ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Cahier de charges : SOZ - Blog Collaboratif

OUMMANY Younes – SADIK Abderrahim – ZBAIR Aya

Table des matières

1. Présentation du projet

1.1 Contexte

Ce projet vise à développer un blog collaboratif complet, comprenant des sections distinctes pour les administrateurs et les utilisateurs. Le blog permettra aux utilisateurs de :

- Créer et publier des articles
- Modifier et supprimer leurs articles
- Commenter les articles des autres utilisateurs
- Gérer leur profil utilisateur

Les administrateurs disposeront de fonctionnalités supplémentaires pour :

- Gérer les utilisateurs (création, modification, suppression, attribution de rôles)
- Modérer le contenu (validation des articles, suppression des articles et commentaires abusifs)

Le blog sera accessible via un navigateur web et une application mobile native développée pour Android.

1.2 Objectifs

Les objectifs du projet sont les suivants :

- Permettre aux utilisateurs de partager leurs connaissances et leurs idées
- Favoriser l'interaction entre les utilisateurs
- Offrir aux administrateurs une interface pour gérer efficacement le contenu et les utilisateurs du blog

1.3 Description de l'existant

Il n'existe pas de solution logicielle interne répondant à ces besoins spécifiques.

1.4 Critères d'acceptabilité du produit

Le produit sera considéré comme acceptable si les critères suivants sont remplis :

- Le blog est accessible via un navigateur web et une application mobile native Android.
- Les utilisateurs peuvent créer, modifier et supprimer des articles, commenter des articles et gérer leur profil.
- Les administrateurs peuvent gérer les utilisateurs, modérer le contenu et valider les articles avant publication.
- Le blog est performant, sécurisé.

2. Expression des besoins

2.1 Besoins fonctionnels

2.1.1 Fonctionnalités communes

- **Gestion des articles**
  - Créer un article
  - Modifier un article
  - Supprimer un article
  - Consulter un article
  - Rechercher des articles
- **Gestion des commentaires**
  - Ajouter un commentaire
  - Modifier un commentaire
  - Supprimer un commentaire (pour les utilisateurs ayant créé le commentaire ou disposant des droits de modération)

2.1.2 Fonctionnalités spécifiques aux utilisateurs

- Créer un compte utilisateur
- Modifier un compte utilisateur
- Supprimer un compte utilisateur
- Se connecter au blog
- Se déconnecter du blog

2.1.3 Fonctionnalités spécifiques aux administrateurs

- **Gestion des utilisateurs**
  - Créer un utilisateur
  - Modifier un utilisateur
  - Supprimer un utilisateur
  - Attribuer des rôles aux utilisateurs (administrateur, utilisateur)
- **Gestion des articles**
  - Valider les articles publiés.
  - Supprimer les articles signalés comme abusifs
- **Gestion des commentaires**

- Supprimer les commentaires signalés comme abusifs

2.2 Besoins non fonctionnels

- **Performance**
  - Le blog doit être accessible rapidement et gérer un grand nombre d'utilisateurs et d'articles.
- **Sécurité**
  - Les données des utilisateurs doivent être protégées.
- **Compatibilité**
  - Le blog doit être compatible avec les navigateurs web les plus courants.
  - L'application mobile doit être compatible avec les smartphones et tablettes Android récents.
- **Planification et Conception**
  - Document de spécifications fonctionnelles et techniques décrivant le projet, les fonctionnalités prévues, et l'architecture système proposée.
  - Les maquettes pour les principaux (user stories)
  - Un backlog de produit initial, comprenant des user stories
  - Un plan des sprints
- **Gestion du code**
  - Structure de projet.
  - Un repository Git (par exemple sur GitHub ou GitLab) contenant le code source, avec des conventions de nommage pour les branches et des commits bien documentés.
  - Intégration d'un fichier README.md décrivant le projet, la façon de le construire et de l'exécuter localement.
- **Intégration Continue (CI)**
  - Configuration d'un serveur CI (par exemple Azure devops, Jenkins, GitLab CI/CD, GitHub Actions) pour automatiser la compilation, les tests unitaires et d'intégration.
  - Scripts de CI qui exécutent les tests automatiquement à chaque push ou merge dans la branche principale.
- **Tests Automatisés**
  - Tests automatisés (unitaires, d'intégration,...) intégrée dans le processus de CI.
  - Rapports de test générés automatiquement après chaque exécution de CI.
- **Déploiement Continu (CD)**
  - Configuration du pipeline de CD pour automatiser le déploiement du code dans un environnement de staging ou de production.
  - Scripts de déploiement.
  - Documentation sur le processus de déploiement et la manière d'accéder à

l'application déployée.

- **Conteneurisation et Orchestration**
  - Dockerfile pour construire des images de conteneurs de l'application.
  - Configuration pour l'orchestrateur de conteneurs (par exemple, Kubernetes ou Docker Swarm) pour gérer le déploiement des conteneurs.
- **Monitoring et Logging**
  - Configuration d'outils de monitoring (par exemple, Prometheus) et de logging (par exemple, ELK Stack) pour surveiller les performances de l'application et les journaux d'activité.
  - Tableaux de bord de monitoring personnalisés pour visualiser les métriques clés de performance et d'utilisation.

3. Contraintes

3.1 Coûts

Budget alloué au projet : 0 DH.

3.2 Délais

Date de livraison du produit : 9 mai 2024

3.3 Autres contraintes

Le code source du blog doit être documenté et commenté.

4. Déroulement du projet

4.1 Planification

Le déroulement du projet s'organisera autour de cycles itératifs Scrum :

- Sprint Planning : Au début de chaque sprint, l'équipe se réunit pour définir les tâches à accomplir (user stories) et les estimer en points d'effort.
- Développement : Pendant la durée du sprint, l'équipe se concentre sur le développement des fonctionnalités prévues.
- Daily Scrum : Chaque jour, l'équipe tient une réunion brève (15 minutes) pour faire le point sur l'avancement des tâches et identifier les obstacles.
- Revue de sprint : À la fin du sprint, l'équipe présente le travail effectué aux parties prenantes (product owner) et recueille leurs retours.
- Rétrospective : L'équipe se réunit pour identifier les points forts et les points faibles du sprint et pour apporter des améliorations au processus de développement pour le sprint suivant.

Cette approche permet une livraison continue du produit et une adaptation aux besoins et aux changements.

4.2 Plan d'assurance qualité

Un plan d'assurance qualité sera mis en place pour garantir la qualité du produit. Ce plan comprendra les éléments suivants :

- Tests unitaires : Les développeurs écriront des tests unitaires pour vérifier le bon fonctionnement des modules individuels du code.
- Tests d'intégration : Des tests d'intégration seront effectués pour vérifier la communication entre les différents modules du système.
- Tests de performance : Des tests de performance seront réalisés pour s'assurer que le blog peut gérer un grand nombre d'utilisateurs et d'articles sans perte de performance.
- Tests de sécurité : Des tests de sécurité seront effectués pour identifier et corriger les vulnérabilités du système.
- Tests d'utilisabilité : Des tests d'utilisabilité seront réalisés avec des utilisateurs finaux pour vérifier que l'interface du blog est intuitive et facile à utiliser.

4.3 Documentation

- Cahier de charge
- Diagramme de Gantt
- Rapport du projet
- Code source

4.4 Technologies

- Backend: Spring Boot (Java)
- API: REST (pour la communication frontend/mobile-backend)
- Frontend: Angular (interface web)
- Mobile: Android Studio et Java (application mobile native)
- Base de données: MySQL (stockage des données)
- Version control : Git (gestion du code source)
- Conteneurisation: Docker (facilite le déploiement et l'exécution des applications)
- Gestion de build: Maven (automatisation de la construction du projet)
- Interface utilisateur: HTML et CSS (conception visuelle de l'application web et mobile)