

Projet

Développement et architecture web

L'objectif du projet est de réaliser une application todo list sur une architecture distribuée.

Il y aura deux rendus:

- Le premier rendu: 23/11/2020
- Le second rendu: 12/01/2020

Le projet est à faire en groupe de deux. Dans vos `composer.json`, pensez à bien renseigner les DEUX auteurs du groupe, avec vos noms, prénoms et adresses mail.

Si des manipulations spécifiques sont nécessaires pour faire fonctionner vos projets, vous pouvez adjoindre une documentation sous la forme d'un `readme.md` à la racine. MAIS je m'attends à pouvoir faire fonctionner votre projet en faisant un `docker-compose up`

1.Premier rendu: une application MVC

L'objectif du premier rendu est de réaliser le site web "ToDo List". Il comptera pour 10 points sur la note globale du projet.

Cette application devra respecter les contraintes suivantes :

- Langage de programmation: PHP 7.4 / XHTML / CSS / JS
- Utilisation de `composer` pour les dépendances PHP
- Utilisation du Framework Slim 4
- Accès à la base de données MySQL avec PDO
- Utilisation du framework d'injection de dépendances PHP-DI
- Utilisation de variables d'environnement pour la configuration de l'app
- Utilisation de NPM pour les dépendances node
- Utilisation du framework Bootstrap 4
- Site web responsive design utilisable sur 3 types de device: smartphone, tablette et desktop
- Utilisation du moteur de template Twig
- Utilisation du pré-processeur de feuille de style Sass
- Utilisation du framework javascript jQuery
- Mise en place d'une structure MVC (modèle - vue - contrôleur)
- Utilisation de docker et de docker compose pour l'infrastructure

Fonctionnalités attendues :

- Fonctionnalité de login et mise en session de l'utilisateur.

Une fois connecté (et seulement une fois connecté), les fonctionnalités suivantes doivent être mise à disposition de l'utilisateur

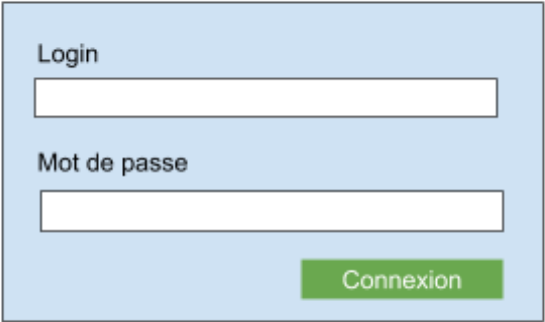
- Listing des tâches
- Filtrage des tâches par utilisateur
- Filtrage des tâches par mots clés
- Affichage du détail d'une tâche, avec la liste des commentaires associés
- Ajout d'une nouvelle tâche (voir champs dans le fichier SQL TP n°1)
- Ajout d'un nouveau commentaire à une tâche
- Fonctionnalité de déconnexion

Cela représente en tout 6 use cases (connexion, puis liste / recherche de tâches, détail d'une tâche, ajout d'une tâche, ajout d'un commentaire et déconnexion)

Une contrainte supplémentaire est de réaliser les deux fonctionnalité d'ajout de tâches et de commentaires en ajax (Javascript asynchrone) avec jQuery. La fonctionnalité d'ajout doit être réalisée au travers d'une fenêtre modale de Bootstrap 4.

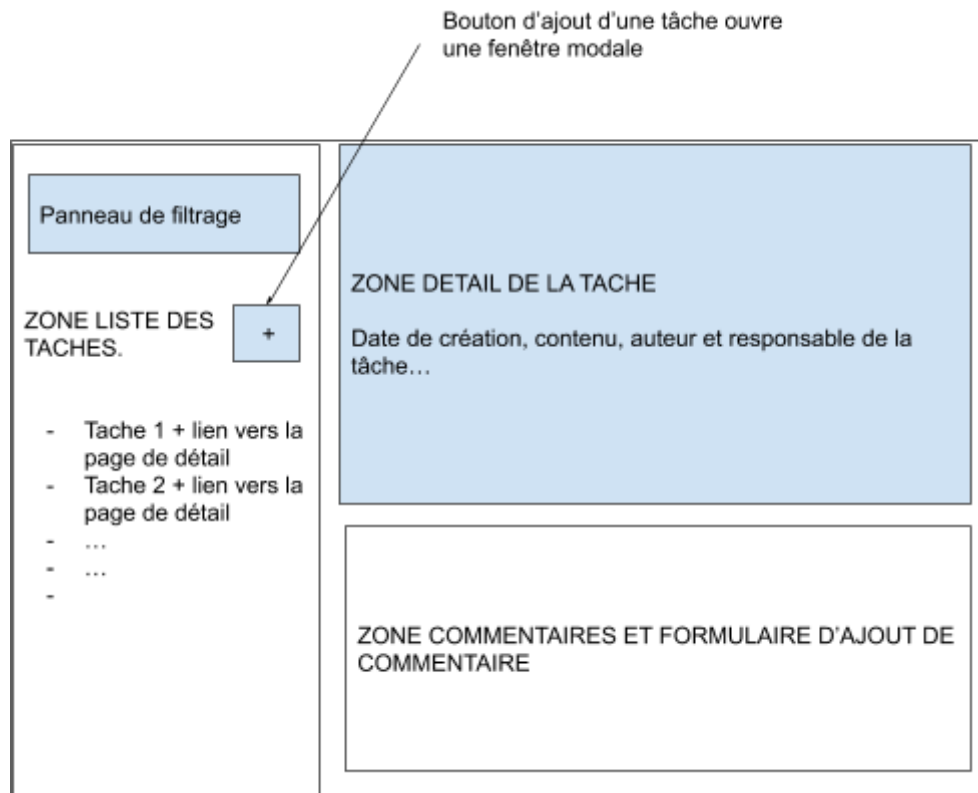
L'expérience utilisateur et l'ergonomie est à votre discrétion, cependant voici un certain nombre d'idées que vous pourrez utiliser pour réaliser votre projet.

Pour le login



The diagram shows a light blue rectangular box representing a login form. Inside the box, at the top left, is the label "Login". Below it is a white rectangular input field. Further down is the label "Mot de passe". Below that is another white rectangular input field. In the bottom right corner of the blue box is a green rectangular button with the white text "Connexion". The entire login form box is centered within a larger white rectangular frame.

Pour la fenêtre une fois connecté



La notation se fera de la manière suivante :

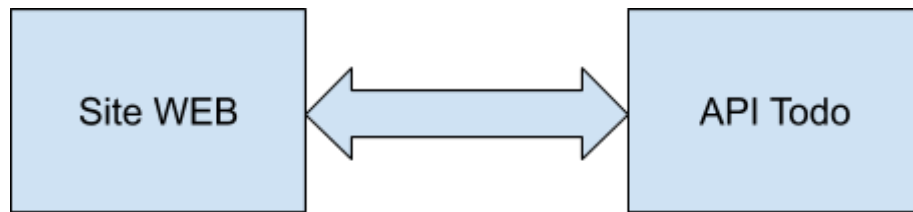
- 3 points pour les use cases
- 3 points liés au respect des consignes
- 4 points sur les aspects lisibilité, cohérence, sécurité, performances et tests

2.Second rendu: une architecture distribuée

Ce rendu comptera lui aussi pour 10 points dans la note du projet.

Le business souhaite que l'équipe de développement publie une application mobile native basée sur les mêmes fonctionnalités que le site web. Nous ne développerons pas l'application mobile mais devons mettre à disposition les données aux équipes chargées de leur développement.

Et donc, l'objectif est dans un premier temps de corriger le site web pour qu'il utilise des repositories et que les requêtes SQL ne soient pas exécutées dans les contrôleurs (ni dans les modèles). Ensuite, il s'agira de développer une API REST, et de récupérer le listing de tâches via cette API.



L'API todo devra respecter les contraintes suivantes:

- Langage de programmation: PHP 7.4 / XHTML / CSS / JS
- Utilisation de composer pour les dépendances PHP
- Utilisation du Framework Slim 4
- Accès à la base de données MySQL avec PDO
- Utilisation du framework d'injection de dépendances PHP-DI
- Utilisation de variables d'environnement pour la configuration de l'app
- Ajout du service "API" dans docker-compose
- Authentification à l'API par token header non chiffré
- Format d'échanges clients / serveur JSON
- Architecture hexagonale

Fonctionnalités attendues :

- Endpoint de listing des tâches

Concrètement, l'objectif est de mettre en place dans un premier temps des repositories et une architecture hexagonale partout où il y a un accès à la base de données sur votre site web. Typiquement, vos contrôleurs doivent avoir en injection de dépendance ces repositories.

Pour les tâches, il y aura deux interfaces distinctes: l'une pour la récupération du listing de tâches, qui passera donc concrètement par des appels http sur le serveur API, et l'autre qui concerne toutes les autres opérations sur les tâches (création, marquer comme terminé etc.) et qui continuera d'utiliser PDO pour accéder à la base.

Sur le site web : vous travaillerez dans une branche et ferez un merge request avec vos modifications. Sur l'API, vous créerez un projet en partant de zéro (nouvelle repository sur gitlab), vous bootstraperez le projet avec les contraintes mentionnées plus haut et vous développerez le use case de listing de todo (en incluant les différents filtres). Puis, sur le site web, vous appelez cette api pour lister et construire vos modèles de todo.

Le site web et l'API todo partagent la même base de données.

La notation se fera sur la base des corrections que vous aurez apporté au premier rendu:

- Modification du site web et bon usage des repositories: 3 points
- Développement de l'API todo: 5 points
- Prise en compte des remarques et corrections relatives au premier rendu: 2 points