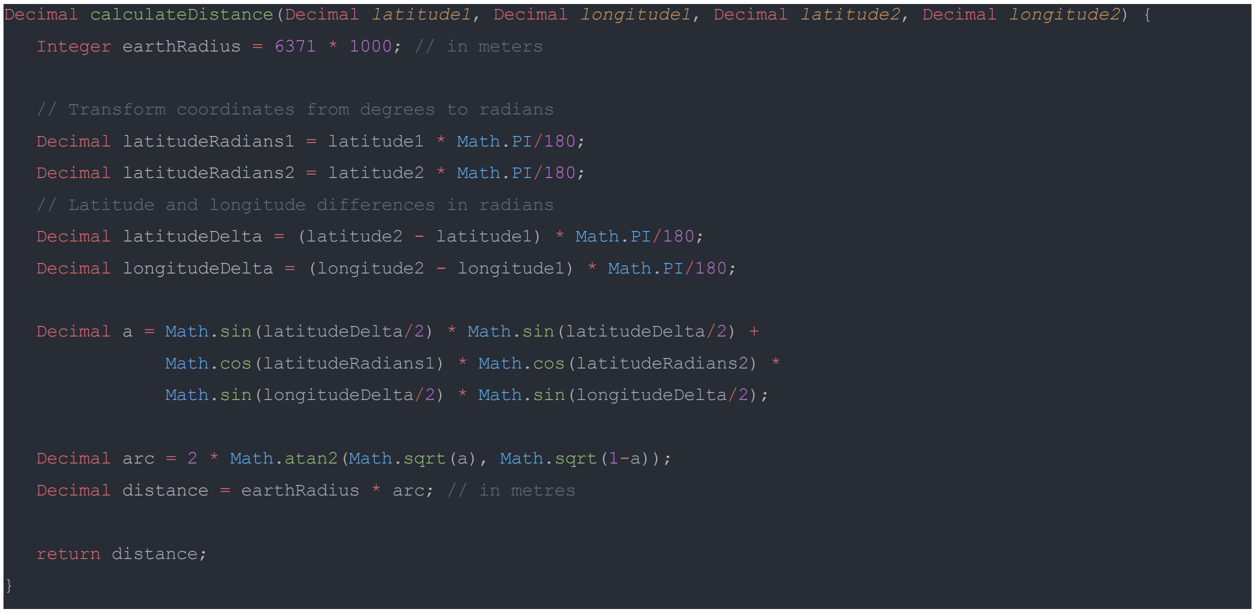
Candidate exercise

Instructions

- The aim of the exercise is to demonstrate your skills understanding and solving the stated problem, demonstrating **coding quality, structure**, **error handling**, **code readability** and use of **best practices**.
- This is **not** designed to be a math problem.
- This is **not** designed to be a frontend exercise, but a simple frontend interface is **required** just to input the necessary data and display the information needed.
- Once the problem is understood it takes an average of *180* minutes to code it
- Email your code as a ZIP file to jobs@nuvolar.eu. Add any information needed for the deployment, please. Alternatively, you can upload your code to a private github repository and send us the download link.

Requirements

- Create a custom LWC/visualforce page in **Salesforce** that allows the user to create a Flight between two airports, calculate the flight distance and save the flight in the database. The application must fulfill the following requirements:
  - Provide a frontend form to fill in the details:
    - Arrival airport: will be retrieved from the database filtering by **IATA code**.
    - Departure airport: will be retrieved from the database filtering by **IATA code**
  - Save flight:
    - The flight is saved to the database, storing the following values:
      - Departure airport
      - Arrival airport
      - Flight distance in **kilometers**
    - After saving the flight, display in the frontend the resulting flight information.
- Other useful information
  - The Airports are identified by a 3-letter code named **IATA** (i.e. Barcelona Airport code is BCN).
  - The Airport must also store longitude and latitude.
    - Latitude: the valid range for latitude values is from +90 to -90 **degrees**.
    - Longitude: the valid range for longitude values is from +180 to -180 **degrees**.
  - The flight distance can be computed using the Haversine formula described in the provided Apex method.



```
Decimal calculateDistance(Decimal latitude1, Decimal longitude1, Decimal latitude2, Decimal longitude2) {
   Integer earthRadius = 6371 * 1000; // in meters

   // Transform coordinates from degrees to radians
   Decimal latitudeRadians1 = latitude1 * Math.PI/180;
   Decimal latitudeRadians2 = latitude2 * Math.PI/180;
   // Latitude and longitude differences in radians
   Decimal latitudeDelta = (latitude2 - latitude1) * Math.PI/180;
   Decimal longitudeDelta = (longitude2 - longitude1) * Math.PI/180;

   Decimal a = Math.sin(latitudeDelta/2) * Math.sin(latitudeDelta/2) +
               Math.cos(latitudeRadians1) * Math.cos(latitudeRadians2) *
               Math.sin(longitudeDelta/2) * Math.sin(longitudeDelta/2);

   Decimal arc = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
   Decimal distance = earthRadius * arc; // in metres

   return distance;
}
```