Documentation Technique - LinkUp

Sommaire

1. Introduction

LinkUp est une application de messagerie instantanée de type Discord qui permet aux utilisateurs de communiquer en temps réel via une interface de bureau. Le projet est développé en Java avec JavaFX pour l'interface utilisateur et Spring Boot pour la partie serveur.

2. Architecture du Projet

L'application est structurée en trois modules principaux :

Structure Globale

```
linkup/
  ├── client/    # Application de bureau JavaFX
  ├── server/    # Serveur WebSocket avec Spring Boot
  ├── shared/    # Modèles et utilitaires partagés
  └── docs/      # Documentation du projet
```

L'architecture suit un modèle client-serveur avec une communication en temps réel via WebSocket/STOMP.

3. Module Shared

Le module `shared` contient tous les éléments communs utilisés par le client et le serveur.

Modèles

- **User** : Gestion des utilisateurs avec leurs informations (nom, email, statut, etc.)
- **Message** : Structure des messages échangés
- **Chat** : Représentation des salons de discussion
- **Friend** : Gestion des relations d'amitié entre utilisateurs
- **FriendRequest** : Gestion des demandes d'amitié
- **StatusUpdate** : Mise à jour du statut des utilisateurs

Énumérations

- **UserStatus** : Statuts possibles d'un utilisateur (ONLINE, AWAY, BUSY, OFFLINE)
- **FriendStatus** : États de relation d'amitié
- **FriendRequestAction** : Actions possibles sur une demande d'amitié

Exceptions

- Exceptions personnalisées pour la gestion des erreurs

4. Module Server

Le module `server` est responsable de la gestion des connexions WebSocket, de l'authentification et du routage des messages.

Composants Principaux

- **Main** : Point d'entrée de l'application Spring Boot
- **WebSocketController** : Contrôleur des échanges WebSocket
- **WebSocketSessionManager** : Gestion des sessions WebSocket
- **WebSocketConfig** : Configuration des WebSockets

Fonctionnalités

- Authentification des utilisateurs
- Routage des messages entre les clients
- Gestion des salons de discussion
- Persistance des données

5. Module Client

Le module `client` est l'application de bureau JavaFX qui permet aux utilisateurs d'interagir avec le système.

Structure MVC

- **Models** : Utilisation des modèles du module shared
- **Views** : Fichiers FXML pour l'interface utilisateur
- **Controllers** : Contrôleurs JavaFX pour la logique de l'interface

Composants Principaux

- **Main** : Point d'entrée de l'application JavaFX
- **Bootstrap** : Initialisation de l'application
- **SceneSwitcher** : Gestion des changements d'écrans

Contrôleurs

- **LoginController** : Gestion de l'authentification
- **RegisterController** : Inscription des utilisateurs
- **ChatController** : Interface principale de chat
- **SidebarController** : Navigation latérale
- **SettingsController** : Configuration utilisateur
- **AddFriendController** : Ajout d'amis
- **FriendListCellController** : Affichage des amis
- **HomeController** : Écran d'accueil
- **PinnedMessagesController** : Gestion des messages épinglés
- **SearchResultsController** : Affichage des résultats de recherche

Services

- **AuthenticationService** : Gestion de l'authentification
- **WebSocket Services** : Communication avec le serveur
- **ChatService** : Gestion des discussions
- **FriendService** : Gestion des amis
- **UIService** : Utilitaires pour l'interface

6. Communication Client-Serveur

WebSocket/STOMP

- Communication en temps réel via STOMP sur WebSocket
- Gestion des événements de connexion/déconnexion
- Envoi et réception de messages

Protocole d'Échange

- Format JSON pour les échanges de données
- Système d'abonnement à des topics spécifiques
- Notifications en temps réel

7. Base de Données

La persistance des données est assurée par MariaDB.

Schéma de Base de Données

Le schéma comprend les tables principales suivantes :

- Users
- Messages
- Chats
- Friends
- FriendRequests

8. Bibliothèques et Dépendances

Principales Dépendances

- **JavaFX** : Interface utilisateur
- **Spring Boot** : Framework serveur
- **Spring Messaging** : Gestion des WebSockets
- **STOMP** : Protocole de messagerie
- **Log4j2** : Journalisation
- **JUnit & Mockito** : Tests unitaires

9. Sécurité

Authentification

- Système de login/password

- Stockage sécurisé des mots de passe

10. Logging

Le système utilise Log4j2 pour la journalisation des événements.

Configuration

- Logs de débogage et d'erreurs
- Rotation des fichiers de logs

11. Tests

Tests Unitaires

- Tests des services et contrôleurs
- Mocks pour les dépendances externes

Tests d'Intégration

- Tests de la communication client-serveur
- Tests des flux utilisateur