TRƯỜNG ĐẠI HỌC THĂNG LONG

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and blue logo

Description automatically generated

**LẬP TRÌNH SPRING BOOT**

**TÀI LIỆU HƯỚNG DẪN XÁC THỰC BẰNG JWT VÀ TÍCH HỢP THANH TOÁN BẰNG VNPAY**

**GIẢNG VIÊN HƯỚNG DẪN: SINH VIÊN THỰC HIỆN:**

**ThS. Nguyễn Hùng Cường A41365 Nguyễn Đức Long**

**HÀ NỘI – 2024**

**Mục lục**

[CHƯƠNG 1. khởi tạo chương trình 1](#_Toc178546679)

[1.1. Front end 1](#_Toc178546680)

[1.2. Back end 2](#_Toc178546681)

[CHƯƠNG 2. Triển khai chương trình 3](#_Toc178546682)

[2.1. Front end 3](#_Toc178546683)

[2.1.1. Cấu hình source code 3](#_Toc178546684)

[2.1.2. Cấu hình redux 3](#_Toc178546685)

[2.1.3. Cấu hình route 4](#_Toc178546686)

[2.1.4. Cấu hình các hàm để call api 5](#_Toc178546687)

[2.1.5. Xử lí chức năng đăng kí tài khoản 7](#_Toc178546688)

[2.1.6. Xử lí chức năng đăng nhập 8](#_Toc178546689)

[2.1.7. Xử lí trạng thái sau khi đăng nhập 9](#_Toc178546690)

[2.1.8. Xử lí chức năng thanh toán 10](#_Toc178546691)

[2.2. Back end 11](#_Toc178546692)

[2.2.1. Cấu hình source code 11](#_Toc178546693)

[2.2.2. Cấu hình kết nối cơ sở dữ liệu 11](#_Toc178546694)

[2.2.3. Cấu hình SecurityConfig 11](#_Toc178546695)

[2.2.4. Cấu hình VNPAY 12](#_Toc178546696)

[2.2.5. Tạo User model 13](#_Toc178546697)

[2.2.6. Cấu hình JWTUtils 15](#_Toc178546698)

[2.2.7. Cấu hình SigninRequest, SignupRequest 15](#_Toc178546699)

[2.2.8. Cấu hình UserRepository 16](#_Toc178546700)

[2.2.9. Cấu hình UserController 16](#_Toc178546701)

[2.2.10. Cấu hình UserDetailsServiceImpl 18](#_Toc178546702)

[2.2.11. Cấu hình UserServices 19](#_Toc178546703)

[2.2.12. Cấu hình PaymentDTO, ResponseObject, TransactionVerifyRequest 19](#_Toc178546704)

[2.2.13. Cấu hình VNPayUtil 21](#_Toc178546705)

[2.2.14. Cấu hình PaymentController 22](#_Toc178546706)

[2.2.15. Cấu hình PaymentServices 22](#_Toc178546707)

**DANH MỤC THAM KHẢO**

[Hình 1. Api đăng kí người dùng 17](#_Toc178546731)

[Hình 2. Api xác thực người dùng 17](#_Toc178546732)

[Hình 3. Api refresh\_token 18](#_Toc178546733)

[Hình 4. Api lấy thông tin người dùng bằng id 18](#_Toc178546734)

[Hình 5. Cấu hình các hàm tiện tích thanh toán VNPAY 21](#_Toc178546735)

# 

# khởi tạo chương trình

## Front end

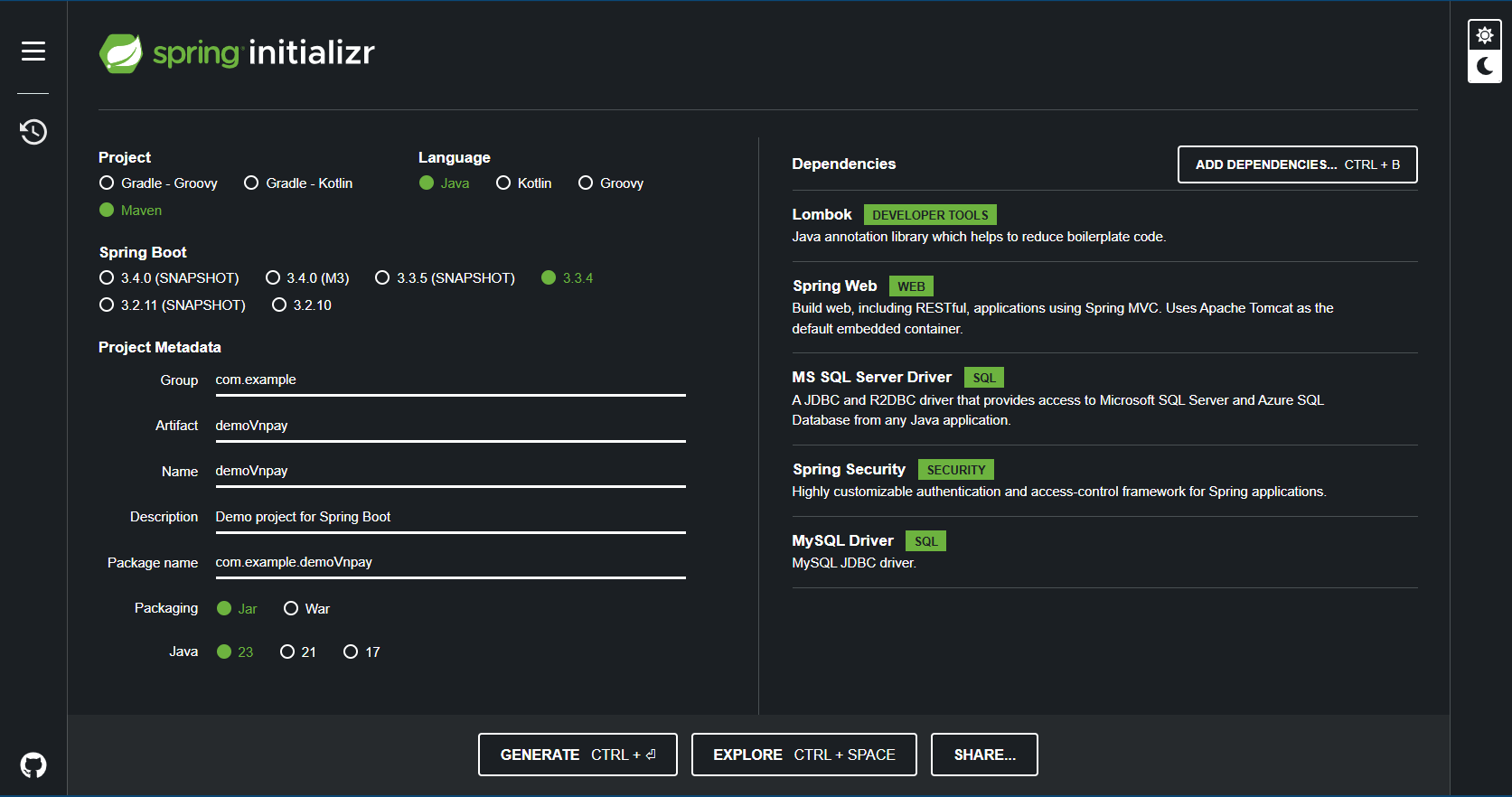
* Truy cập trang web [Install Tailwind CSS with Vite - Tailwind CSS](https://tailwindcss.com/docs/guides/vite) khởi tạo chương trình react vite theo hướng dẫn
* Cài đặt các thư viện cần thiết



* **react-dom:**
* Câu lệnh cài đặt: *npm install react-dom@18.3.1*
* Công dụng: Cung cấp các phương thức để render các thành phần React vào DOM và quản lý các thay đổi trên giao diện web.
* **axios:**
* Câu lệnh cài đặt: *npm install axios@1.7.3*
* Công dụng: Thư viện HTTP client giúp gửi các yêu cầu HTTP (GET, POST, PUT, DELETE) dễ dàng trong React, đặc biệt là khi làm việc với API.
* **jwt-decode:**
* Câu lệnh cài đặt: *npm install jwt-decode@4.0.0*
* Công dụng: Giải mã JWT (JSON Web Token) từ chuỗi JWT mà không cần xác thực token, giúp lấy thông tin payload (dữ liệu bên trong token).
* **@reduxjs/toolkit:**
* Câu lệnh cài đặt: *npm install @reduxjs/toolkit@2.2.7*
* Công dụng: Cung cấp các công cụ để quản lý trạng thái (state) trong ứng dụng React theo mô hình Redux. Redux Toolkit giúp giảm bớt cấu hình phức tạp của Redux.
* **@tanstack/react-query:**
* Câu lệnh cài đặt: *npm install @tanstack/react-query@5.51.21*
* Công dụng: Thư viện quản lý trạng thái dữ liệu server trong React, giúp thực hiện các truy vấn API, caching, đồng bộ dữ liệu và quản lý trạng thái một cách tự động.
* **react-redux:**
* Câu lệnh cài đặt: *npm install react-redux@9.1.2*
* Công dụng: Thư viện kết nối giữa React và Redux, giúp các thành phần React có thể truy cập và cập nhật state của Redux store.
* **react-router-dom:**
* Câu lệnh cài đặt: *npm install react-router-dom@6.24.1*
* Công dụng: Thư viện định tuyến (routing) cho React, cho phép tạo các đường dẫn URL và điều hướng giữa các trang trong ứng dụng web.

## Back end

* Truy cập trang web [Spring Initializr](https://start.spring.io/) khởi tạo chương trình back-end Spring boot, thêm các dependencies cần thiết, ấn generate để tải chương trình về.



# Triển khai chương trình

## Front end

### Cấu hình source code

A screenshot of a computer screen

Description automatically generated

**access:** Quản lí các ảnh

**pages:** Quản lí các trang của chương trình

**redux:** Quản lí trạng thái của chương trình

**services:** Call api từ phía back-end

**routes.jsx:** cấu hình route

### Cấu hình redux

Trong Folder redux tạo 2 file **userSlice.js** và **store.js**

* **userSlice.js :** Quản lí và cập nhật trạng thái người dùng



* **store.js:** Cấu hình store
* Sử dụng configureStore để tạo Redux store. Reducer của user được thêm vào store để quản lý state liên quan đến người dùng.



### Cấu hình route

Trong file routes.jsx cấu hình route để tạo cơ chế tải layout

A screen shot of a computer program

Description automatically generated

### Cấu hình các hàm để call api

Trong folder services tạo 2 file **AuthServices.js**, **VnpayServices.js**

* **AuthServices.js:** Call api liên quan đến đăng kí, xác thực người dùng

A screenshot of a computer program

Description automatically generated

* **signIn:** Call api đăng nhập
* **signUp:** Call api đăng kí người dùng
* **getDetailUser:** Call api lấy thông tin người dùng
* **VnpayServices.js:** Call api liên quan tới thanh toán bằng VNPAY

A screenshot of a computer program

Description automatically generated

* **createVnPayPayment:** Tạo thanh toán và điều hướng sang trang thanh toán VNPAY
* **verifyTransaction:** Xác thực thanh toán xem có hợp lệ không

### Xử lí chức năng đăng kí tài khoản



Gọi hàm signUp từ AuthServices gửi thông tin đăng kí call api

### Xử lí chức năng đăng nhập

A screen shot of a computer

Description automatically generated

* Gọi hàm **signIn** từ **AuthServices** gửi thông tin đăng nhập, nhận được response trả về gồm **status** và **access\_token** sau đó set **access\_token** vào **localStorage**
* Dùng **JwtDecode** mã hóa token nhận được **userId**, gọi hàm **getDetailUser** truyền **userId** lấy được từ trước nhận được thông tin user cập nhật lại state của chương trình bằng user nhận được

### Xử lí trạng thái sau khi đăng nhập

A screen shot of a computer screen

Description automatically generated

* Lấy dữ liệu **accessToken** từ **localStorage** mã hóa lấy được **userId**, gọi hàm **getDetailUser** lấy được dữ liệu **user**, cập nhật vào **state**
* Xử lí khi **token** hết hạn gọi hàm **refreshToken** cấp lại token mới

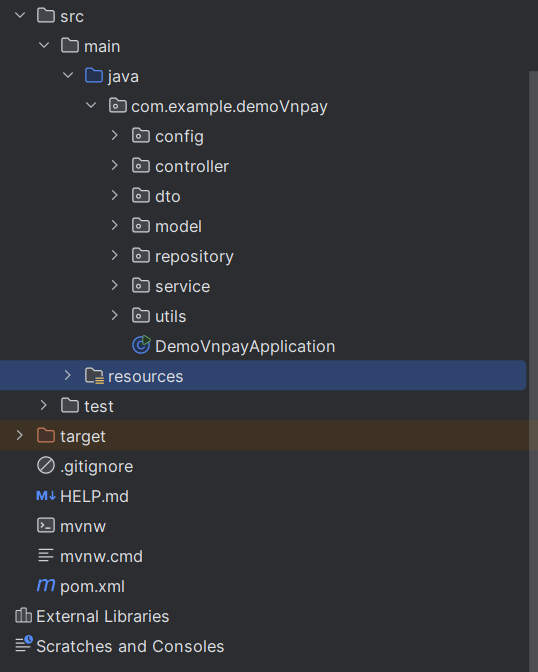
### Xử lí chức năng thanh toán

* Tạo một đơn hàng demo gọi hàm **createVnpayPayment** truyền tổng số tiền đơn hàng, sau đó trình duyệt sẽ chuyển tới trang thanh toán của VNPay nhập thông tin tài khoản test (xem hướng dẫn dưới phần back-end) xác nhận thanh toán.
* Sau đó trình duyệt lại điều hướng lại trang web với mã giao dịch và một đoạn mã đã được mã hóa truyền dữ liệu 2 mã này vào hàm **verifyTransaction** để call api xác thực nếu hợp lệ sẽ báo lại thành công(Hãy xử lí lưu thông tin đơn hàng khi thành công)



## Back end

### Cấu hình source code



* **config:** cấu hình bảo mật và cấu hình VNPAY
* **controller:** Cấu hình bộ điều khiển api
* **dto:** Cấu hình dữ liệu nhận và gửi
* **model:** Cấu hình model đối tượng của chương trình
* **repository:** Quản lí, truy xuất, lưu dữ liệu,…
* **services:** Lớp logic nghiệp vụ trung gian giữa controller và repository

### Cấu hình kết nối cơ sở dữ liệu

A screenshot of a computer

Description automatically generated

### Cấu hình SecurityConfig

Tạo file SecurityConfig.java trong thư mục config

A screenshot of a computer program

Description automatically generated

* Cấp các quyền cho các api
* Cấu hình CORS cho phép front-end call api
* Cấu hình mã hóa mật khẩu

### Cấu hình VNPAY

Truy cập trang web [Giới thiệu · Cổng thanh toán VNPAY (vnpayment.vn)](https://sandbox.vnpayment.vn/apis/docs/gioi-thieu/) đăng kí tài khoản.

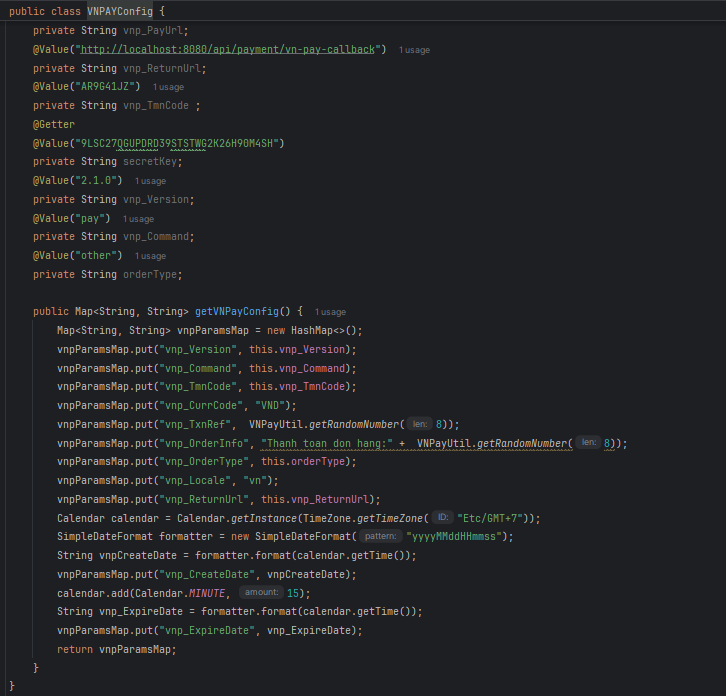
Sau đó sẽ nhận được thông tin cấu hình VNPAY và tài khoản test trong gmail đăng kí:

A screenshot of a computer

Description automatically generated A close up of a number

Description automatically generated

Dựa vào thông tin nhận được tạo một file VNPAYConfig.java trong thư mục config:



Thay đổi mã website và chuỗi bí mật nhận được trong gmail

### Tạo User model

Tạo file User.java trong thư mục model cấu hình setter, getter:

A screen shot of a computer program

Description automatically generated

### Cấu hình JWTUtils

A computer screen shot of text

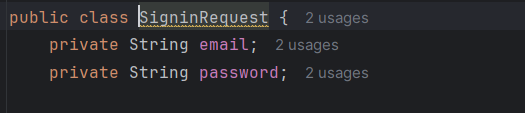
Description automatically generated

Cấu hình các hàm tiện ích xác thực:

* **generateAccessToken:** Tạo accessToken
* **generateRefreshToken:** Tạo refreshToken
* **validateRefreshToken:** Kiểm tra refreshToken
* **getUserIdFromRefreshToken:** Get user từ refreshToken

### Cấu hình SigninRequest, SignupRequest

Tạo 2 file SigninRequest và SignupRequest trong thư mục dto

 A screen shot of a computer program

Description automatically generated

### Cấu hình UserRepository

A computer screen shot of text

Description automatically generated

### Cấu hình UserController

A computer screen with text on it

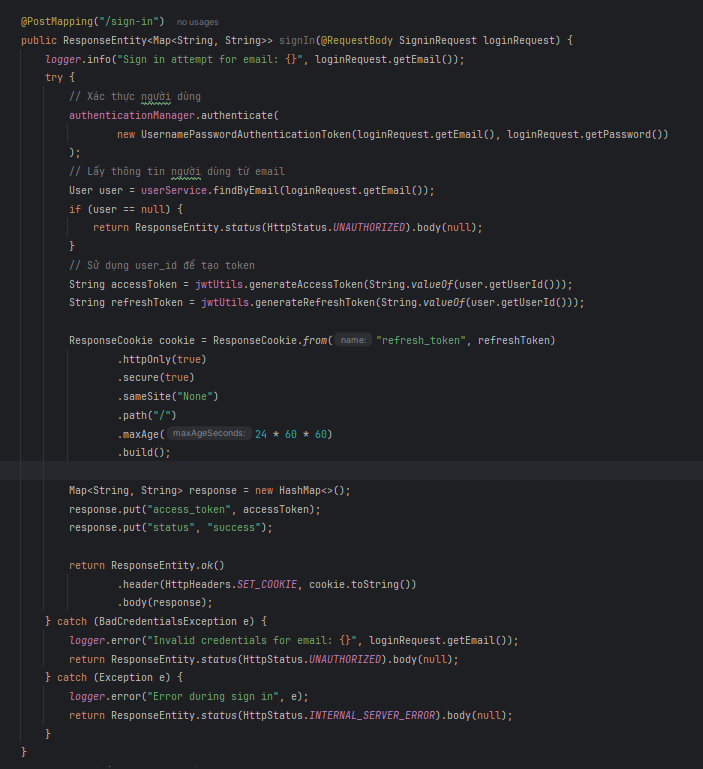
Description automatically generated

Hình 1. Api đăng kí người dùng

A screen shot of a computer code

Description automatically generated

Hình 2. Api xác thực người dùng



Api sign-in sẽ nhận tài khoản mật khẩu từ phía client, sau đó get user bằng email nhận được, tạo token bằng userId, trả về access\_token còn refresh\_token thì set vào cookie

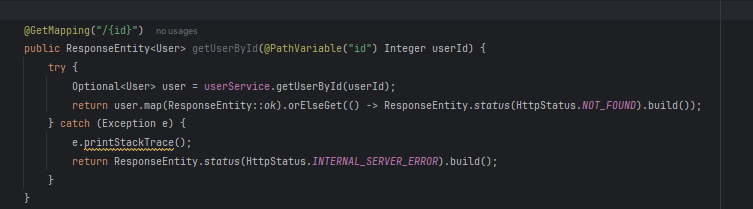
Hình 3. Api refresh\_token

A screen shot of a computer program

Description automatically generated

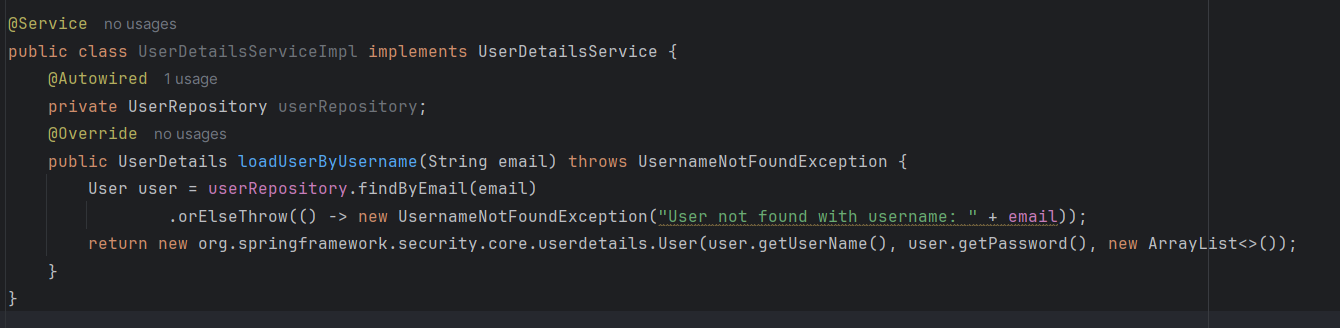
Api refresh-token sẽ lấy token từ cookie và cấp lại token mới cho người dùng

Hình 4. Api lấy thông tin người dùng bằng id

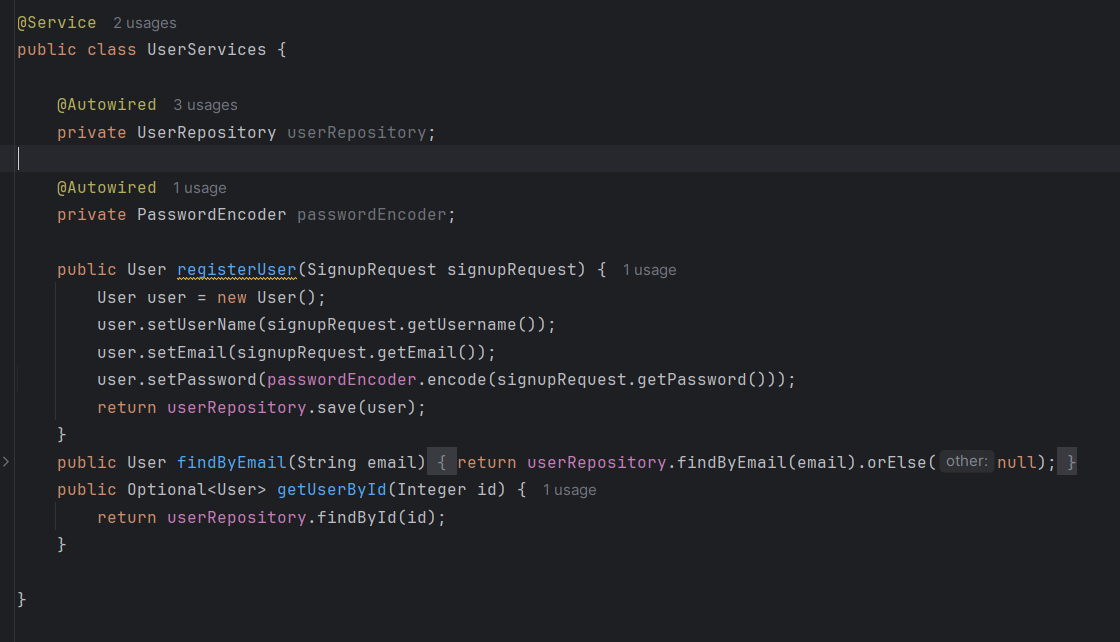


### Cấu hình UserDetailsServiceImpl

Tạo file UserDetailsServiceImpl.java trong services để lấy thông tin người dùng trong quá trình xác thực

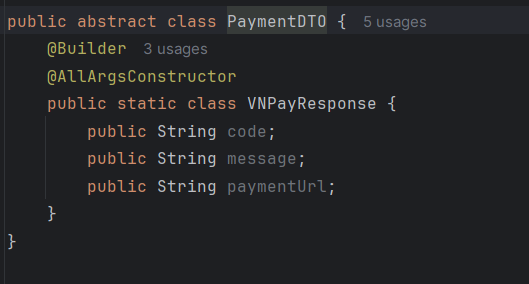


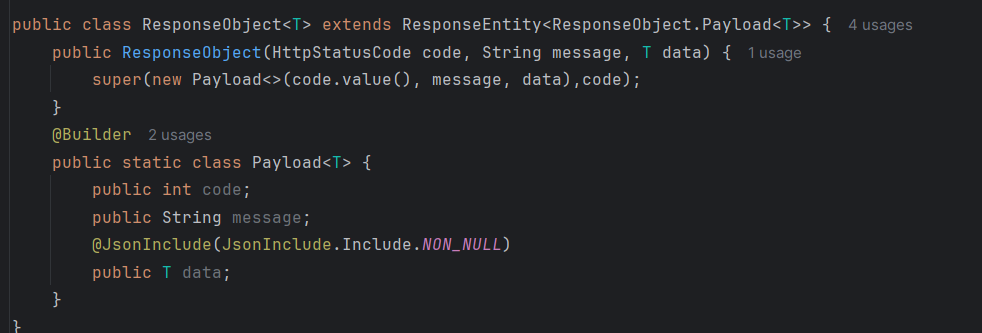
### Cấu hình UserServices



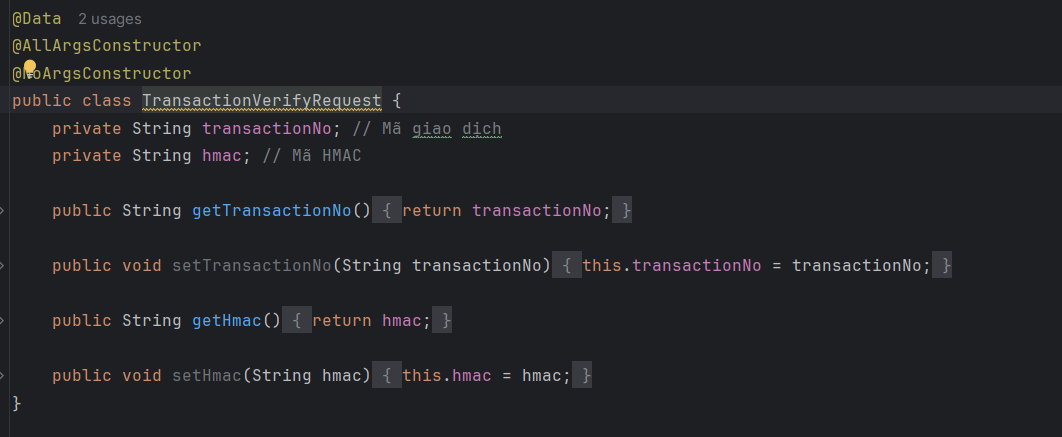
### Cấu hình PaymentDTO, ResponseObject, TransactionVerifyRequest

Cấu hình PaymentDTO và ResponseObject để format dữ liệu trả về khi hoàn tất thanh toán





Cấu hình TransactionVerifyRequest format dữ liệu gửi về backend



### Cấu hình VNPayUtil

Hình 5. Cấu hình các hàm tiện tích thanh toán VNPAY



### Cấu hình PaymentController

A screenshot of a computer program

Description automatically generated

Khi client call api /vn-pay sẽ trả về một url thanh toán, truy cập url và hoàn tất thanh toán thì tự động call api vn-pay-callback nếu thành công sẽ điều hướng về frontent để xử lí, tiếp tục call api verify-transaction để xác thực.

### Cấu hình PaymentServices

Hàm xử lí tạo thanh toán VNPAY trả về trạng thái , và link thanh toán

