Documentation de Gestion de Projet

Auteurs: Équipe AP4
Date de création: 03/11/2024
Dernière mise à jour: 03/11/2024

Sommaire

Introduction

Cette documentation décrit l'organisation du travail adoptée pour assurer un avancement régulier du projet de chat AP4. Elle détaille la méthodologie en cascade suivie, les outils utilisés, notamment GitLab comme plateforme centrale de gestion des tickets, ainsi que les processus mis en place pour garantir la qualité du produit final.

Organisation de l'équipe

Rôles et responsabilités

L'équipe de développement est organisée selon une structure hiérarchique avec les rôles suivants :

- **Chef de projet** : Responsable de la coordination générale, de la planification, du suivi de l'avancement et de l'avancement

Méthodologie de développement

Approche en Cascade (Waterfall)

Le projet est développé selon la méthodologie en cascade, qui divise le développement en phases séquentielles distinctes, chaque phase devant être complétée avant de passer à la suivante.

Phases du projet

1. **Étude préalable** : Analyse du besoin et faisabilité
    - Définition du périmètre du projet
    - Étude de faisabilité technique
    - Estimation des délais

2. **Analyse et conception** : Spécification détaillée
    - Conception de l'architecture
    - Élaboration des modèles de données
    - Conception de l'interface utilisateur
3. **Développement** : Implémentation du système
    - Codage des différents modules
    - Intégration des composants
    - Développement des interfaces
4. **Tests** : Vérification et validation
    - Tests unitaires
5. **Déploiement** : Mise en production
    - Installation du système
6. **Maintenance** : Support et évolution
    - Correction des bugs
    - Ajout de nouvelles fonctionnalités
    - Optimisations

Livrables par phase

Chaque phase produit des livrables spécifiques qui sont validés avant de passer à la phase suivante :

- **Étude préalable** : Document de cadrage, cahier des charges
- **Analyse et conception** : Spécifications détaillées, diagrammes UML, maquettes d'interface
- **Développement** : Code source, documentation technique
- **Tests** : Plans de test, rapports de test, résultats de validation
- **Déploiement** : Manuels d'installation et d'utilisation, système opérationnel
- **Maintenance** : Rapports d'incidents, documentation des modifications

Outils de gestion de projet

GitLab comme plateforme centrale

GitLab est utilisé comme plateforme centrale pour la gestion du projet, avec une organisation adaptée à la méthodologie en cascade :

- Un système de contrôle de version (Git)
- Un outil de suivi des tickets (Issues)
- Un outil de gestion des versions (Releases)
- Un outil d'intégration continue (CI/CD)

Principales fonctionnalités utilisées

1. **Issues (Tickets)** :

    - Organisation par phases du projet
    - Suivi des tâches par étapes de développement
    - Attribution des tickets en cours
    - Étiquetage par priorité et catégorie (server, client....)

2. **Milestones (Jalons)** :

    - Définition des jalons correspondant aux phases du projet
    - Suivi de l'achèvement des différentes phases
    - Contrôle des délais par rapport au planning initial

3. **Repository (Dépôt de code)** :

    - Stockage centralisé du code source
    - Gestion des versions du code
    - Organisation par branches stables pour chaque phase
    - Contrôle et validation des modifications

4. **CI/CD** :

    - Exécution automatique des tests
    - Vérification de la qualité du code
    - Génération des livrables pour chaque phase

Autres outils

- **Figma** : Conception des interfaces utilisateur
- **IntelliJ IDEA / Eclipse** : Environnements de développement
- **Draw.io / Mermaid / PlantUML** : Création des diagrammes UML et autres schémas techniques

Suivi de l'avancement

Le suivi de l'avancement du projet est effectué à l'aide des outils suivants :

- **Tableaux de bord GitLab** : Suivi des tickets et de leur état
- **Rapports d'avancement hebdomadaires** : Synthèse de l'avancement, des risques et des actions à mener

Gestion des versions et du code source

Stratégie de branches

La gestion des branches suit un modèle adapté à la méthodologie en cascade :

- **main** : Branche principale, contient la version stable et validée du projet
- **dev** : Branche de développement intégrant progressivement les fonctionnalités
- **feature/*** : Branches temporaires pour le développement des fonctionnalités
- **release/*** : Branches de préparation des versions à déployer
- **hotfix/*** : Branches pour les corrections urgentes

Workflow de développement

1. Création d'un ticket détaillant la tâche à réaliser
2. Attribution du ticket
3. Création d'une branche dédiée à partir de dev
4. Développement avec commits réguliers
5. Tests unitaires locaux
6. Merge dans dev après validation
7. Tests sur dev
8. Création d'une branche de release pour regrouper les fonctionnalités développées
9. Tests système et validation sur la branche de release
10. Merge dans main après validation complète

Conventions de nommage

- **Branches** : `type/phase-nom-fonctionnalité` (ex: `feature/dev-login-screen`)
- **Commits** : Message explicite commençant par un verbe à l'infinitif (ex: `Ajouter l'écran de connexion`) + identifier de partie (ex: [DOC], [FEATURE], [FIX])
- **Tickets** : description (ex: `Implémentation de l'écran de connexion`)