## Running and Testing Locally

1. Replace OPEN\_WEATHER\_API\_KEY\_PLACEHOLDER string in the application.yaml with your Open Weather API Key or load it as environment variable under the name OPEN\_WEATHER\_API\_KEY.
2. Use the following command to start the application.  
   ./mvnw spring-boot:run
3. Use following CURL to get the weather updates. Replace {REPLACE\_WITH\_API\_KEY} string by one of the following 5 API Keys which are loaded to database during the application startup to replace.  
     
   5cf9dd49-c5f2-4634-893a-23a31abf2bdc  
   76002345-0196-47d1-8ee7-743b4c7ce86f  
   7e967fbb-4fb9-4ed5-847b-de0c032bb23f  
   570ace6f-b516-459c-9a24-25b8b2f8e576  
   744443ad-9fcd-40b8-ae49-01c6838d5041

curl --location 'localhost:8080/v1/weather?city=melbourne&countryCode=au' \  
--header 'X-API-KEY: {REPLACE\_WITH\_API\_KEY}'

## Design Decisions

A screenshot of a computer

Description automatically generated

* api\_key table is used to track the existing API keys and their active status.
* access\_log table is used to track the usage of the above API keys.
* weather\_update table is used to store the weather updates obtain from the OpenWeatherMap API. A unique key index of columns city, country\_code and period\_code is used to determine whether requested weather update for a given location exists during a certain hour of the day. If it exists, update will be returned from database otherwise OpenWeatherMap API will be invoked.
* period\_code value will be saved in the format – ‘yyyyMMddHH’. (E.g., 2024080613)
* Spring AOP has been used to validate and track the API key usage and API key protected endpoints will be annotated with @RateControlled.

## Future Enhancements

* Store all the database credentials and external credentials stored in configuration file in AWS SecretManager and load them as environment variables.
* Incorporate caching to reduce database reads on frequently fetched locations.