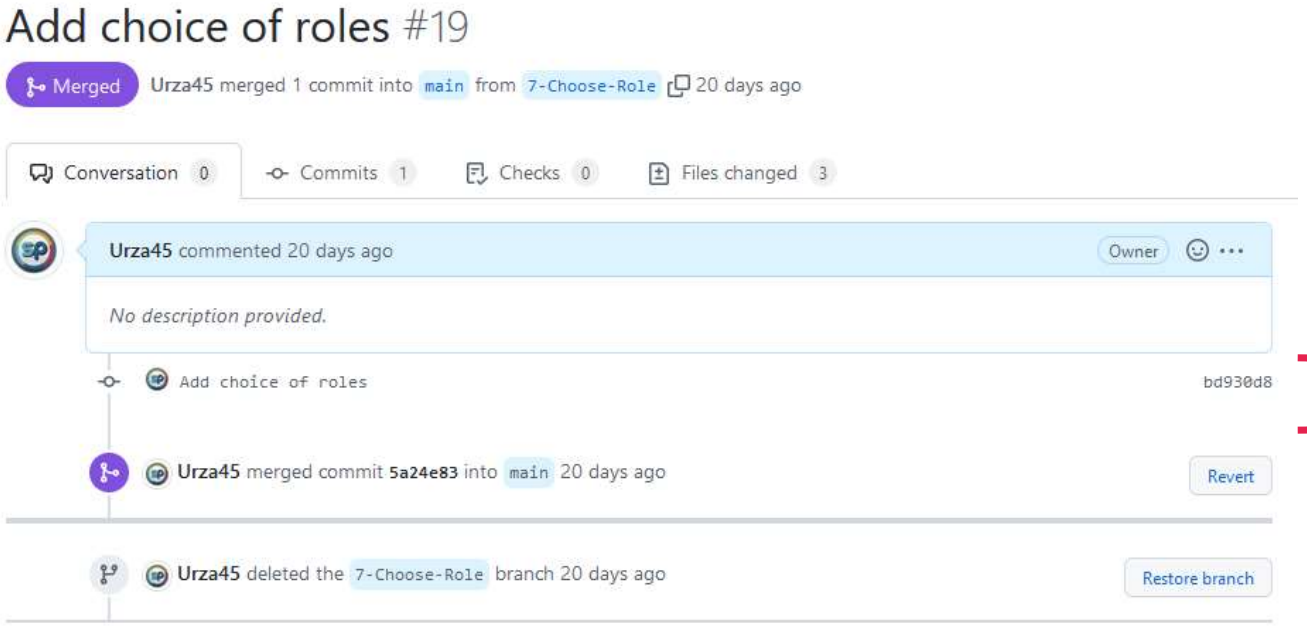
Evolution de l'architecture de Symfony

LIBRAIRIES EMPLOYÉES

- Les librairies employées ont été installées avec Composer (https://getcomposer.org/)
    - "symfony/webpack-encore-bundle": "^1.14" pour gérer le javascript et le CSS

  Et en mode dev uniquement :
    - "dama/doctrine-test-bundle": "^7.1" pour restaurer la base de données lors des tests.
    - "doctrine/doctrine-fixtures-bundle": "^3.4" pour les données de tests
    - "phpunit/phpunit": "^9.5« pour les tests
- Les librairies nécessaires dans le futur de l'application devront être également installées avec Composer. Les librairies installables avec Composer sont listées sur le site https://packagist.org
- Pour les tests, la librairie PHPUnit est installée. Pour avoir un rendu HTML de la couverture des tests, vous devrez installer xDebug sur votre serveur (https://xdebug.org/docs/install)

EXEMPLE D'UNE PULL REQUEST



Liste des commits

Vérification et merge de la branche

Suppression de la branche devenue inutile

GESTION DE LA SÉCURITÉ

- La sécurité est gérée à plusieurs niveaux :

1. Au niveau des routes : **config\packages\security.yaml**
2. Au niveau des authentifications : **src\Security\AppAuthenticator.php**
3. Au niveau des autorisations : **src\Security\Voter\TaskVoter.php**

TESTS UNITAIRES ET FONCTIONNELS

- Les tests ont été mis en place avec le bundle PHPUnit et sont gérés dans le répertoire **tests**
- Le bundle **DAMA\DoctrineTestBundle\PHPUnit\PHPUnitExtension a été installé.** Sa fonction est d'utiliser les transactions Doctrine pour permettre à chaque test d'interagir avec une base de données non modifiée. Il commence une transaction de base de données avant chaque test et l'annule automatiquement une fois le test terminé pour annuler toutes les modifications.
- Un rapport de couverture de test se situe dans le répertoire suivant : **public\test-coverage**

DOCUMENTATION TECHNIQUE

- La documentation technique est fournie dans le fichie

Pillay_Serge_05_Documentation technique_052022.pdf

- Sommaire :

AUDITS

- Deux types d'audits ont été réalisés :
  - Audits de qualités :
    - Analyse avec Codacy
    - Analyse avec Code Climate
    - Analyse avec PHPCode Sniffer
  - Audits de performance
    - Analyse avec BackFire

- Ces audits sont détaillés dans le fichier

Pillay_Serge_06_Rapport d'audit_052022.pdf