**TRƯỜNG ĐẠI HỌC SƯ PHAM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**&œ**

A logo with hands holding a book and a flame

Description automatically generated

**ĐỒ Án CUỐI KỲ**

**Tên đề tài: App du lịch**

**MÃ MÔN HỌC:**  **Lập trinh di động\_Nhom 02\_A309\_CT6**

**HỌC KỲ II NĂM HỌC 2023-2024**

**Thực hiện: Nhóm 17  
Thành viên : Vũ Đình Khang-22162015  
 Hồ Tùng Lâm-22162018**

**Giảng viên hướng dẫn: thầy Nguyễn Hữu Trung  
TP.HCM, 20/5/2024**

**Mục lục:**

1. **Giao diện  
   1.1 Đăng ký   
   1.2 Đăng nhập  
   1.3 Màn hình chính   
   1.4 Màn hình chi tiết từng khách sạn  
   1.5 Profile tài khoản  
   1.6 Màn hình các địa điểm đã thích**
2. **Code app  
   2.1 Đăng nhập  
   2.2 Đăng ký  
   2.3 Xem tất cả các khách sạn  
   2.4 Xem các địa điểm đã đặt  
   2.5 Xem profile cá nhân**
3. **API**
4. **Firebase**

1. Giao diện

1.1 Giao diện giới thiệu.

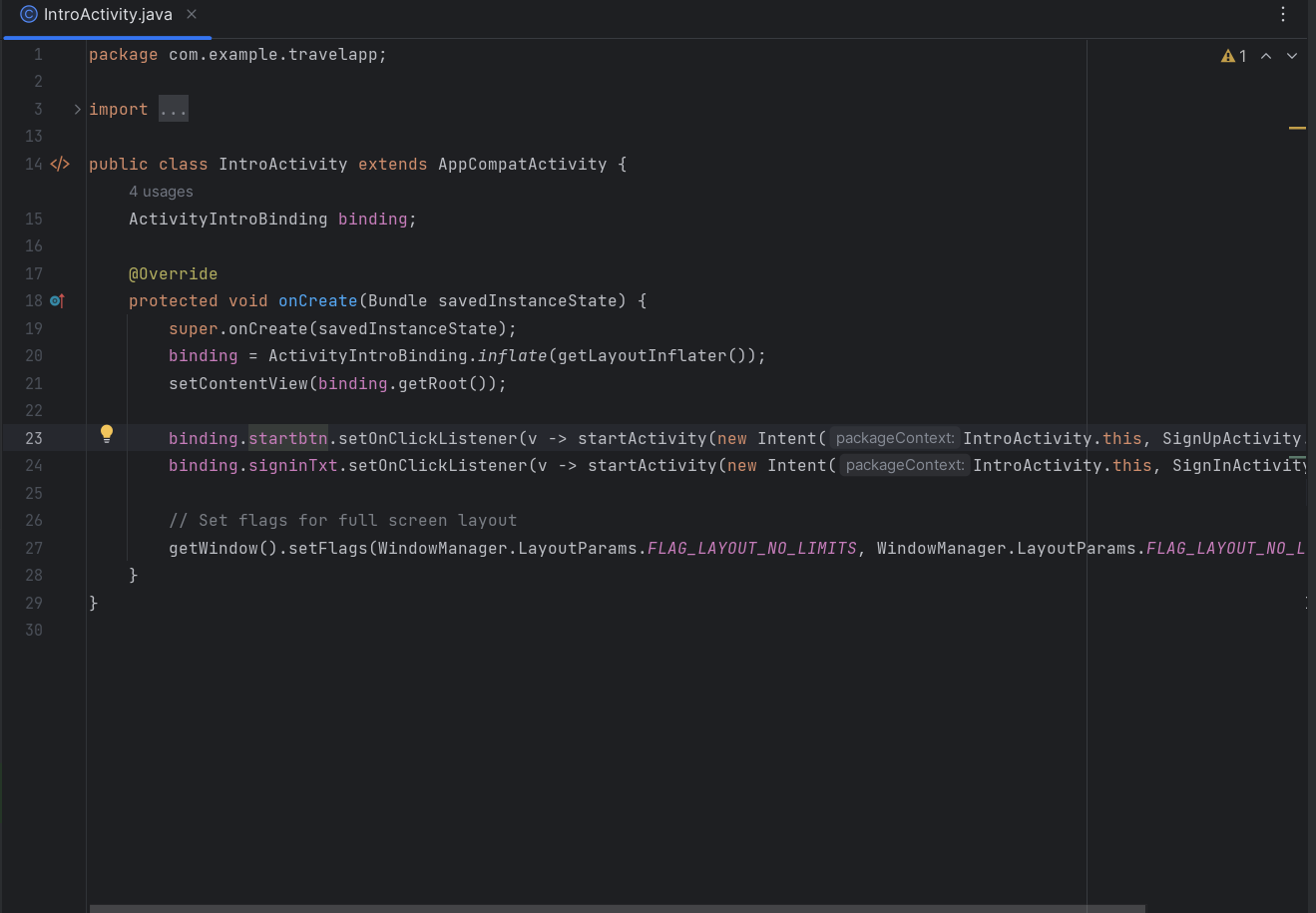
A person holding a small house

Description automatically generated

Code xml gồm các thành phần:A screenshot of a computer

Description automatically generated

Code activity gồm



Cách thức hoạt động:

Khởi tạo một đối tường binding cho layout của hoạt động sử dụng layout inflater.

binding.startbtn.setOnClickListener(...) đặt một listener cho nút "start" để bắt đầu SignUpActivity khi được nhấn.

binding.signinTxt.setOnClickListener(...) đặt một listener cho text "sign in" để bắt đầu SignInActivity khi được nhấn.  
A screen shot of a computer

Description automatically generated

1.2 Giao diện đăng ký

A screenshot of a phone

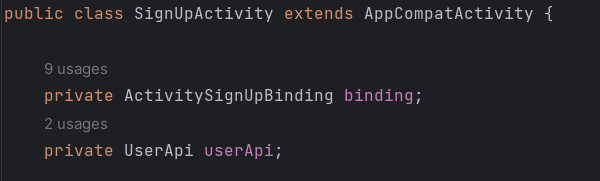
Description automatically generated

Code xml gồm các thành phần:

A screenshot of a computer

Description automatically generated

Code activity gồm:



Khai báo các biến binding và userApi để sử dụng view binding và gọi API.

A computer screen shot of a program code

Description automatically generated

Khởi tạo binding với ActivitySignUpBinding để liên kết với layout.

Thiết lập Retrofit với URL cơ sở và cấu hình GSON.

Khởi tạo userApi để gọi API.

Đặt sự kiện click cho nút đăng ký để gọi phương thức addUser.

A screen shot of a computer program

Description automatically generated

Phương thức addUser lấy dữ liệu từ các trường nhập liệu và tạo một đối tượng UserRequest.

Gọi phương thức addUser của userApi để gửi yêu cầu đăng ký.

Sử dụng enqueue để gọi API bất đồng bộ và xử lý phản hồi.

Nếu đăng ký thành công (response.isSuccessful()), chuyển sang màn hình đăng nhập (SignInActivity).

Nếu không thành công, bạn có thể xử lý lỗi (ví dụ: hiển thị thông báo lỗi cho người dùng).

Trong trường hợp gặp lỗi kết nối hoặc lỗi khác, xử lý trong phương thức onFailure.

1.3 Giao diện đăng nhập

A screenshot of a login screen

Description automatically generated

Code xml gồm các thành phần:

A screenshot of a computer

Description automatically generated

Code activity gồm:

A black screen with white text

Description automatically generated

Khai báo lớp SignInActivity kế thừa từ AppCompatActivity.

Khai báo các biến binding và userApi để sử dụng view binding và gọi API.

A computer screen with white text

Description automatically generated

Phương thức onCreate được gọi khi hoạt động được tạo.

Khởi tạo binding với ActivitySignInBinding để liên kết với layout.

Thiết lập Retrofit với URL cơ sở và cấu hình GSON.

Khởi tạo userApi để gọi API.

Đặt sự kiện click cho nút đăng nhập để gọi phương thức loginUser.

A screen shot of a computer program

Description automatically generated

Phương thức loginUser lấy dữ liệu từ các trường nhập liệu và kiểm tra xem chúng có trống không.

Nếu email hoặc mật khẩu trống, hiển thị thông báo lỗi.

Tạo đối tượng JSON chứa email và mật khẩu.

Chuyển đổi đối tượng JSON thành chuỗi và tạo RequestBody từ chuỗi JSON đó.

Gọi phương thức loginUser của userApi để gửi yêu cầu đăng nhập.

Sử dụng enqueue để gọi API bất đồng bộ và xử lý phản hồi.

Nếu đăng nhập thành công (response.isSuccessful()), hiển thị thông báo thành công và chuyển đến MainActivity.

Nếu không thành công, hiển thị thông báo lỗi và ghi lại lỗi trong log.

Trong trường hợp gặp lỗi kết nối hoặc lỗi khác, xử lý trong phương thức onFailure.  
Lược đồ Sequence:  
A diagram of a sign in a line

Description automatically generated  
Lược đồ Use Case:  
A drawing of a person with arrows and words

Description automatically generated

1.4 Giao diện trang chủ

A screenshot of a phone

Description automatically generated

Code xml gồm các thành phần:

A screenshot of a computer

Description automatically generated

Code activity gồm:

A screen shot of a computer program

Description automatically generated

Khai báo lớp MainActivity kế thừa từ AppCompatActivity.

Khai báo các biến binding, allItems, adapterRecom, adapterNear, và propertyApi để sử dụng view binding, quản lý dữ liệu và gọi API.

A screenshot of a computer program

Description automatically generated

Phương thức onCreate được gọi khi hoạt động được tạo.

Khởi tạo binding với ActivityMainBinding để liên kết với layout.

Thiết lập Retrofit với URL cơ sở và cấu hình GSON.

Khởi tạo propertyApi để gọi API.

Gọi các phương thức initLocation, initTypeButtons, initBottomNavigationButtons, và fetchProperties.

Thiết lập giao diện toàn màn hình.

A computer screen shot of a program

Description automatically generated

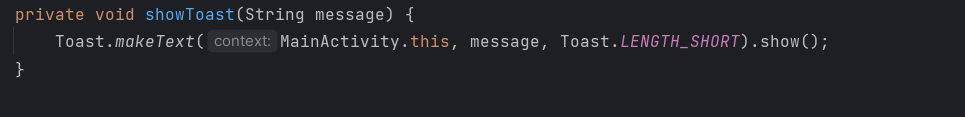
Phương thức fetchProperties gọi API để lấy danh sách địa điểm.

Sử dụng Retrofit để thực hiện yêu cầu và xử lý phản hồi.

Nếu phản hồi thành công, lưu danh sách bất động sản vào allItems và khởi tạo danh sách.

Nếu phản hồi không thành công, hiển thị thông báo lỗi phù hợp.

Trong trường hợp gặp lỗi kết nối hoặc lỗi khác, ghi lại thông tin lỗi và hiển thị thông báo lỗi.



Phương thức showToast hiển thị một thông báo Toast với thông điệp được truyền vào.

A computer screen with white text

Description automatically generated

Phương thức initList khởi tạo danh sách các mục được đề xuất và hiển thị trong recommendedView.

Lọc các mục có điểm số lớn hơn 4.5 và cập nhật giao diện.

Hiển thị tất cả các mục trong nearView.

A screen shot of a computer program

Description automatically generated

Phương thức initTypeButtons thiết lập các nút lọc theo loại địa điểm.

Khi một nút được nhấn, gọi phương thức filterByType với loại tương ứng.

A computer screen shot of a program code

Description automatically generated

Phương thức filterByType lọc danh sách bất động sản theo loại được chọn và cập nhật giao diện.

A computer screen with text

Description automatically generated

Phương thức initLocation thiết lập spinner cho các địa điểm.

Khi một địa điểm được chọn, gọi phương thức filterNearView với địa điểm được chọn.

A computer screen with white text

Description automatically generated

Phương thức filterNearView lọc danh sách bất động sản theo địa điểm được chọn và cập nhật giao diện.

A screen shot of a computer

Description automatically generated

Phương thức updateNearView cập nhật danh sách bất động sản gần địa điểm được chọn trong nearView.

A screen shot of a computer program

Description automatically generated

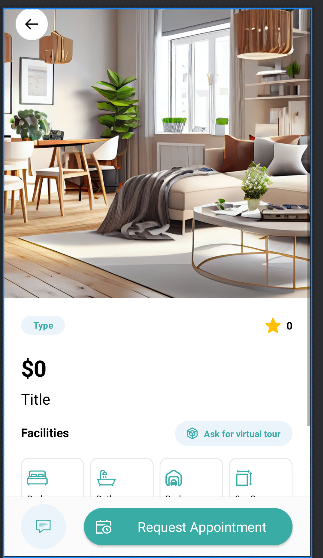
Phương thức initBottomNavigationButtons thiết lập các nút điều hướng dưới cùng.

Khi một nút được nhấn, khởi chạy hoạt động tương ứng và chuyển dữ liệu nếu cần thiết.  
Lược đồ Sequence:  
A diagram of a person with arrows

Description automatically generated  
Lược đồ Use Case:  
A drawing of a person with text

Description automatically generated

1.5 Giao diện hiển thị chi tiết từng sản phẩm



Code xml gồm các thành phần:

A screenshot of a computer

Description automatically generated

Code activity gồm:

A black screen with white text

Description automatically generated

Khai báo lớp DetailActivity kế thừa từ AppCompatActivity.

Khai báo biến binding để sử dụng view binding và biến object để lưu trữ thông tin bất động sản.

A screen shot of a computer program

Description automatically generated

Phương thức onCreate được gọi khi hoạt động được tạo.

Khởi tạo binding với ActivityDetailBinding để liên kết với layout.

Gọi các phương thức getBundles và setVariable để lấy dữ liệu từ Intent và thiết lập các biến.

A computer screen shot of a program code

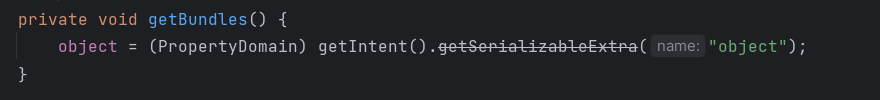
Description automatically generated

Phương thức setVariable thiết lập các biến và sự kiện cho giao diện.

Gắn sự kiện OnClickListener cho nút quay lại (backBtn) để quay lại màn hình trước.

Nếu object không null, sử dụng Glide để tải hình ảnh từ đường dẫn picPath và hiển thị các thông tin khác của bất động sản như tiêu đề, loại, mô tả, giá, số giường, số phòng tắm, diện tích, điểm số, và tình trạng nhà xe.

Gắn sự kiện OnClickListener cho nút đặt phòng (addbtn) để hiển thị thông báo Toast khi người dùng nhấn vào.



Phương thức getBundles lấy đối tượng PropertyDomain từ Intent sử dụng phương thức getSerializableExtra.

Gán đối tượng này cho biến object để sử dụng trong setVariable.

1.6 Giao diện hiển thị toàn bộ địa điểm.

A screenshot of a computer

Description automatically generated

Code xml gồm các thành phần:

A screenshot of a computer

Description automatically generated

Code activity gồm:

A screen shot of a computer program

Description automatically generated

Khai báo lớp ExplorerActivity kế thừa từ AppCompatActivity.

Khai báo các biến cần thiết: recyclerViewExplorer để hiển thị danh sách, itemsAdapter để quản lý dữ liệu và propertyApi để gọi API.

A computer screen shot of a program code

Description automatically generated

Phương thức onCreate được gọi khi hoạt động được tạo.

Liên kết recyclerViewExplorer với layout và thiết lập layout manager để hiển thị danh sách dọc.

Khởi tạo Retrofit với URL cơ sở của API và tạo propertyApi từ Retrofit.

Gọi phương thức fetchAllProperties để lấy dữ liệu từ API.

Liên kết backBtn với nút quay lại và gắn sự kiện OnClickListener để quay lại hoạt động trước đó khi người dùng nhấn nút.

A screen shot of a computer program

Description automatically generated

Phương thức fetchAllProperties gọi API để lấy danh sách các bất động sản.

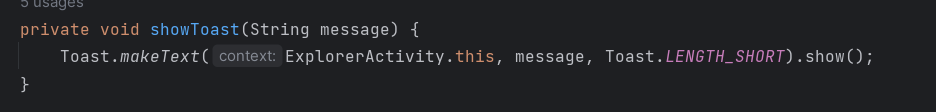
Sử dụng Retrofit để thực hiện cuộc gọi không đồng bộ với enqueue.

Nếu phản hồi thành công và có dữ liệu, tạo một ItemsAdapter với danh sách bất động sản và gán cho recyclerViewExplorer.

Nếu không có dữ liệu, hiển thị thông báo "No properties found".

Xử lý các mã phản hồi khác nhau (404, 500) và hiển thị thông báo tương ứng.

Nếu cuộc gọi thất bại, hiển thị thông báo lỗi.



Phương thức showToast để hiển thị thông báo ngắn bằng Toast.

1.7 Giao diện hiển thị các địa điểm được yêu thích

A screenshot of a computer

Description automatically generated

Code xml gồm các thành phần

A black background with white text

Description automatically generated

Code activity gồm:

A screen shot of a computer program

Description automatically generated

Khai báo lớp FavoriteActivity kế thừa từ AppCompatActivity.

Khai báo các biến cần thiết: recyclerViewFav để hiển thị danh sách, favoriteAdapter để quản lý dữ liệu và propertyApi để gọi API.

A computer screen shot of a program

Description automatically generated

Phương thức onCreate được gọi khi hoạt động được tạo.

Liên kết recyclerViewFav với layout và thiết lập layout manager để hiển thị danh sách dọc.

Khởi tạo Retrofit với URL cơ sở của API và tạo propertyApi từ Retrofit.

Gọi phương thức fetchFavoriteProperties để lấy dữ liệu từ API.

Liên kết backBtn với nút quay lại và gắn sự kiện OnClickListener để quay lại hoạt động trước đó khi người dùng nhấn nút.

A screen shot of a computer

Description automatically generated

Phương thức fetchFavoriteProperties gọi API để lấy danh sách các bất động sản yêu thích.

Sử dụng Retrofit để thực hiện cuộc gọi không đồng bộ với enqueue.

Nếu phản hồi thành công và có dữ liệu, tạo một FavoriteAdapter với danh sách bất động sản yêu thích và gán cho recyclerViewFav.

Nếu không có dữ liệu, hiển thị thông báo "No favorite properties found".

Xử lý các mã phản hồi khác nhau (404, 500) và hiển thị thông báo tương ứng.

Nếu cuộc gọi thất bại, hiển thị thông báo lỗi.

A black background with white text

Description automatically generated

Phương thức showToast để hiển thị thông báo ngắn bằng Toast.

1.8 Giao diện hiển thị thông tin người dùng



Code xml gồm các thành phần:

A screenshot of a computer

Description automatically generated

Code activity gồm

A black background with text

Description automatically generated

Khai báo lớp ProfileActivity kế thừa từ AppCompatActivity.

Khai báo các biến: binding để liên kết với view binding và userApi để gọi API.

A screenshot of a computer program

Description automatically generated

Phương thức onCreate được gọi khi hoạt động được tạo.

Sử dụng view binding để liên kết giao diện người dùng.

Lấy email người dùng từ Intent.

Khởi tạo Retrofit với URL cơ sở của API và tạo userApi.

Gọi phương thức fetchUserData để lấy thông tin người dùng từ API.

Liên kết backBtn với nút quay lại và gắn sự kiện OnClickListener để quay lại hoạt động trước đó khi người dùng nhấn nút.

A screen shot of a computer program

Description automatically generated

Phương thức fetchUserData gửi yêu cầu API để lấy thông tin người dùng dựa trên email.

Sử dụng Retrofit để thực hiện cuộc gọi không đồng bộ với enqueue.

Nếu phản hồi thành công và có dữ liệu, gọi phương thức displayUserData để hiển thị thông tin người dùng.

Nếu không có dữ liệu hoặc cuộc gọi thất bại, hiển thị thông báo lỗi tương ứng.

A screen shot of a computer program

Description automatically generated

Phương thức displayUserData để hiển thị thông tin người dùng trên giao diện.

Thiết lập giá trị cho các TextView trong ActivityProfileBinding với dữ liệu từ đối tượng UserDomain.

A black screen with white text

Description automatically generated

Phương thức showToast để hiển thị thông báo ngắn bằng Toast.

1.10 Code Adapter

A screen shot of a computer program

Description automatically generated

Constructor nhận vào một danh sách các PropertyDomain và một Context. Danh sách này chứa các mục cần hiển thị, trong khi Context được sử dụng để khởi tạo Intent để chuyển đến DetailActivity khi một mục được nhấp.

A screen shot of a computer

Description automatically generated

Phương thức này tạo một ViewHolder mới cho mỗi mục trong danh sách. Nó sử dụng LayoutInflater để inflate layout cho mỗi mục từ ViewholderItemBinding, sau đó trả về một đối tượng ViewHolder.

A computer screen shot of a program code

Description automatically generated

Phương thức này được gọi khi RecyclerView yêu cầu một ViewHolder được gắn với một vị trí cụ thể trong tập dữ liệu.

Nó thiết lập dữ liệu cho mỗi ViewHolder bằng cách lấy mục từ danh sách tại vị trí tương ứng và gán các giá trị thuộc tính của mục vào các thành phần giao diện của ViewHolder.

Sử dụng thư viện Glide để tải ảnh từ URL được cung cấp trong PropertyDomain.

Định nghĩa sự kiện click cho tiêu đề (titleTxt) của mỗi mục để chuyển đến DetailActivity và chuyển dữ liệu của mục được nhấp qua Intent.

Định nghĩa sự kiện click cho nút yêu thích (favbtn) của mỗi mục để thay đổi trạng thái yêu thích của mục và cập nhật giao diện người dùng phản ánh trạng thái mới. Ngoài ra, hiển thị một thông báo ngắn khi nút được nhấp và gọi notifyDataSetChanged() để thông báo rằng dữ liệu đã thay đổi.

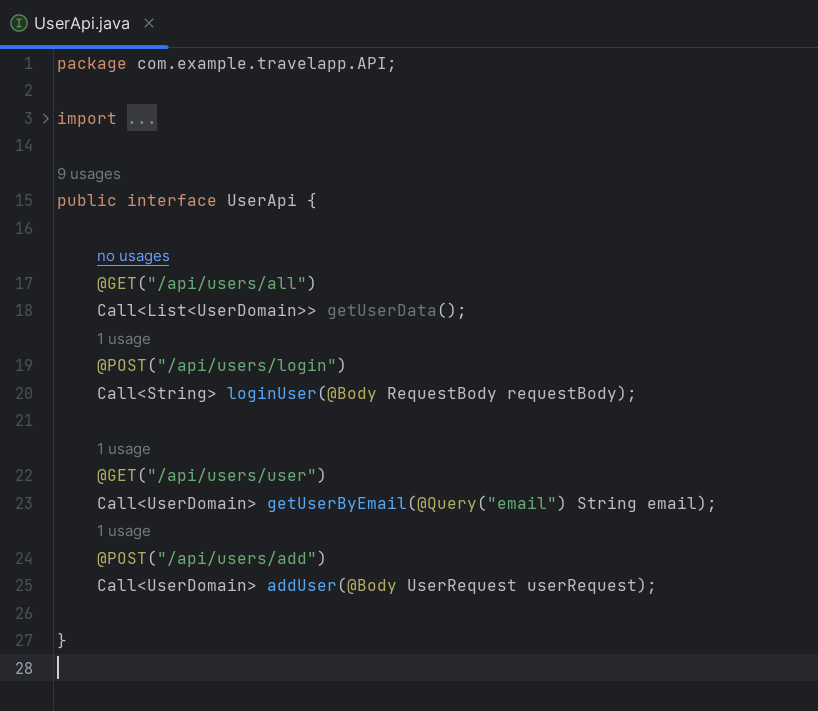
A screen shot of a computer program

Description automatically generated

Phương thức getItemCount: Phương thức này trả về số lượng mục trong danh sách.

Lớp ViewHolder: Lớp này đại diện cho mỗi mục trong RecyclerView. Nó giữ các tham chiếu đến các thành phần giao diện trong mỗi mục, được lưu trữ trong ViewholderItemBinding.

2. API



Định nghĩa các phương thức để giao tiếp với API liên quan đến các user

A screenshot of a computer program

Description automatically generated

Định nghĩa các phương thức để giao tiếp với API liên quan đến các thuộc tính

2.1 Tạo dự án Spring Boot

A screenshot of a computer

Description automatically generated

spring.application.name=travelapp-2: Đặt tên của ứng dụng Spring Boot là travelapp-2.

server.port=9092: Cấu hình cổng mạng mà ứng dụng sẽ lắng nghe các yêu cầu HTTP.

- Cấu hình cơ sở dữ liệu MySQL:

spring.datasource.url=jdbc:mysql://localhost:3306/doanck: URL kết nối đến cơ sở dữ liệu MySQL trên máy cục bộ, cổng mặc định là 3306, và tên cơ sở dữ liệu là doanck.

spring.datasource.username=root: Tên người dùng của cơ sở dữ liệu MySQL.

spring.datasource.password=11892456Ko4!: Mật khẩu để đăng nhập vào cơ sở dữ liệu MySQL.

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver: Tên lớp driver JDBC để sử dụng cho kết nối cơ sở dữ liệu MySQL.

- Cấu hình JPA và Hibernate:

spring.jpa.show-sql=true: Cấu hình hiển thị các truy vấn SQL được tạo ra bởi Hibernate trong quá trình chạy ứng dụng.

spring.jpa.hibernate.ddl-auto=update: Cấu hình Hibernate tự động tạo, cập nhật hoặc không làm gì với cơ sở dữ liệu. Trong trường hợp này, Hibernate sẽ cố gắng cập nhật cơ sở dữ liệu tự động dựa trên các thay đổi trong cấu trúc thực thể (entity).

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect: Xác định loại hình SQL sẽ được Hibernate sử dụng cho cơ sở dữ liệu MySQL. Trong trường hợp này, sử dụng MySQL8Dialect cho MySQL version 8.x.

2.2 API cho properties

- PropertiesController.java: để quản lý các thuộc tính

A screenshot of a computer

Description automatically generated

@RestController: Đánh dấu lớp này là một REST Controller, nơi xử lý các yêu cầu REST.

@RequestMapping("/api"): Chỉ định đường dẫn cơ sở cho tất cả các endpoint trong lớp này.

@Autowired PropertiesService propertiesService: Sử dụng để thêm các thành phần của PropertiesService vào trong Controller.

Các phương thức:

savePro:

A screenshot of a computer program

Description automatically generated  
Mục đích: Endpoint POST để thêm một thuộc tính mới.

Tham số:

propertiesRequest: Đối tượng PropertiesRequest chứa thông tin của thuộc tính cần thêm.

Trả về:

Nếu thành công, trả về mã HTTP 200 OK với thông báo "User added...".

Nếu thất bại, trả về mã HTTP 500 INTERNAL\_SERVER\_ERROR với thông báo "Something went wrong".

getAllPro:

A computer code with text

Description automatically generated

Mục đích: Endpoint GET để lấy danh sách tất cả các thuộc tính.

Trả về:

Nếu danh sách không trống, trả về danh sách các thuộc tính với mã HTTP 200 OK.

Nếu danh sách trống, trả về mã HTTP 204 NO\_CONTENT.

getFavoritePro:

A computer code with text

Description automatically generated

Mục đích: Endpoint GET để lấy danh sách các thuộc tính được đánh dấu là yêu thích.

Trả về:

Nếu danh sách không trống, trả về danh sách các thuộc tính yêu thích với mã HTTP 200 OK.

Nếu danh sách trống, trả về mã HTTP 204 NO\_CONTENT.

updateFavoriteStatus:

A computer code with text

Description automatically generated

Mục đích: Endpoint POST để cập nhật trạng thái yêu thích của một thuộc tính.

Tham số:

title: Tiêu đề của thuộc tính cần cập nhật.

isFavorite: Trạng thái yêu thích mới của thuộc tính (true hoặc false).

Trả về:

Nếu cập nhật thành công, trả về mã HTTP 200 OK với thông báo "Favorite status updated successfully".

Nếu thuộc tính không được tìm thấy, trả về mã HTTP 404 NOT\_FOUND với thông báo "Property not found".

Nếu xảy ra lỗi khác, trả về mã HTTP 500 INTERNAL\_SERVER\_ERROR với thông báo "Failed to update favorite status".

deleteAllPro:

A close-up of a computer code

Description automatically generated

Mục đích: Endpoint GET để xóa tất cả các thuộc tính.

Trả về: Mã HTTP 200 OK với thông báo "All properties deleted successfully" khi xóa thành công.

- Properties.java: là một entity trong ứng dụng của bạn để đại diện cho các thuộc tính

A screen shot of a computer code

Description automatically generated

@Entity: Đánh dấu lớp này là một Entity, tương ứng với bảng trong cơ sở dữ liệu.

@Table(name = "properties\_dbs"): Chỉ định tên bảng trong cơ sở dữ liệu.

Các trường:

id: Một số nguyên được tự động tạo và tăng dần để đại diện cho ID của mỗi thuộc tính.

title: Tiêu đề của thuộc tính.

type: Loại của thuộc tính (ví dụ: nhà, căn hộ, đất đai, v.v.).

address: Địa chỉ của thuộc tính.

description: Mô tả chi tiết về thuộc tính.

picPath: Đường dẫn đến hình ảnh của thuộc tính.

price: Giá của thuộc tính.

bed: Số lượng phòng ngủ trong thuộc tính.

bath: Số lượng phòng tắm trong thuộc tính.

area: Diện tích của thuộc tính.

garage: Biến boolean cho biết liệu thuộc tính có garage hay không.

score: Điểm đánh giá của thuộc tính.

isFav: Biến boolean cho biết liệu thuộc tính có được đánh dấu là yêu thích hay không.

Getters và Setters: Cung cấp các phương thức để truy cập và thiết lập giá trị của các trường.

toString(): Override phương thức này để tạo một chuỗi biểu diễn của đối tượng, giúp dễ dàng khi debug và logging.

- PropertiesRepository.java: là một giao diện (interface) của Spring Data JPA được sử dụng để tương tác với cơ sở dữ liệu cho thực thể Properties. Giao diện này kế thừa từ JpaRepository và cung cấp các phương thức mặc định để thao tác với thực thể Properties

A screenshot of a computer

Description automatically generated

JpaRepository<Properties, Integer>: Kế thừa từ JpaRepository và cung cấp các phương thức CRUD mặc định cho thực thể Properties.

Properties: Thực thể mà repository này sẽ quản lý.

Integer: Kiểu dữ liệu của khóa chính (ID) của thực thể Properties.Các phương thức:

findByIsFavTrue: Tìm kiếm các thuộc tính được đánh dấu là yêu thích.

findByTitle: Tìm kiếm thuộc tính bằng tiêu đề.

- PropertiesRequest.java : được sử dụng để nhận dữ liệu từ yêu cầu của client khi thêm hoặc cập nhật một thuộc tính

A screenshot of a computer

Description automatically generated

Đây là một class POJO (Plain Old Java Object) được sử dụng để nhận dữ liệu từ yêu cầu HTTP, thường là JSON, và chuyển đổi nó thành đối tượng Java.

- ProperetiesServiceImpl.java: một lớp thực hiện các phương thức được định nghĩa trong giao diện PropertiesService. Lớp này tương tác với cơ sở dữ liệu thông qua PropertiesRepository để quản lý các thực thể Properties

A screenshot of a computer program

Description automatically generated

@Service: Annotates lớp này là một lớp service trong Spring.

@Autowired: Injects một instance của PropertiesRepository vào lớp này để sử dụng các phương thức của repository.

savePro: Thêm một thuộc tính mới vào cơ sở dữ liệu.

getAllPro: Lấy danh sách tất cả các thuộc tính.

deleteAllPro: Xóa tất cả các thuộc tính.

getFavoriteProperties: Lấy danh sách các thuộc tính được đánh dấu là yêu thích.

updateFavoriteStatus: Cập nhật trạng thái yêu thích của một thuộc tính.

- PropertiesService.java : Định nghĩa các phương thức cơ bản cho dịch vụ Properties

A screenshot of a computer code

Description automatically generated

savePro(PropertiesRequest propertiesRequest): Lưu một thực thể Properties mới.

getAllPro(): Lấy tất cả các thực thể Properties.

deleteAllPro(): Xóa tất cả các thực thể Properties.

getFavoriteProperties(): Lấy tất cả các thực thể Properties được đánh dấu là yêu thích.

updateFavoriteStatus(String title, boolean isFavorite): Cập nhật trạng thái yêu thích của một thực thể Properties dựa trên tiêu đề.

2.3 API cho User

- UserController: là một lớp điều khiển RESTful trong ứng dụng Spring Boot, có nhiệm vụ xử lý các yêu cầu HTTP liên quan đến người dùng (User). Lớp này sử dụng các dịch vụ được cung cấp bởi UserService để thực hiện các tác vụ như thêm người dùng mới, đăng nhập, lấy thông tin người dùng và lấy tất cả người dùng.

A screenshot of a computer

Description automatically generated

Phương thức addUser:

Mô tả: Phương thức này xử lý yêu cầu POST để thêm một người dùng mới.

Tham số đầu vào: userRequest - một đối tượng JSON đại diện cho thông tin người dùng.

Xử lý: Tạo một đối tượng User từ UserRequest, sau đó gọi userService.addUser(user) để thêm người dùng mới.

Phản hồi: Trả về đối tượng User vừa được thêm cùng với mã trạng thái HTTP 200 (OK).

Phương thức loginUser:

Mô tả: Phương thức này xử lý yêu cầu POST để đăng nhập người dùng.

Tham số đầu vào: userRequest - một đối tượng JSON chứa thông tin đăng nhập (email và mật khẩu).

Xử lý: Gọi userService.authenticateUser(email, password) để xác thực người dùng.

Phản hồi: Trả về thông báo đăng nhập thành công hoặc lỗi đăng nhập dưới dạng JSON. Nếu đăng nhập thành công, trả về mã trạng thái HTTP 200 (OK), nếu không trả về mã trạng thái HTTP 401 (Unauthorized).

Phương thức getUserByEmail:

Mô tả: Phương thức này xử lý yêu cầu GET để lấy thông tin người dùng dựa trên email.

Tham số đầu vào: email - email của người dùng cần lấy thông tin.

Xử lý: Gọi userService.getUserByEmail(email) để lấy thông tin người dùng.

Phản hồi: Trả về đối tượng User nếu tìm thấy, nếu không trả về mã trạng thái HTTP 404 (Not Found).

Phương thức getAllUsers:

Mô tả: Phương thức này xử lý yêu cầu GET để lấy tất cả người dùng.

Tham số đầu vào: Không có.

Xử lý: Gọi userService.getAllUsers() để lấy danh sách tất cả người dùng.

Phản hồi: Trả về danh sách các đối tượng User cùng với mã trạng thái HTTP 200 (OK).

Các thành phần khác

Autowired: @Autowired được sử dụng để tự động tiêm đối tượng UserService vào UserController.

ResponseEntity: Được sử dụng để tạo đối tượng phản hồi HTTP với mã trạng thái và dữ liệu trả về.

Gson: Được sử dụng để chuyển đổi thông điệp phản hồi thành định dạng JSON.

- User.java: sử dụng các chú thích từ JPA (Java Persistence API) để xác định nó là một thực thể và ánh xạ các trường của nó tới các cột trong bảng cơ sở dữ liệu.

A screenshot of a computer

Description automatically generated

@Entity: Xác định lớp này là một thực thể JPA.

@Table(name = "user\_dbs"): Chỉ định bảng trong cơ sở dữ liệu mà thực thể này được ánh xạ tới, ở đây là bảng user\_dbs.

@Id: Đánh dấu trường id là khóa chính của thực thể.

@GeneratedValue(strategy = GenerationType.IDENTITY): Chỉ định rằng giá trị của id sẽ được tự động tạo ra bởi cơ sở dữ liệu khi một bản ghi mới được thêm.

@Column(name = "Id"): Ánh xạ trường id tới cột Id trong bảng user\_dbs

Mỗi trường được ánh xạ tới một cột tương ứng trong bảng user\_dbs.

@Column(name = "column\_name"): Xác định tên cột trong cơ sở dữ liệu mà trường dữ liệu được ánh xạ tới.

Các phương thức getter và setter

Các phương thức này được sử dụng để truy xuất và cập nhật giá trị của các trường dữ liệu

Phương thức toString cung cấp một cách để biểu diễn đối tượng User dưới dạng chuỗi, thuận tiện cho việc in ấn hoặc ghi log.

- UserRepository.java: được sử dụng để quản lý các thao tác CRUD (Create, Read, Update, Delete) cho thực thể User trong cơ sở dữ liệu. Nó mở rộng từ JpaRepository của Spring Data JPA, cung cấp một bộ công cụ đầy đủ để làm việc với thực thể User

A screenshot of a computer

Description automatically generated

@Repository: Chú thích này đánh dấu interface này là một bean Spring của lớp Repository, điều này giúp Spring quản lý nó như một phần của tầng truy cập dữ liệu.

extends JpaRepository<User, Integer>: JpaRepository là một interface từ Spring Data JPA, cung cấp các phương thức CRUD cơ bản. Tham số đầu tiên là kiểu của thực thể (User), và tham số thứ hai là kiểu dữ liệu của khóa chính (Integer).

User findByEmail(String email): Phương thức này được định nghĩa để tìm kiếm người dùng bằng email. Spring Data JPA tự động tạo ra truy vấn SQL tương ứng dựa trên tên của phương thức.

- UserRequest.java: được sử dụng để chuyển dữ liệu từ phía client đến server trong các yêu cầu HTTP, thường được sử dụng trong các controller để nhận dữ liệu từ client.

A screenshot of a computer

Description automatically generated

Khai báo các thuộc tính

name: Tên người dùng.

fullName: Họ và tên đầy đủ của người dùng.

email: Địa chỉ email của người dùng.

phone: Số điện thoại của người dùng.

address: Địa chỉ của người dùng.

password: Mật khẩu của người dùng.

Các phương thức getter và setter cho phép truy cập và thay đổi giá trị của các thuộc tính.

Phương thức toString: Phương thức này trả về chuỗi đại diện của đối tượng UserRequest, bao gồm tất cả các thuộc tính. Điều này hữu ích cho việc ghi log hoặc debug.

- UserService.java: định nghĩa các phương thức cần thiết để thao tác với các đối tượng User

A screenshot of a computer

Description automatically generated

Phương thức addUser: Thêm một người dùng mới vào hệ thống.

Phương thức getAllUsers: Lấy danh sách tất cả người dùng trong hệ thống.

Phương thức authenticateUser: Xác thực người dùng dựa trên email và mật khẩu.

Phương thức getUserByEmail: Lấy thông tin người dùng dựa trên email.

- UserServiceImpl.java: là một lớp triển khai của UserService trong ứng dụng của bạn, thực hiện các phương thức để thao tác với các đối tượng User bằng cách sử dụng UserRepository.

A screenshot of a computer

Description automatically generated

Chức năng: UserRepository là một phụ thuộc được Spring quản lý, dùng để thao tác với cơ sở dữ liệu.

Annotation: @Autowired dùng để tiêm phụ thuộc tự động.

Phương thức addUsers: Thêm một người dùng mới vào cơ sở dữ liệu.

Phương thức getAllUsers: Lấy danh sách tất cả người dùng trong cơ sở dữ liệu.

Phương thức authenticateUser: Xác thực người dùng dựa trên email và mật khẩu.

Phương thức getUserByEmail: Lấy thông tin người dùng dựa trên email.

3. Các quyền và dependencies



yêu cầu quyền truy cập Internet.

A screen shot of a black and white screen

Description automatically generated

cho phép việc sử dụng View Binding.

A screenshot of a computer program

Description automatically generated

com.github.bumptech.glide:glide:4.12.0: Glide là một thư viện dùng để tải ảnh và video từ nhiều nguồn khác nhau như Internet, Firebase, và cả local.

com.squareup.retrofit2:retrofit:2.9.0 và com.squareup.retrofit2:converter-gson:2.9.0: Retrofit là một thư viện HTTP client được sử dụng để tạo và thực hiện các yêu cầu RESTful trên mạng. converter-gson là một bộ chuyển đổi (converter) cho Retrofit sử dụng Gson để chuyển đổi JSON thành các đối tượng Java và ngược lại.

com.google.android.gms:play-services-base:18.4.0: Google Play Services là một bộ công cụ cung cấp các tính năng như truy cập vào Google Maps, Google Drive, phân tích, quảng cáo và nhiều tính năng khác.

com.google.firebase:firebase-bom:33.0.0, com.google.firebase:firebase-analytics, com.google.firebase:firebase-storage-ktx: Firebase là một nền tảng phát triển ứng dụng di động và web do Google cung cấp. Bom là tập hợp (bill of materials) các thư viện Firebase giúp quản lý phiên bản của các thư viện này.

com.github.bumptech.glide:compiler:4.12.0: Glide Compiler là một công cụ được sử dụng trong quá trình biên dịch để tạo ra mã Java tối ưu hóa cho việc tải ảnh với Glide.