最終課題　チャットアプリケーションのリソースサーバー

設定ファイル

build.gradle
Gradleビルドツールの設定ファイルです。依存関係やプラグインの設定が含まれています。

application.properties
Spring Bootの設定ファイルです。データベース接続やその他の設定が含まれています。

技術スタック

バックエンドサーバー

- 言語/フレームワーク　Java/Spring Boot
- アーキテクチャ　レイヤーアーキテクチャ
- API　Rest API
- クエリ管理　jooq

データベース

- RDB　PostgreSQL

コンテナ

- docker

バージョン管理ツール

- Git

サーバーの起動

データベースの初期化

ローカルのdockerコンテナ上でPostgreSQLを立ち上げます。
(注)dockerをインストールしていない場合はインストールしてください

databaseディレクトリに移動してdockerコンテナを起動してください

```
cd database
```

```
docker compose up
```

これでDBの初期化の完了です。

サーバー起動時に接続できない場合は、DBeaverなどのDBクライアントツールから接続してdatabaseが構築されていることを確認してください。

サーバーの起動

サーバーはコンテナかしていないのでローカルホスト上に直接立ち上げます。
rootに移動して実行してください

```
cd ../
```

```
./gradlew build clean
```

```
./gradlew bootRun
```

HTTPリクエストの送信の仕方

以下に実装しているエンドポイント一覧を記述していますが、エンドポイントへのリクエストを送信する方法を記載しておきます。今回はローカル環境で飲みの実行を想定しています。
ローカルホストの8090ポートでリクエストを受け付ける想定なので、以下の8090以下とリクエストメソッドを変更して試行してください。

```
curl -X GET "http://localhost:8090/message/1"
```

実装エンドポイント

UserController

- **POST /user/login**: ユーザーのログインを処理します。
  もしユーザーが存在しない場合はエラーを返します。
- **POST /user/register**: 新しいユーザーの登録を処理します。
- **POST /user/health**: 疎通確認用のエンドポイントです。

ChatRoomController

- **POST /chatroom**: 新しいチャットルームを作成します。
  request bodyの例

```
{
        "name": "テスト用のchatroom",
        "isPrivate": false
}
```

- **PUT /chatroom/{id}**: 指定されたIDのチャットルームを更新します。
  実装はできていませんが、リクエストのみ受け付けるようにしています。
- **DELETE /chatroom/{id}**: 指定されたIDのチャットルームを削除します。
  実装はできていませんが、リクエストのみ受け付けるようにしています。
- **GET /chatroom/{id}**: 指定されたIDのチャットルームを取得します。
- **GET /chatroom**: チャットルームの一覧を取得します。

ChatRoomUsersController

- **POST /join**: チャットルームにUserに参加する
- **DELETE /join**: 実装はできていませんが、リクエストのみ受け付けるようにしています。

MessageController:

- **GET /message/{chatRoomID}**: 指定されたIDのsチャットルームのメッセージを入手します。
- **POST /message**: メッセージを登録します。
  request bodyの例

```
{
        "userID": "d5c1a192-37d6-4f38-b5e2-c8ef8e1fc7f8",
        "message": "test message",
        "chatRoomID": 1
}
```