BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN
TRƯỜNG ĐẠI HỌC THỦY LỢI

BÀI TẬP LỚN

HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG

ĐỀ TÀI: Xây dựng ứng dụng dự báo thời tiết

Mã Sinh Viên	Họ và Tên	Ngày Sinh	Lớp
2151172783	Nguyễn Quốc Trường	27/9/2003	63KTPM2
2151170568	Nguyễn Xuân Thức	01/07/2003	63KTPM2
2151173827	Đoàn Mạnh Thưởng	9/12/2003	63KTPM2
2151170590	Thái Vương Kiên	16/052003	63KTPM2

Hà Nội, năm 2024

BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN
TRƯỜNG ĐẠI HỌC THỦY LỢI

BÀI TẬP LỚN

HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỀ TÀI: Xây dựng ứng dụng dự báo thời tiết

Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
			Bằng Số	Bằng Chữ
2151172783	Nguyễn Quốc Trường	27/9/2003		
2151170568	Nguyễn Xuân Thức	01/07/2003		
2151173827	Đoàn Mạnh Thưởng	9/12/2003		
2151170590	Thái Vương Kiên	16/052003		

CÁN BỘ CHẤM THI 1 **CÁN BỘ CHẤM THI 2**

Hà Nội, năm 2024

LỜI NÓI ĐẦU

Trong bối cảnh thời tiết ngày càng trở nên khó lường và có nhiều biến động do biến đổi khí hậu, việc nắm bắt thông tin thời tiết chính xác và kịp thời trở thành nhu cầu thiết yếu của con người. Thông tin thời tiết không chỉ giúp chúng ta chuẩn bị cho các hoạt động hằng ngày mà còn đóng vai trò quan trọng trong việc bảo vệ an toàn trước những hiện tượng thiên nhiên bất thường.

Dự án **Ứng dụng Dự Báo Thời Tiết** trên nền tảng Android ra đời nhằm mục đích cung cấp một công cụ hữu ích cho người dùng trong việc tra cứu và cập nhật tình hình thời tiết theo thời gian thực. Ứng dụng không chỉ hiển thị thông tin thời tiết hiện tại mà còn cung cấp dự báo thời tiết cho những ngày tiếp theo, giúp người dùng dễ dàng lập kế hoạch và đưa ra các quyết định phù hợp cho các hoạt động ngoài trời.

Với giao diện thân thiện, dễ sử dụng, cùng với khả năng cung cấp các cảnh báo thời tiết khẩn cấp, chúng tôi tin rằng ứng dụng này sẽ trở thành một trợ thủ đắc lực trong đời sống hằng ngày của người dùng. Trong suốt quá trình phát triển, nhóm dự án đã nỗ lực không ngừng để mang đến sản phẩm chất lượng, đồng thời áp dụng những công nghệ mới nhất nhằm đảm bảo hiệu suất cao và tính ổn định.

Chúng tôi xin chân thành cảm ơn sự giúp đỡ của các thầy cô, đồng nghiệp và bạn bè đã tạo điều kiện thuận lợi để dự án này được hoàn thành đúng tiến độ. Chúng tôi cũng rất mong nhận được những ý kiến đóng góp quý báu từ người dùng để hoàn thiện sản phẩm hơn nữa trong tương lai.

MỤC LỤC

I. MÔ TẢ BÀI TOÁN

1. Giới thiệu

Ứng dụng dự báo thời tiết là một công cụ hữu hiệu để người dùng theo dõi các thông tin thời tiết, cảnh báo thiên tai được dự báo trong tương lai trong và ngoài khu vực với độ chính xác cao. Ứng dụng được xây dựng bằng ngôn ngữ Java.

2. Các chức năng chính

- **Dự báo thời tiết theo vị trí hiện tại**: Người dùng sau khi cấp quyền cho ứng dụng xác định vị trí hiện tại sẽ tự động xác định vị trí của người dùng và hiển thị thông tin thời tiết tại khu vực.
- **Tìm kiếm địa điểm khác**: Người dùng có thể tìm kiếm và xem thông tin thời tiết của các thành phố khác nhau.
- **Hiển thị dự báo theo thời gian thực**: Dự báo thời tiết chi tiết theo giờ, ngày (trong vòng 1 - 5 ngày).
- **Lưu và xóa các địa điểm:** Người dùng có thể lưu để xem thời tiết thành phố, và cũng có thể xóa đi khi không xem nữa.
- **Thông báo cảnh báo**: Ứng dụng sẽ gửi cảnh báo đẩy (push notifications) khi có thời tiết xấu hoặc các hiện tượng thiên nhiên nguy hiểm.

3. Các yêu cầu phi chức năng

- **Dễ sử dụng:** Giao diện cần có bố cục rõ ràng. dể hiểu, dễ sử dụng.
- **Độ tin cậy:** Ứng dụng phải được sự cho phép của người dùng để có thể thực hiện một số chức năng liên quan tới định vị.
- **Hiệu năng:** Ứng dụng cần hoạt động ổn định, nhanh chóng, mượt mà.
- **Màu sắc:** Màu sắc cần phù hợp, không làm ảnh hưởng xấu tới quá trình sử dụng của người dùng.

4. Công nghệ sử dụng

- **Ngôn ngữ lập trình**: Java.
- **IDE**: Android Studio.
- **API thời tiết**: OpenWeather
- **Thư viện và công nghệ**:
  - **OpenWeather**: Để lấy dữ liệu từ API.
  - **Real time database firebase**: Để lưu trữ dữ liệu thời tiết cục bộ.
  - **Recycle view và ViewModel**: Sử dụng cho quản lý dữ liệu và cập nhật UI.
  - **Google Play Services (Location Services)**: Để xác định vị trí người dùng.

5. Yêu cầu hệ thống

- **Phiên bản Android: Ứng dụng sẽ hỗ trợ từ Android 8.0 trở lên.Yêu cầu phần cứng:**

- Thiết bị Android có kết nối Internet để cập nhật thời tiết theo thời gian thực.
- Thiết bị Android phải có phần cứng GPS để thực hiện việc xem thời tiết theo vị trí hiện tại.
- Thiết bị Android phải từ Android 8.0 đổ lên.

6. Nguy cơ và rủi ro

API thời tiết không ổn định: Nên có chiến lược xử lý lỗi khi không kết nối được API hoặc dữ liệu không chính xác.

Pin của thiết bị: Ứng dụng có thể ảnh hưởng đến thời lượng pin do việc sử dụng GPS và kết nối mạng liên tục.

Quyền riêng tư: Phải đảm bảo xử lý quyền vị trí của người dùng một cách hợp lý và bảo mật dữ liệu người dùng.

7. Kế hoạch bảo trì

- Cập nhật thường xuyên để tối ưu hiệu suất và sửa lỗi.
- Đảm bảo cập nhật dữ liệu từ API mới nhất và quản lý cảnh báo thời tiết kịp thời.

8. Bảng phân công nhiệm vụ:

STT	Họ và tên	Phân chia công việc
1	Nguyễn Quốc Trường	- Setup firebase , Api OpenWeather cho Project - Notification hàng ngày vào 8h sáng - Chức năng thêm địa điểm, thành phố cần xem thời tiết - Chức năng xóa địa điểm, thành phố cần xem thời tiết - Chức năng lưu lịch sử thời tiết để sử dụng lúc api Lỗi hoặc không kết nối internet
2	Nguyễn Xuân Thức	- Chức năng dự báo thời tiết chi tiết cho 12 giờ tới - Chức năng dự báo thời tiết cho 1- 5 ngày tới - Chức năng cảnh báo lũ lụt thiên tai trong ứng dụng tương ứng với vị trí
3	Đoàn Mạnh Thưởng	- Chức năng xem thời tiết theo vị trí hiện tại - Chức năng thay đổi hình nền theo thời tiết - Chức năng xem bản đồ.
4	Thái Vương Kiên	- Chức năng tìm kiếm thời tiết theo thành phố, địa điểm - Chức năng xem thời tiết theo thành phố, địa điểm

II. PHÂN TÍCH YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG

1. Phân tích yêu cầu

Xác định người dùng:

- Người dùng cuối của ứng dụng dự báo thời tiết là bất kỳ ai có nhu cầu cập nhật thông tin thời tiết hàng ngày. Điều này bao gồm học sinh, sinh viên, người đi làm, hoặc bất kỳ ai cần theo dõi tình hình thời tiết để lên kế hoạch cho các hoạt động ngoài trời.
- Ứng dụng phải dễ sử dụng và có giao diện trực quan, vì đối tượng người dùng có thể có mức độ hiểu biết về công nghệ khác nhau.

Thu thập yêu cầu:

- Dựa trên các nhu cầu thường gặp về dự báo thời tiết, ta xác định các chức năng chính của ứng dụng:
  - Hiển thị thời tiết hiện tại: Nhiệt độ, độ ẩm, tốc độ gió, trạng thái thời tiết (mưa, nắng, mây, v.v.).
  - Dự báo thời tiết trong tương lai: Dự báo theo ngày và theo giờ.
  - Tìm kiếm địa điểm khác: Cung cấp khả năng tra cứu thời tiết cho nhiều địa điểm khác nhau.
  - Cảnh báo thời tiết: Thông báo khi có các hiện tượng thời tiết nguy hiểm (bão, lũ, nắng nóng gay gắt, v.v.).
  - Lưu trữ lịch sử thời tiết: Giúp người dùng xem lại dữ liệu thời tiết đã tra cứu.
- Yêu cầu phi chức năng của ứng dụng bao gồm:
  - Dễ sử dụng: Ứng dụng cần dễ dàng thao tác cho mọi đối tượng người dùng, kể cả những người không quen thuộc với công nghệ.
  - Hiệu năng: Ứng dụng cần phản hồi nhanh chóng khi người dùng yêu cầu tra cứu thông tin thời tiết.
  - Độ tin cậy: Ứng dụng phải cung cấp thông tin thời tiết chính xác và cập nhật kịp thời.

Phân tích yêu cầu:

- **Xác định các tác nhân:** Người dùng
- **Đặc tả chức năng**
  1. **Hiển thị thời tiết vị trí hiện tại:**
     - Ứng dụng sẽ tự động xác định vị trí của người dùng bằng cách sử dụng GPS
     - Thông tin thời tiết hiển thị gồm:
       - Nhiệt độ hiện tại.
       - Tình trạng thời tiết (nắng, mưa, mây, v.v.).
       - Độ ẩm.
       - Tốc độ và hướng gió.

- Dữ liệu này cần được cập nhật liên tục theo thời gian thực từ API thời tiết .

2. **Dự báo thời tiết trong tương lai:**

2.1. Hiển thị thời tiết theo các tùy chọn trong khoảng từ 1 - 5 ngày

- Ứng dụng sẽ cung cấp dự báo thời tiết trong 1-5 ngày tới, bao gồm:
  - Nhiệt độ cao nhất và thấp nhất trong ngày.
  - Tình trạng thời tiết dự kiến (nắng, mưa, bão, v.v.).
- Dự báo theo giờ: Dự báo nhiệt độ, khả năng mưa và tình trạng mây trong ngày.
- Người dùng có thể cuộn qua các ngày để xem thông tin chi tiết.

2.2. Chức năng xem thời tiết trong khoảng thời gian từ 1 - 12 giờ

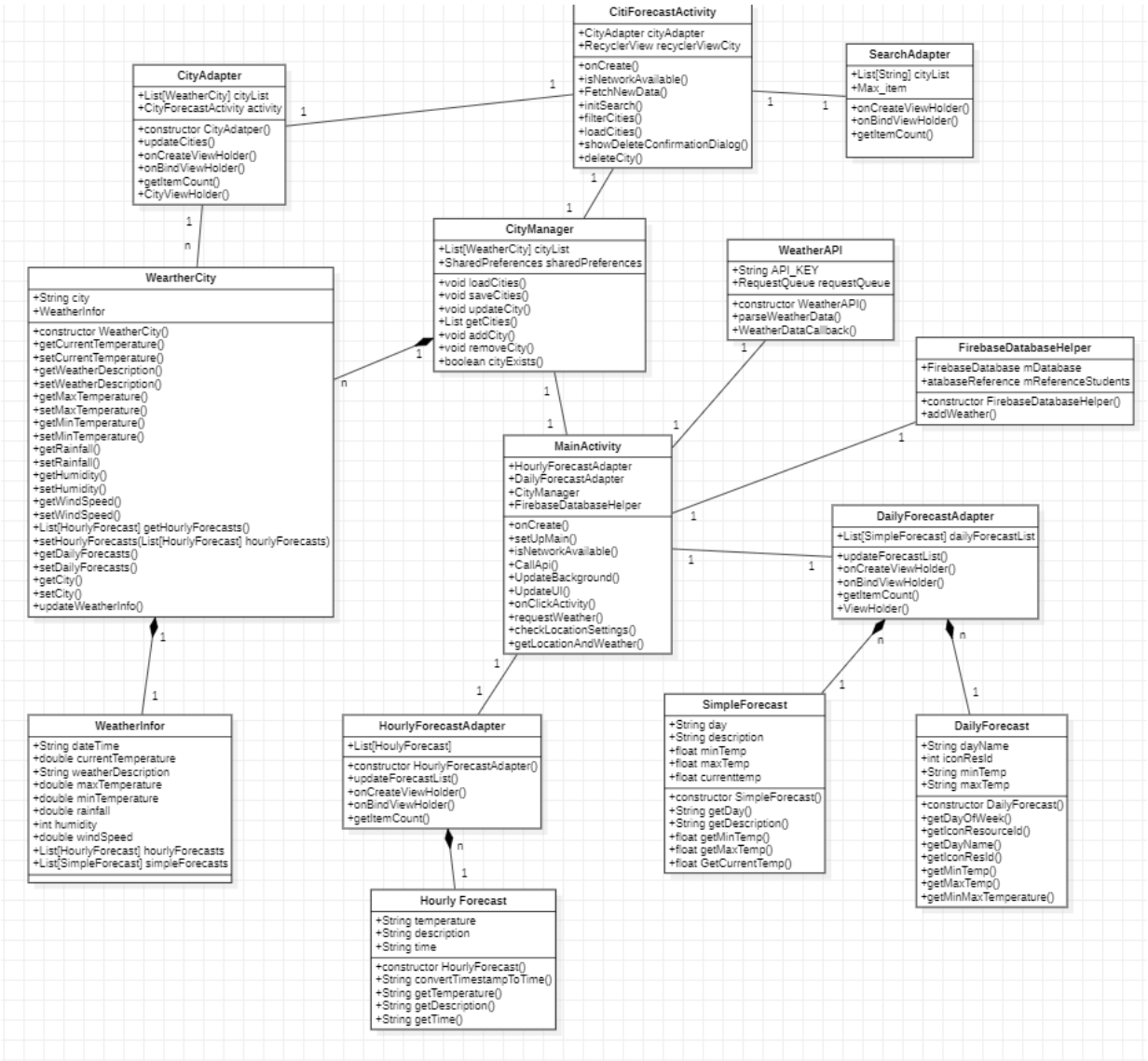
- Ứng dụng sẽ cung cấp dự báo thời tiết trong 1-12 giờ tới, bao gồm:
  - Nhiệt độ tại thời điểm tiếp theo.
  - Tình trạng thời tiết trong thời điểm tiếp theo.
  - Dự báo theo giờ: Dự báo nhiệt độ, khả năng mưa và tình trạng mây cho từng giờ.
- Người dùng có thể cuộn qua các giờ để xem thông tin chi tiết.

2.3. Chức năng thông báo khi có lũ lụt, thiên tai: Ứng dụng sẽ phát 1 thông báo nếu vị trí hiện tại có khả năng xảy ra lũ lụt, thiên tai.

3. **Tìm kiếm và xem thời tiết theo thành phố, địa điểm**
   - Ứng dụng cho phép người dùng tìm kiếm và xem thời tiết tại các địa điểm khác bằng cách nhập tên thành phố, quốc gia hoặc khu vực.
   - Thông tin thời tiết sẽ được lấy từ API dựa trên địa điểm mà người dùng đã nhập.
   - Người dùng có thể lưu các địa điểm yêu thích để dễ dàng truy cập nhanh sau này.
4. **Lưu trữ và sửa đổi địa điểm:**
   - Ứng dụng sẽ lưu trữ các dữ liệu liên quan tới địa điểm mà người dùng đã tra cứu để có thể xem lại sau, ngay cả khi không có kết nối Internet.
   - Lưu lịch sử thời tiết trên Firebase để lưu trữ địa điểm.

2. Thiết kế hệ thống:

Biểu đồ Class Diagram:



Thiết kế kiến trúc (MVC):

Models

- **CityForecast:** Đại diện cho dự báo thời tiết của một thành phố, lưu các thông tin chính như tên thành phố, nhiệt độ hiện tại, nhiệt độ cao nhất, nhiệt độ thấp nhất, và thông tin mô tả thời tiết.
- **WeatherInfo:** Lưu trữ thông tin thời tiết chi tiết, bao gồm nhiệt độ hiện tại, thông tin thời tiết, nhiệt độ cao nhất, nhiệt độ thấp nhất, lượng mưa, độ ẩm, tốc độ gió, và danh sách dự báo thời tiết theo giờ và theo ngày.
- **WeatherCity:** Đại diện cho thời tiết của một thành phố, kết hợp với WeatherInfo để lưu trữ thông tin thời tiết hiện tại của thành phố, danh sách dự báo theo giờ (HourlyForecast) và danh sách dự báo thời tiết trong ngày (SimpleForecast).
- **SimpleForecast:** Đại diện cho dự báo thời tiết trong ngày với các thông tin như ngày, mô tả thời tiết, nhiệt độ hiện tại, nhiệt độ cao nhất và thấp nhất.

- **HourlyForecast:** Dự báo thời tiết theo giờ, với các thông tin thời gian, hình ảnh minh họa, nhiệt độ và thông tin thời tiết.

View

- **activity_city_manager:** Giao diện cho phép quản lý các thành phố trong ứng dụng với chức năng tìm kiếm, thêm, và xóa thành phố. Giao diện gồm tiêu đề, ô tìm kiếm, và danh sách thành phố được hiển thị dưới dạng các item.
- **activity_main:** Giao diện chính để hiển thị thông tin thời tiết của thành phố và là nơi tương tác với các giao diện khác.
- **activity_map:** Giao diện bản đồ, hỗ trợ người dùng trong việc xác định và lựa chọn vị trí thành phố.
- **city_item:** Item giao diện trong activity_city_manager, hiển thị các thông tin thành phố như tên, mô tả thời tiết, nhiệt độ cao nhất và thấp nhất.
- **item_city_manager_forecast:** Item giao diện trong activity_city_manager, hiển thị các thông tin thành phố như tên, mô tả thời tiết, nhiệt độ cao nhất và thấp nhất.
- **history_item:** Hiển thị lịch sử các thông tin thời tiết được lưu lại.
- **hourly_forecast_item:** Item trong activity_main, hiển thị dự báo thời tiết theo giờ gồm thời gian, hình ảnh minh họa, nhiệt độ, thông tin và mô tả thời tiết.
- **item_daily_forecast:** Item trong activity_main, hiển thị dự báo thời tiết hàng ngày với thời gian, hình ảnh minh họa, nhiệt độ, thông tin và mô tả.
- **disaster_warning_popup:** Popup hiển thị cảnh báo thiên tai trong ứng dụng, cung cấp thông tin quan trọng về các tình huống khẩn cấp đến người dùng với tiêu đề, nội dung cảnh báo, và nút đóng.
- **simple_item_search:** Phục vụ cho việc tìm kiếm hoặc điều hướng trong ứng dụng thời tiết, giúp người dùng dễ dàng tương tác để tìm kiếm thông tin về các thành phố.

Controller

- **CityAdapter:** Xử lý các yêu cầu từ người dùng thông qua WeatherCity. Class này tương tác với WeatherCity, gọi các phương thức cần thiết để hiển thị thông tin thời tiết của thành phố trong activity_main.
- **WeatherAPI:** Đảm nhiệm việc giao tiếp với API OpenWeather. Class này chứa API key và các phương thức để kết nối, truy xuất dữ liệu thời tiết, và ghi nhận các thông tin cần thiết để cập nhật dữ liệu vào ứng dụng.
- **DailyForecast:** Dự báo thời tiết cho những ngày tiếp theo, bao gồm các thông tin ngày trong tuần, hình ảnh minh họa thời tiết, và nhiệt độ cao nhất, thấp nhất.
- **CityManager:** Quản lý thành phố, xử lý các yêu cầu thêm và xóa thành phố từ người dùng thông qua WeatherCity và phản hồi kết quả để hiển thị trên activity_main.
- **CitySearch:** Xử lý các yêu cầu tìm kiếm thành phố thông qua SearchView.
- **CityForecastActivity và CityForecastAdapter:** Quản lý hoạt động dự báo thời tiết cho một thành phố cụ thể và cung cấp bộ điều hợp (adapter) để hiển thị dữ liệu dự báo trên giao diện.
- **CitySearchAdapter:** Adapter giúp hiển thị các kết quả tìm kiếm thành phố.

- **CurrentLocationActivity:** Xử lý thông tin dự báo thời tiết cho vị trí hiện tại của người dùng.
- **DailyForecastAdapter:** Adapter cho phép hiển thị danh sách dự báo thời tiết hàng ngày.
- **FirebaseDatabaseHelper:** Class hỗ trợ tương tác với Firebase để lưu và truy xuất thông tin thời tiết.
- **MainActivity:** Giao diện chính điều hướng đến các hoạt động và chức năng khác của ứng dụng.
- **SharePreferenceControl:** Hỗ trợ lưu trữ các cài đặt ứng dụng hoặc thông tin cần thiết để hiển thị lại khi khởi động lại ứng dụng.
- **Hourly Forecast Activity** cung cấp thông tin thời tiết chi tiết theo từng giờ, giúp người dùng dễ dàng lên kế hoạch và chuẩn bị cho các hoạt động trong ngày, tránh được các thay đổi đột ngột của thời tiết.

Dựa trên kiến trúc MVC trên, Ta có thể xác định các lớp sau:

CityAdapter: Adapter cho RecyclerView, hiển thị danh sách WeatherCity (thành phố và thông tin thời tiết).

- **Thuộc tính**:
  - cityList: Danh sách các WeatherCity.
- **Phương thức**:
  - CityAdapter(): Khởi tạo adapter với danh sách thành phố.
  - updateCities(): Cập nhật danh sách thành phố mới.
  - onCreateViewHolder(): Tạo ViewHolder.
  - onBindViewHolder(): Gán dữ liệu cho item.
  - getItemCount(): Trả về số lượng thành phố.

CityViewHolder: ViewHolder đại diện cho từng thành phố trong RecyclerView.

- **Thuộc tính**:
  - dayName, currentTemp, minTemp, maxTemp: Thông tin thời tiết.
  - weatherIcon: Biểu tượng thời tiết.
  - deleteButton: Xóa thành phố.
- **Phương thức**:
  - CityViewHolder(): Khởi tạo các thành phần giao diện.
  - bind(): Gán dữ liệu và xử lý xóa khi nhấn deleteButton.

CityForecast: Lớp đại diện cho dự báo thời tiết của một thành phố.

- **Thuộc tính:**
  - **cityName:** Tên thành phố.
  - **temperature:** Nhiệt độ hiện tại.
  - **highTemp:** Nhiệt độ cao nhất.

    - **lowTemp:** Nhiệt độ thấp nhất.
    - **description:** Mô tả thời tiết.
- **Phương thức:**
    - **Getters** (getCityName(), getTemperature(), getHighTemp(), getLowTemp(), getDescription()): Trả về giá trị của từng thuộc tính.

CityForecastActivity: Lớp hiển thị dự báo thời tiết của các thành phố.

- **Thuộc tính**:
    - cityManager: Quản lý danh sách các thành phố và thêm dữ liệu thành phố mới.
    - cityAdapter: Adapter cho RecyclerView để hiển thị danh sách các WeatherCity.
- **Phương thức**:
    - onCreate(Bundle savedInstanceState): Khởi tạo giao diện, thiết lập RecyclerView với CityAdapter. Nếu danh sách thành phố trống, sử dụng WeatherAPI để tải dữ liệu thời tiết cho thành phố mặc định ("Ha Noi") và cập nhật lại RecyclerView.

CityForecastAdapter: Adapter cho RecyclerView, hiển thị danh sách dự báo thời tiết của các thành phố (CityForecast).

- **Thuộc tính**:
    - cityForecastList: Danh sách các đối tượng CityForecast.
- **Phương thức**:
    - CityForecastAdapter(): Khởi tạo adapter với danh sách CityForecast.
    - onCreateViewHolder(): Tạo ViewHolder cho mỗi item trong danh sách.
    - onBindViewHolder(): Gán dữ liệu CityForecast vào ViewHolder.
    - getItemCount(): Trả về số lượng CityForecast trong danh sách.

CityForecastViewHolder: ViewHolder đại diện cho từng item CityForecast trong RecyclerView.

- **Thuộc tính**:
    - txtCityName, txtTemp, txtHigh, txtLow, txtDescription: Hiển thị thông tin thời tiết.
    - button: Nút bấm chuyển đến MainActivity.
- **Phương thức**:
    - CityForecastViewHolder(): Khởi tạo các thành phần giao diện.
    - bind(): Gán dữ liệu và thiết lập sự kiện OnClickListener cho button để chuyển sang MainActivity với tên thành phố đã chọn.

CityManager: Lớp quản lý danh sách các thành phố (WeatherCity) và lưu trữ dữ liệu trong SharedPreferences để duy trì trạng thái của ứng dụng.

- **Thuộc tính**:
    - instance: Thể hiện duy nhất của CityManager (theo mô hình singleton).

    - PREFS_NAME, CITIES_KEY: Khóa để lưu và truy xuất dữ liệu từ SharedPreferences.
    - sharedPreferences: Đối tượng lưu trữ thông tin thời tiết của các thành phố.
    - cityList: Danh sách các thành phố đã lưu.
- **Phương thức**:
    - CityManager(): Khởi tạo CityManager và tải dữ liệu các thành phố từ SharedPreferences.
    - loadCities(): Tải danh sách thành phố từ SharedPreferences sử dụng Gson.
    - saveCities(): Lưu danh sách thành phố vào SharedPreferences dưới dạng JSON.
    - getCities(): Trả về danh sách thành phố hiện tại.
    - addCity(): Thêm thành phố mới vào danh sách và lưu lại.
    - removeCity(): Xóa thành phố khỏi danh sách và lưu lại.

CitySearch: Lớp hoạt động tìm kiếm thành phố.

- **Thuộc tính**:
    - searchView: Thanh tìm kiếm để người dùng nhập tên thành phố.
- **Phương thức**:
    - onCreate(): Thiết lập giao diện người dùng và gọi setupSearchView().
    - setupSearchView(): Đăng ký lắng nghe sự kiện tìm kiếm.
        - **onQueryTextSubmit()**: Khi người dùng nhập xong và nhấn Enter, một Intent sẽ được khởi tạo để mở CitySearchAdapter, truyền tên thành phố vừa nhập (query) qua Intent.
        - **onQueryTextChange()**: Tùy chọn xử lý khi văn bản trong thanh tìm kiếm thay đổi.

CitySearchAdapter: Lớp adapter cho RecyclerView hiển thị danh sách tên các thành phố tìm kiếm.

- **Thuộc tính**:
    - cityList: Danh sách các thành phố.
- **Phương thức**:
    - onCreateViewHolder(): Tạo ViewHolder từ city_item.xml.
    - onBindViewHolder(): Gán tên thành phố từ cityList vào cityTextView của ViewHolder.
    - getItemCount(): Trả về số lượng thành phố trong cityList.
- **ViewHolder**:
    - cityTextView: Hiển thị tên thành phố trong từng mục của danh sách.

CurrentLocationWeatherActivity: Lớp Activity để lấy vị trí hiện tại của người dùng và hiển thị thời tiết tương ứng.

- **Thuộc tính**:
    - fusedLocationProviderClient: Để lấy vị trí hiện tại.

- tvLocation, tvTemperature, tvStatus: TextView hiển thị thông tin về vị trí và thời tiết.
- btnGetWeather: Nút để người dùng lấy thông tin thời tiết.
- API_KEY: Khóa API cho dịch vụ thời tiết.
- REQUEST_LOCATION_PERMISSION, REQUEST_CHECK_SETTINGS: Mã yêu cầu cho quyền truy cập vị trí và kiểm tra cài đặt vị trí.

- **Phương thức**:
  - onCreate(): Khởi tạo Activity và thực hiện kiểm tra quyền truy cập vị trí.
  - fetchWeather(): Kiểm tra quyền truy cập vị trí và gọi phương thức kiểm tra cài đặt vị trí.
  - checkLocationSettings(): Kiểm tra cài đặt vị trí và gọi getLocationAndWeather() nếu tất cả cài đặt đều thoả mãn.
  - getLocationAndWeather(): Lấy vị trí của người dùng và gọi phương thức để chuyển đổi tọa độ thành tên thành phố.
  - getCityNameFromLocation(): Chuyển đổi tọa độ thành tên thành phố sử dụng Geocoder.
  - fetchWeatherForCity(): Gọi API thời tiết với tên thành phố đã tìm được.
  - onRequestPermissionsResult(): Xử lý kết quả yêu cầu quyền truy cập vị trí.
  - onActivityResult(): Xử lý kết quả kiểm tra cài đặt vị trí.
  - onFailure(): Xử lý lỗi khi không thể kiểm tra cài đặt vị trí.

DailyForecast: Lớp đại diện cho dự báo thời tiết hàng ngày, bao gồm thông tin về ngày, biểu tượng thời tiết, và nhiệt độ tối thiểu và tối đa.

- **Thuộc tính**:
  - dayName: Tên của ngày trong tuần.
  - iconResId: ID của biểu tượng thời tiết tương ứng với ngày.
  - minTemp: Nhiệt độ tối thiểu dự đoán.
  - maxTemp: Nhiệt độ tối đa dự đoán.
- **Phương thức**:
  - **Constructor**: Khởi tạo các thuộc tính với thông tin ngày, ID biểu tượng, nhiệt độ tối thiểu và tối đa.
  - getDayOfWeek(long timestamp): Nhận vào timestamp (thời gian theo giây) và trả về tên ngày trong tuần tương ứng (ví dụ: "Mon", "Tue",...).
  - getIconResourceId(String iconCode): Nhận vào mã biểu tượng thời tiết và trả về ID của biểu tượng tương ứng với mã đó. Nếu không có mã phù hợp, trả về biểu tượng mặc định (sunny).
  - getDayName(): Trả về tên ngày.
  - getIconResId(): Trả về ID biểu tượng thời tiết.
  - getMinTemp(): Trả về nhiệt độ tối thiểu.
  - getMaxTemp(): Trả về nhiệt độ tối đa.
  - getMinMaxTemperature(): Trả về chuỗi kết hợp nhiệt độ tối thiểu và tối đa theo định dạng "min / max".

DailyForecastAdapter: Lớp adapter cho RecyclerView hiển thị dự báo thời tiết hàng ngày.

- **Thuộc tính**:
  - dailyForecastList: Danh sách chứa các đối tượng SimpleForecast đại diện cho dự báo thời tiết hàng ngày.
- **Phương thức**:
  - **Constructor**: Nhận vào danh sách dự báo và khởi tạo dailyForecastList.
  - updateForecastList(List<SimpleForecast> newForecastList): Cập nhật danh sách dự báo bằng cách làm trống danh sách hiện tại và thêm danh sách mới vào, sau đó thông báo cho RecyclerView rằng dữ liệu đã thay đổi.
  - onCreateViewHolder(@NonNull ViewGroup parent, int viewType): Tạo ViewHolder từ layout item_daily_forecast.xml.
  - onBindViewHolder(@NonNull ViewHolder holder, int position): Gán dữ liệu từ dailyForecastList vào các View trong ViewHolder dựa trên vị trí.
    - Gán tên ngày vào dayNameTextView.
    - Gán mô tả nhiệt độ vào temperatureTextView.
    - Gán nhiệt độ tối thiểu vào minTemperatureTextView.
    - Gán nhiệt độ tối đa vào maxTemperatureTextView.
    - **Ghi chú**: Phần gán biểu tượng thời tiết (weatherIconImageView) hiện tại bị bỏ qua và cần được hoàn thiện.
  - getItemCount(): Trả về số lượng dự báo trong dailyForecastList.
- **ViewHolder**:
  - **Các thuộc tính**:
    - dayNameTextView: Hiển thị tên ngày.
    - temperatureTextView: Hiển thị nhiệt độ hiện tại hoặc mô tả thời tiết.
    - minTemperatureTextView: Hiển thị nhiệt độ tối thiểu.
    - maxTemperatureTextView: Hiển thị nhiệt độ tối đa.
    - weatherIconImageView: Hiển thị biểu tượng thời tiết.

FirebaseDatabaseHelper: Lớp giúp tương tác với Firebase Realtime Database để lưu trữ thông tin thời tiết.

- **Thuộc tính**:
  - mDatabase: Tham chiếu đến phiên bản của Firebase Database.
  - mReferenceStudents: Tham chiếu đến nút "Weather" trong cơ sở dữ liệu Firebase, nơi lưu trữ thông tin thời tiết.
- **Phương thức**:
  - **Constructor**: Khởi tạo mDatabase và mReferenceStudents, cho phép truy cập vào cơ sở dữ liệu Firebase.
  - addWeather(WeatherCity weatherCity):
    - Nhận một đối tượng WeatherCity và thêm thông tin thời tiết vào Firebase.
    - **Chi tiết**:
      - Tạo một HashMap để lưu thông tin thời tiết từ weatherCity.

        - Thêm các thuộc tính như CurrentTemperature, WeatherDescription, MinTemperature, MaxTemperature, RainFall, Humidity, và WindSpeed vào HashMap.
        - Lưu danh sách dự báo theo giờ (HourlyForecast) bằng cách chuyển đổi nó thành định dạng JSON với Gson và thêm vào HashMap.
        - **Ghi chú**: Phần mã để lưu danh sách dự báo hàng ngày (listDailyForecast) hiện tại không có và cần được hoàn thiện.
        - Sử dụng studentId (tên thành phố) làm khóa cho nút dữ liệu trong Firebase. Nếu studentId không null, lưu weatherCityHashMap vào cơ sở dữ liệu với khóa là tên thành phố.

HourlyForecast: Lớp đại diện cho dự báo thời tiết theo giờ với thông tin chi tiết về nhiệt độ, mô tả thời tiết và thời gian.

- **Thuộc tính**:
    - temperature: Nhiệt độ trong khoảng thời gian cụ thể, kiểu dữ liệu String.
    - description: Mô tả về thời tiết tại thời điểm đó, kiểu dữ liệu String.
    - time: Thời gian tương ứng với dự báo thời tiết, được chuyển đổi từ Unix timestamp sang định dạng thời gian dễ đọc.
- **Constructor**:
    - Nhận ba tham số: temperature, description, và timestamp (được truyền dưới dạng kiểu long).
    - Gọi phương thức convertTimestampToTime(timestamp) để chuyển đổi Unix timestamp thành định dạng thời gian đọc được.
- **Phương thức**:
    - convertTimestampToTime(long timestamp):
        - Chuyển đổi Unix timestamp (giây) thành một chuỗi thời gian.
        - Sử dụng SimpleDateFormat để định dạng thời gian thành "ha" (ví dụ: "3:00 PM").
        - Cài đặt múi giờ mặc định để đảm bảo thời gian hiển thị đúng theo vị trí người dùng.
    - getTemperature(): Trả về giá trị của thuộc tính temperature.
    - getDescription(): Trả về giá trị của thuộc tính description.
    - getTime(): Trả về giá trị của thuộc tính time.

HourlyForecastAdapter: Lớp này kế thừa từ RecyclerView.Adapter để hiển thị danh sách dự báo thời tiết theo giờ trong một RecyclerView.

Cấu trúc của lớp:

- **Thuộc tính**:
    - hourlyForecastList: Danh sách các đối tượng HourlyForecast đại diện cho dự báo thời tiết theo giờ.
- **Constructor**:
    - Nhận một danh sách hourlyForecastList và gán giá trị cho thuộc tính này.
- **Phương thức**:

- updateForecastList(List<HourlyForecast> newForecastList): Cập nhật danh sách dự báo bằng cách xóa tất cả các phần tử hiện tại và thêm các phần tử mới từ newForecastList. Sau đó, thông báo cho RecyclerView rằng dữ liệu đã thay đổi.
- onCreateViewHolder(ViewGroup parent, int viewType):
  - Inflates layout cho từng item trong RecyclerView bằng cách sử dụng LayoutInflater.
  - Trả về một đối tượng HourlyForecastViewHolder.
- onBindViewHolder(@NonNull HourlyForecastViewHolder holder, int position):
  - Nhận một ViewHolder và vị trí của item.
  - Thiết lập các giá trị cho các TextView trong ViewHolder bằng cách lấy dữ liệu từ đối tượng HourlyForecast tương ứng trong hourlyForecastList.
  - Hiển thị nhiệt độ, mô tả thời tiết và thời gian.
- getItemCount():
  - Trả về số lượng item trong hourlyForecastList, cho phép RecyclerView biết số lượng mục để hiển thị.

Lớp HourlyForecastViewHolder:

- **Thuộc tính**:
  - tempTextView: TextView hiển thị nhiệt độ.
  - descTextView: TextView hiển thị mô tả thời tiết.
  - timeTextView: TextView hiển thị thời gian.
- **Constructor**:
  - Nhận một View itemView và khởi tạo các TextView bằng cách tìm ID từ layout.

LocationHistory: Lớp đại diện cho lịch sử vị trí, lưu trữ thông tin về tọa độ địa lý và thành phố theo thời gian.

Thuộc Tính:

- latitude: Độ vĩ (latitude) của vị trí, kiểu dữ liệu double.
- longitude: Độ kinh (longitude) của vị trí, kiểu dữ liệu double.
- city: Tên thành phố tương ứng với vị trí, kiểu dữ liệu String.
- timestamp: Thời gian ghi lại vị trí, kiểu dữ liệu String.

Phương Thức:

- **Constructor Mặc Định**: Cần thiết cho Firebase để khởi tạo một đối tượng mà không cần tham số.
- **Constructor Có Tham Số**:
  - Nhận vào các tham số: latitude, longitude, city, và timestamp.
  - Khởi tạo thuộc tính tương ứng với các giá trị được truyền vào.

- **Getters và Setters**:
  - getLatitude(): Trả về giá trị latitude.
  - setLatitude(double latitude): Cập nhật giá trị latitude.
  - getLongitude(): Trả về giá trị longitude.
  - setLongitude(double longitude): Cập nhật giá trị longitude.
  - getCity(): Trả về tên thành phố (city).
  - setCity(String city): Cập nhật tên thành phố.
  - getTimestamp(): Trả về giá trị timestamp.
  - setTimestamp(String timestamp): Cập nhật giá trị timestamp.

MainActivity: Lớp chính của ứng dụng thời tiết, hiển thị thông tin thời tiết hiện tại và dự báo thời tiết theo giờ.

Thuộc Tính:

- **UI Components**:
  - LinearLayout layoutBackground: Khu vực nền cho hoạt động.
  - HourlyForecastAdapter hourlyForecastAdapter: Adapter cho RecyclerView hiển thị dự báo thời tiết theo giờ.
  - TextView và ImageView: Các thành phần UI hiển thị thông tin thời tiết, như tên thành phố, nhiệt độ hiện tại, mô tả thời tiết, tốc độ gió, độ ẩm, v.v.
- **Constants**:
  - REQUEST_LOCATION_PERMISSION và REQUEST_CHECK_SETTINGS: Các mã yêu cầu để kiểm tra và yêu cầu quyền vị trí.

Phương Thức:

- **onCreate(Bundle savedInstanceState)**:
  - Khởi tạo các thành phần UI và thiết lập RecyclerView cho dự báo thời tiết theo giờ.
  - Nhận tên thành phố từ Intent, nếu không có thì mặc định gọi API cho Tokyo.
  - Thiết lập sự kiện click cho biểu tượng bản đồ.
- **getDayOfWeek(long timestamp)**:
  - Nhận vào một timestamp và trả về tên ngày trong tuần.
- **getIconResourceId(String iconCode)**:
  - Nhận vào mã biểu tượng thời tiết và trả về ID tài nguyên tương ứng.
- **CallApi(String city)**:
  - Gọi API thời tiết với tên thành phố đã cung cấp và cập nhật UI sau khi nhận dữ liệu.
- **updateBackground(String description)**:
  - Cập nhật nền của hoạt động dựa trên mô tả thời tiết.
- **updateUI(WeatherCity weatherCity)**:

    - Cập nhật các thành phần UI với dữ liệu thời tiết nhận được từ API, bao gồm cập nhật nền, tên thành phố, nhiệt độ, mô tả thời tiết, và dữ liệu dự báo theo giờ.
- **OnClickCityActivity(View view)**:
    - Xử lý sự kiện click để chuyển sang hoạt động CityForecastActivity, cho phép người dùng xem dự báo thời tiết cho các thành phố khác.

MapActivity: Lớp này quản lý việc hiển thị bản đồ và lớp phủ thời tiết từ OpenWeather trên ứng dụng thời tiết.

Thuộc Tính:

- **GoogleMap mMap**: Đối tượng bản đồ để thao tác với các tính năng bản đồ.
- **String API_KEY**: Khóa API cho OpenWeather, được sử dụng để truy cập dữ liệu thời tiết.

Phương Thức:

- **onCreate(Bundle savedInstanceState)**:
    - Khởi tạo giao diện của hoạt động và thiết lập Toolbar với nút quay lại.
    - Khởi tạo SupportMapFragment và yêu cầu bản đồ sẵn sàng thông qua callback onMapReady.
- **onMapReady(@NonNull GoogleMap googleMap)**:
    - Nhận đối tượng GoogleMap khi bản đồ đã sẵn sàng.
    - Đặt camera vào một vị trí cụ thể (ví dụ: TP Hồ Chí Minh).
    - Gọi phương thức để thêm lớp phủ thời tiết vào bản đồ.
- **addWeatherOverlay(String layer)**:
    - Tạo lớp phủ thời tiết từ OpenWeather bằng cách sử dụng UrlTileProvider.
    - Tạo URL để lấy lớp phủ theo thông số như zoom, x, y, và khóa API.
    - Thêm lớp phủ vào bản đồ bằng cách sử dụng addTileOverlay.

SharedPreferencesControl: Lớp này quản lý việc lưu trữ và truy xuất thông tin dự báo thời tiết thành phố trong ứng dụng bằng cách sử dụng SharedPreferences.

Thuộc Tính:

- **String PREF_NAME**: Tên của tệp SharedPreferences dùng để lưu trữ dữ liệu.
- **String KEY_CITIES**: Khóa để lưu danh sách các thành phố.
- **String KEY_FIRST_LAUNCH**: Khóa để kiểm tra lần đầu tiên khởi chạy ứng dụng.
- **SharedPreferences sharedPreferences**: Đối tượng SharedPreferences để thao tác với dữ liệu.
- **SharedPreferences.Editor editor**: Đối tượng dùng để chỉnh sửa dữ liệu trong SharedPreferences.

Phương Thức:

- **SharedPreferencesControl(Context context)**:
  - Khởi tạo đối tượng SharedPreferences và Editor với tên tệp xác định.
- **void saveCities(List<CityForecast> cityForecast)**:
  - Chuyển danh sách các đối tượng CityForecast thành chuỗi JSON và lưu trữ trong SharedPreferences dưới dạng StringSet.
- **List<CityForecast> getCities()**:
  - Kiểm tra xem đây có phải là lần đầu tiên khởi chạy ứng dụng không bằng cách đọc giá trị từ KEY_FIRST_LAUNCH.
  - Nếu đây là lần đầu, thêm dữ liệu mẫu vào HashSet và cập nhật giá trị của KEY_FIRST_LAUNCH.
  - Chuyển đổi các chuỗi JSON từ HashSet về danh sách các đối tượng CityForecast và trả về danh sách này.

SimpleForecast: Lớp này đại diện cho dự báo thời tiết đơn giản cho một ngày, bao gồm thông tin như ngày, mô tả thời tiết, nhiệt độ tối thiểu, tối đa và nhiệt độ hiện tại.

Thuộc Tính:

- **String day**: Ngày mà dự báo thời tiết áp dụng.
- **String description**: Mô tả ngắn gọn về thời tiết trong ngày (ví dụ: "Nắng", "Mưa").
- **float minTemp**: Nhiệt độ tối thiểu dự kiến trong ngày.
- **float maxTemp**: Nhiệt độ tối đa dự kiến trong ngày.
- **float currenttemp**: Nhiệt độ hiện tại.

Phương Thức:

- **SimpleForecast(String day, String description, float minTemp, float maxTemp, float currenttemp)**:
  - Hàm khởi tạo, nhận các tham số để thiết lập giá trị cho các thuộc tính của đối tượng.
- **String getDay()**:
  - Phương thức truy xuất giá trị của thuộc tính day.
- **String getDescription()**:
  - Phương thức truy xuất giá trị của thuộc tính description.
- **float getMinTemp()**:
  - Phương thức truy xuất giá trị của thuộc tính minTemp.
- **float getMaxTemp()**:
  - Phương thức truy xuất giá trị của thuộc tính maxTemp.
- **private float GetCurrentTemp()**:
  - Phương thức truy xuất giá trị của thuộc tính currenttemp, nhưng bị đặt ở chế độ private, nghĩa là không thể truy cập từ bên ngoài lớp.

WeatherAPI: Lớp này quản lý việc lấy dữ liệu thời tiết từ OpenWeather API bằng cách sử dụng thư viện Volley để thực hiện các yêu cầu mạng. Nó bao gồm các phương thức để lấy thông tin thời tiết cho một thành phố cụ thể.

Thuộc Tính:

- **static final String API_KEY**: Khóa API dùng để xác thực yêu cầu đến OpenWeather.
- **RequestQueue requestQueue**: Hàng đợi yêu cầu cho Volley, dùng để gửi và xử lý các yêu cầu mạng.

Phương Thức:

- **WeatherAPI(Context context)**:
  - Hàm khởi tạo, nhận một Context để khởi tạo hàng đợi yêu cầu.
- **void fetchWeatherData(String city, WeatherDataCallback callback)**:
  - Phương thức này nhận tên thành phố và một callback để xử lý kết quả.
  - Tạo URL yêu cầu đến OpenWeather API với tên thành phố và gửi yêu cầu mạng bằng JsonObjectRequest.
  - Khi nhận được phản hồi, nó sẽ gọi parseWeatherData để phân tích dữ liệu và trả kết quả qua callback.
- **private WeatherCity parseWeatherData(JSONObject response, String city) throws JSONException**:
  - Phân tích dữ liệu JSON nhận được từ API.
  - Lấy thông tin thời tiết hiện tại như nhiệt độ, độ ẩm, tốc độ gió và mô tả thời tiết.
  - Tạo một đối tượng WeatherCity với thông tin đã phân tích và trả về đối tượng này.
  - Cũng lấy dữ liệu dự báo theo giờ cho 12 khoảng thời gian và thêm vào danh sách các dự báo.

Giao Diện:

- **WeatherDataCallback**:
  - Giao diện này định nghĩa hai phương thức:
    - **void onSuccess(WeatherCity weatherCity)**: Gọi khi dữ liệu thời tiết được lấy thành công.
    - **void onFailure(String errorMessage)**: Gọi khi có lỗi xảy ra trong quá trình lấy dữ liệu.

WeatherInfo: Lớp này chứa các thông tin chi tiết về thời tiết hiện tại và các dự báo thời tiết cho một thành phố.

Thuộc Tính:

- **double currentTemperature**: Nhiệt độ hiện tại của thành phố.
- **String weatherDescription**: Mô tả tình trạng thời tiết (ví dụ: "Clear sky", "Rain", etc.).
- **double maxTemperature**: Nhiệt độ cao nhất dự kiến trong một khoảng thời gian nhất định.

- **double minTemperature**: Nhiệt độ thấp nhất dự kiến trong một khoảng thời gian nhất định.
- **double rainfall**: Lượng mưa dự kiến trong khoảng thời gian nhất định (có thể được dùng để theo dõi lượng mưa trong ngày hoặc theo giờ).
- **int humidity**: Độ ẩm hiện tại, thường được biểu diễn dưới dạng phần trăm.
- **double windSpeed**: Tốc độ gió hiện tại (có thể được đo bằng km/h hoặc m/s).
- **List<HourlyForecast> hourlyForecasts**: Danh sách các đối tượng HourlyForecast để lưu trữ thông tin dự báo thời tiết theo giờ.
- **List<SimpleForecast> simpleForecasts**: Danh sách các đối tượng SimpleForecast để lưu trữ thông tin dự báo thời tiết theo ngày.

WeatherCity: Lớp này đại diện cho thông tin thời tiết cho một thành phố cụ thể, bao gồm các thông tin chi tiết như nhiệt độ, độ ẩm, tốc độ gió và các dự báo thời tiết.

Thuộc Tính:

- **String city**: Tên thành phố.
- **WeatherInfo weatherInfo**: Đối tượng chứa thông tin thời tiết chi tiết như nhiệt độ, độ ẩm, tốc độ gió, và các dự báo theo giờ và theo ngày.

Phương Thức:

- **WeatherCity(String city)**:
  - Hàm khởi tạo nhận tên thành phố và khởi tạo đối tượng WeatherInfo.
- **getCurrentTemperature()**:Trả về nhiệt độ hiện tại.
- **setCurrentTemperature(double currentTemperature)**:Thiết lập nhiệt độ hiện tại.
- **getWeatherDescription()**:Trả về mô tả thời tiết.
- **setWeatherDescription(String weatherDescription)**:Thiết lập mô tả thời tiết.
- **getMaxTemperature()**:Trả về nhiệt độ tối đa.
- **setMaxTemperature(double maxTemperature)**:Thiết lập nhiệt độ tối đa.
- **getMinTemperature()**:Trả về nhiệt độ tối thiểu.
- **setMinTemperature(double minTemperature)**:Thiết lập nhiệt độ tối thiểu.
- **getRainfall()**:Trả về lượng mưa.
- **setRainfall(double rainfall)**:Thiết lập lượng mưa.
- **getHumidity()**:Trả về độ ẩm.
- **void setHumidity(int humidity)**:Thiết lập độ ẩm.
- **getWindSpeed()**:Trả về tốc độ gió.
- **setWindSpeed(double windSpeed)**:Thiết lập tốc độ gió.
- **getHourlyForecasts()**:Trả về danh sách các dự báo thời tiết theo giờ.
- **hourlyForecasts**:Thiết lập danh sách các dự báo thời tiết theo giờ.
- **getDailyForecasts()**:Trả về danh sách các dự báo thời tiết theo ngày.
- **setDailyForecasts)**:Thiết lập danh sách các dự báo thời tiết theo ngày.
- **setCity(String city)**:Thiết lập tên thành phố.
- **setWeatherInfo(WeatherInfo weatherInfo)**:Thiết lập đối tượng WeatherInfo.
- **getCity()**:Trả về tên thành phố.

CHƯƠNG 3. KẾT QUẢ THỰC HIỆN

3.1. Công nghệ đã sử dụng

- Ngôn ngữ lập trình: Java
- Công cụ: IntelliJ IDEA
- Thư viện: Gson, Volley, Play Service Location.

3.2. Tiến độ thực hiện

Link github tới dự án: https://github.com/DevVNpro/CSE441_PROJECT

Link Figma tới dự án: https://www.figma.com/design/SFy53duer4BGd3BZPFd17E/design_android_63KTPM2?node-id=0-1&node-type=canvas&t=uqhlSqdpVIOa0o5O-0

KẾT LUẬN