E- CAHIER DES CLAUSES TECHNIQUES ET DÉTAILLÉES

1. **Technologies et compatibilité**

Pour la mise en place du site et de l’application mobile nous utiliserons différentes technologies aussi bien pour le front-end que pour le back-end.  
Technologies pour le front-end :

* Angular JS

Le choix s’est porté sur Angular JS car c’est une technologie que nous maîtrisons bien. Elle est rapide et son fonctionnement nous permet d’offrir une expérience fluide et agréable pour nos visiteur : PWA ou progressive Web App.

Technologies pour le back-end :

* API Platform et Symfony

Nous utiliserons principalement API Platform et Symfony pour le backend du projet car ce sont des technologies assez solides, rapides et fiables pour mettre en place un backend API et une documentation prêt à l’usage. De plus nous maîtrisons tous également le php et son framework symfony pour la mise en place de site web complet.

* MySQL

Nous utiliserons MySQL car c’est une base de données connue et adaptée pour la taille de notre projet. Étant une base de donnée sql, il est de mise d’utiliser des reqêtes sql pour discuter avec la base de données. Cependant, avec l’ORM Doctrine de symfony, cet aspect du code devient abstrait pour le développeur. L’ORM doctrine de Symfony se connecte facilement à la base de donnée mysql et nous pouvons visualiser directement les modifications dans l’espace phpMyadmin.

1. **Logiciels et outils utilisés**

Visual studio code : Pour le développement  
Google pour le déboggage

Postman pour les test de l’API

1. **Test applicatifs**

Pour assurer une qualité élevée et une stabilité de l'application web et mobile. A développer chaque type de test effectué répond à des objectifs spécifiques et permet de couvrir différents aspects du développement, garantissant que l'application répond aux attentes des utilisateurs et aux exigences de performance. On a entre autre :

1. Tests Unitaires

Objectif : Vérifier que chaque unité de code fonctionne comme prévu.

Justification :

• Les tests unitaires sont fondamentaux pour assurer que les composants individuels de l'application (fonctions, méthodes, classes) fonctionnent correctement.

• Ils facilitent la détection et la correction des bugs dès les premières phases de développement, ce qui réduit les coûts de maintenance.

Outils Utilisés :

• Front-end (AngularJS) : Jasmine et Karma.

• Jasmine est un framework de test comportemental pour JavaScript.

• Karma est un test runner pour JavaScript, conçu pour exécuter des tests dans différents navigateurs.

• Back-end (API Platform/Symfony) : PHPUnit.

• PHPUnit est un framework de test unitaire pour PHP.

2. Tests d’Intégration

Objectif : Vérifier que les différentes unités de code fonctionnent ensemble correctement.

Justification :

• Les tests d'intégration permettent de s'assurer que les composants ou modules intégrés fonctionnent comme attendu lorsqu'ils sont combinés.

• Ils sont cruciaux pour détecter les problèmes d’interaction entre les composants, qui ne peuvent pas être identifiés par des tests unitaires.

Outils Utilisés :

• Front-end : Protractor.

• Protractor est un framework de test d'intégration end-to-end pour les applications Angular.

• Back-end : PHPUnit avec des outils de mocking comme Prophecy.

• Prophecy est une bibliothèque de prédiction et de simulation pour les tests PHP, souvent utilisée avec PHPUnit.

3. Tests Fonctionnels

Objectif : Vérifier que le système répond aux exigences fonctionnelles spécifiées.

Justification :

• Les tests fonctionnels sont essentiels pour valider que l'application répond aux spécifications fonctionnelles et aux besoins des utilisateurs.

• Ils simulent des scénarios réels d'utilisation pour vérifier que le système fonctionne dans des conditions réelles.

Outils Utilisés :

• Front-end : Jasmine avec Protractor pour des tests end-to-end.

• Les tests end-to-end avec Protractor permettent de simuler l'interaction des utilisateurs avec l'application.

• Back-end : Behat.

• Behat est un framework de test fonctionnel pour PHP, basé sur l'approche BDD (Behavior Driven Development).

4. Tests de Performance

Objectif : Vérifier que l'application répond aux exigences de performance sous différentes charges.

Justification :

• Les tests de performance permettent de s'assurer que l'application est réactive, rapide et capable de gérer une charge utilisateur élevée.

• Ils aident à identifier les goulots d'étranglement et à améliorer la performance globale de l'application.

Outils Utilisés :

• Front-end : Lighthouse et JMeter.

• Lighthouse est un outil automatisé d'audit de performance, utilisé pour analyser les performances web.

• JMeter est un outil de test de charge open-source pour mesurer les performances de divers services, avec une extension pour tester les applications web.

• Back-end : Apache JMeter et Blackfire.

• Blackfire est un outil de profilage de performance pour les applications PHP.

Outils et Environnements

CI/CD Pipelines :

• GitLab CI/CD ou Jenkins pour automatiser l'exécution des tests et le déploiement continu.

Suivi et Reporting des Tests :

• SonarQube pour l'analyse continue de la qualité du code.

• Allure Report pour la génération de rapports de tests détaillés.

1. **Hébergement et non de domaine**

Hébergement

Nous avons décidé d’héberger le site web et l’application mobile sur firebase qui nous offre des solutions et des services d’administrations adéquats pour la configuration et la maintenance de nos différents projets.

Nom de domaine

Le nom de domaine utilisé pour le site web est : atipickhouse.fr

Il coûtera 5€ HT/an pour 1 an et 11€HT/an

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

Nom de l’application mobile : atypickhouse