Documentation de l'Environnement Technologique

Aperçu

Ce document décrit l'environnement technologique utilisé dans le projet d'Application de Chat, détaillant les technologies, frameworks, bibliothèques et outils utilisés dans les différents composants du système.

Environnement de Développement

Langages

- **Java** (JDK 21) : Langage de programmation principal pour les composants client et serveur
- **SQL** : Utilisé pour les opérations de base de données et la définition du schéma

Outils de Build

- **Maven 3.9+** : Gestion des dépendances et automatisation des builds
- **Makefile** : Automatisation supplémentaire et simplification des commandes

IDEs & Outils de Développement

- **Git** : Système de contrôle de version
- **GitLab CI/CD** : Pipeline d'intégration et de déploiement continus

Architecture

L'application suit une architecture modulaire client-serveur avec trois composants principaux :

1. **chat-common** : Module partagé contenant le code commun, les modèles et les utilitaires
2. **chat-server** : Serveur backend utilisant Spring Boot
3. **chat-client** : Application client desktop construite avec JavaFX

Technologies Backend

Frameworks Principaux

- **Spring Boot** : Framework principal pour l'application serveur
- **Spring WebSocket** : Fournit des capacités WebSocket pour la messagerie en temps réel
- **Protocole STOMP** : Utilisé pour la structure des messages WebSocket

Gestion des Données

- **PostgreSQL** : Base de données relationnelle pour la persistance
- **Spring Data JPA** : Accès à la base de données et ORM
- **Hibernate** : Implémentation JPA pour le mapping objet-relationnel

Logging & Surveillance

- **Log4j2** : Framework de journalisation

- **Spring Boot Actuator** : Surveillance et gestion de l'application

Technologies Frontend

Framework UI

- **JavaFX** : Framework d'application desktop
- **FXML** : Définition de layout UI basée sur XML

Style

- **CSS** : Style personnalisé pour les composants JavaFX

Client WebSocket

- **Spring WebSocket Client** : Connectivité au serveur WebSocket
- **STOMP Client** : Implémentation du protocole STOMP pour la messagerie

Outils de Test

Frameworks de Test

- **JUnit 5** : Framework de test unitaire
- **Mockito** : Framework de mockup pour les tests unitaires

Qualité du Code

- **JaCoCo** : Rapport de couverture de code

Déploiement & Distribution

Conteneurisation

- **Docker** : Conteneurisation du composant serveur
- **Docker Compose** : Orchestration multi-conteneurs pour le déploiement en production

Packaging

- **JPackage** : Packaging natif de l'application client
- **Installateurs spécifiques à la plateforme** :
    - Windows : Installateurs EXE et MSI
    - Linux : Packages DEB et RPM

Pipeline CI/CD

- **GitLab CI/CD** : Pipeline automatisé de build, test et déploiement
- **Pipeline multi-étapes** : Validation, test, build, packaging et déploiement

Sécurité

- **Hachage de mot de passe** : Protection des identifiants utilisateur

- **Spring Security** : Framework d'authentification et d'autorisation
- **Gestion de session WebSocket** : Gestion sécurisée des connexions

Communication Réseau

- **WebSockets** : Communication bidirectionnelle en temps réel
- **STOMP** : Protocole de messagerie orienté texte simple pour la communication WebSocket
- **JSON** : Format de sérialisation de données pour l'échange de messages

Support Multi-Plateforme

- **JVM** : Exécution indépendante de la plateforme
- **Packages natifs** : Distribution spécifique à la plateforme pour Windows et Linux