

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: 16/11/2020
- 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

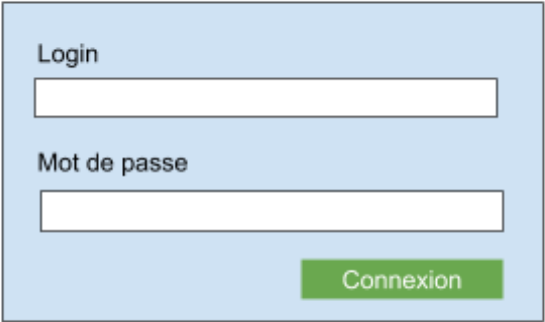
- Listing des tâches
- Filtrage des tâches par utilisateur
- Filtrage des tâches par mots clés
- Affichage du détail d'une tâche, avec la liste des commentaires associés
- Ajout d'une nouvelle tâche (voir champs dans le fichier SQL TP n°1)
- Ajout d'un nouveau commentaire à une tâche
- Fonctionnalité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

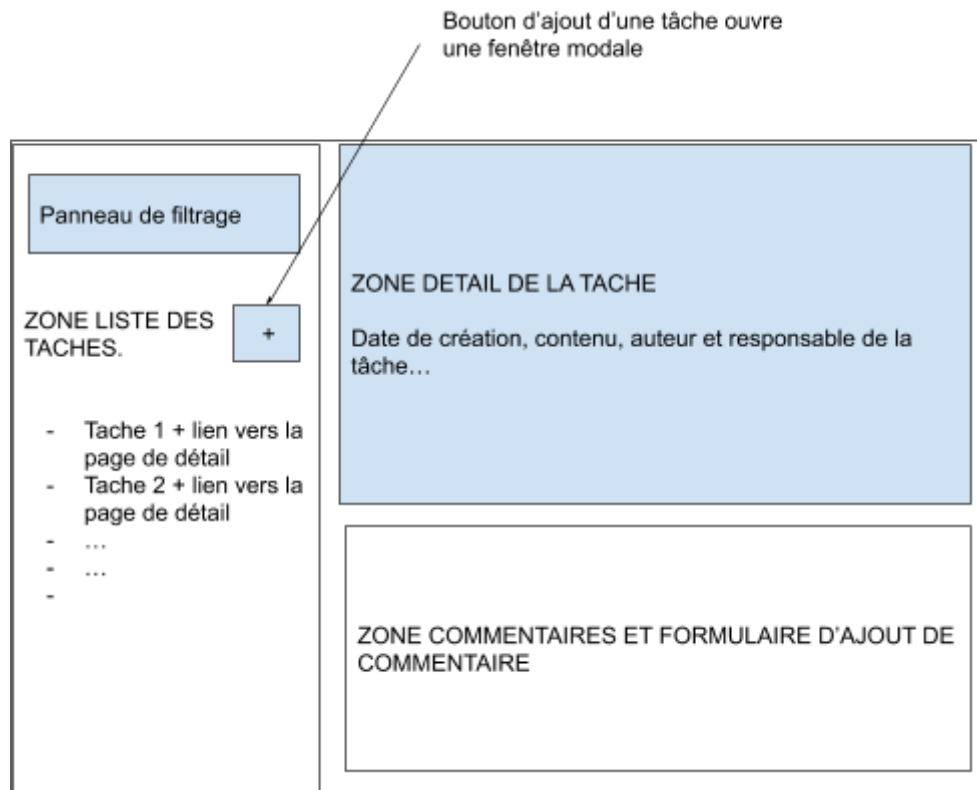
L'expérience utilisateur et l'ergonomie est à votre discrétion, cependant voici un certain nombre d'idées que vous pourrez utiliser pour réaliser votre projet.

Pour le login



The diagram shows a login form with a light blue background. It contains two input fields: the first is labeled 'Login' and the second is labeled 'Mot de passe'. Below the second input field is a green button with the text 'Connexion'.

Pour la fenêtre une fois connecté



La notation se fera de la manière suivante :

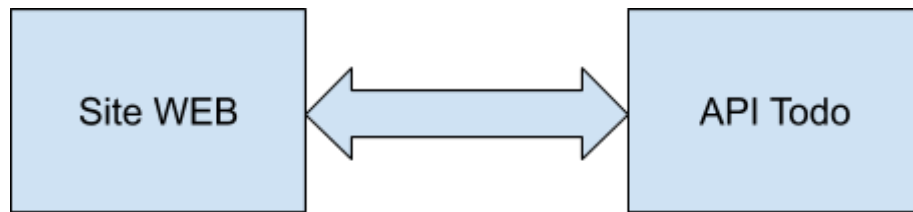
- 3 points pour les use cases
- 3 points liés au respect des consignes
- 4 points sur les aspects lisibilité, cohérence, sécurité, performances et tests

2.[WIP] 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 en charge de leurs développement.

Et donc, l'objectif est de développer une API REST ainsi qu'un SDK qui simplifiera les appels à cette API. La couche d'accès aux données de l'application web du premier rendu sera remplacée par un accès API. Le site web utilisera le SDK pour appeler l'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

Fonctionnalités attendues :

- Endpoint d'authentification utilisateur (il faudra simplement retourner les données de l'utilisateur qui tente de se connecter)
- Endpoint de listing des tâches
- Endpoint de détail d'une tâche
- Endpoint de listing des commentaires
- Endpoint d'ajout d'une tâche
- Endpoint d'ajout d'un commentaire

Il est attendu un SDK, qui sera développée dans le répertoire sdk (composer.json à mettre à jour) dans le projet "web", puisque vous ne publiez pas votre SDK sur packagist. Les fonctionnalités du SDK doivent permettre de traiter les 6 use cases détaillés ci-dessus.

La notation se fera sur la même base que l'appli WEB :

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