Software Architecture Document

Flight Information Query System

SOEN 6441 (W) - Project

Team Members:
Peter Sakr - 40237311
Abhishek Amola - 40105405

Table of Contents

1. Purpose of this Document

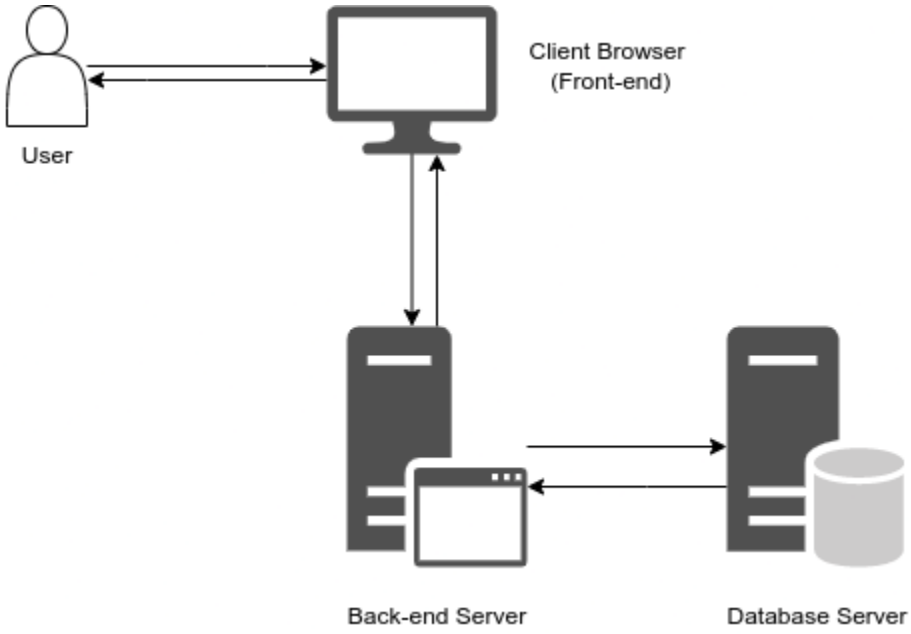
This document will provide a detailed architecture design of the Flight Information Query system that is built as part of the SOEN 6441 Course project at Concordia University.

More specifically, this document will explain the project based on its user requirements. Based on these requirements, the document will illustrate our architectural overview diagram to showcase our overall system's design and how the user is expected to interact with it. Then, the data model will be defined and explained through the domain class diagrams and an ER diagram. Based on the domain classes, a class diagram will be presented to pinpoint the different classes of the system and their associations. Finally a sequence diagram will be shown to give an example of a sample interaction with the system.

2. Overall Architecture

2.1 Architecture Diagram

The following diagram depicts the interactions between the components in our proposed system:



The user shall interact with the system using their browser, which will communicate with the back-end server. The back-end will perform all required database queries and send them to the client to present to the user.

2.2 Patterns To Be Implemented

Client-Server Pattern:

As shown in the overall architectural diagram, the system will be developed using a Client-Server architecture. The client will send all user requests to the backend (the server), which will process and respond with all necessary data, in a simple format capable of presentation to the user.

Table Data Gateway:

The Table Data Gateway allows decoupling domain class code from database query code. This occurs by having all database code (in this case, SQL code) in the Table Data Gateway class. This class will have information passed into its functions from the rest of the system, which will be translated into SQL queries and executed.

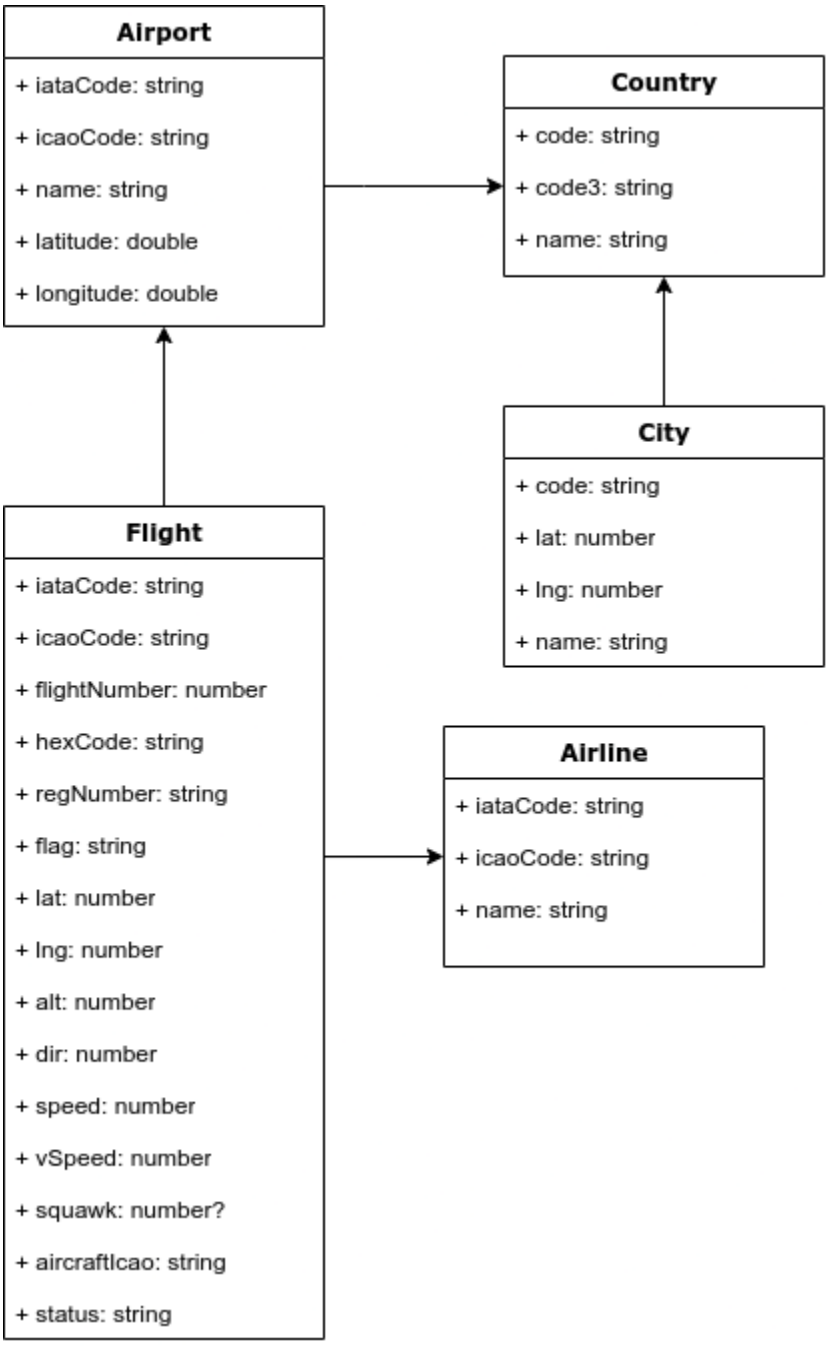
Data Mapper:

The Data Mapper works hand-in-hand with the table data gateway. This class performs all interactions with the table data gateway, parsing any database responses and creating domain classes from the responses. This allows removing any required mapping from domain classes, allowing us to re-use the classes even if the database mapping changes.

3. Data Model

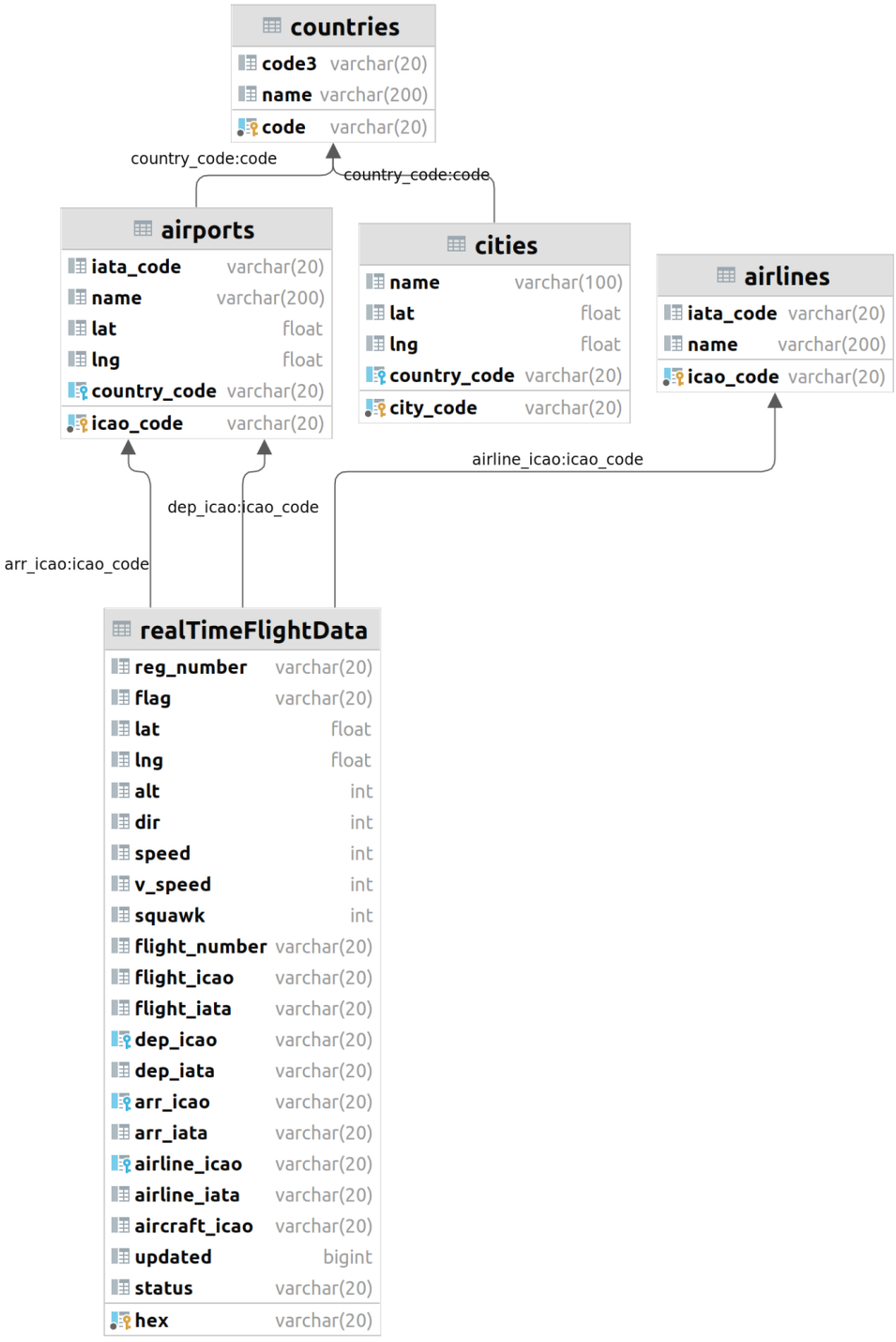
3.1 Domain Class Model

After analyzing our system's requirements, we designed the following domain class model:



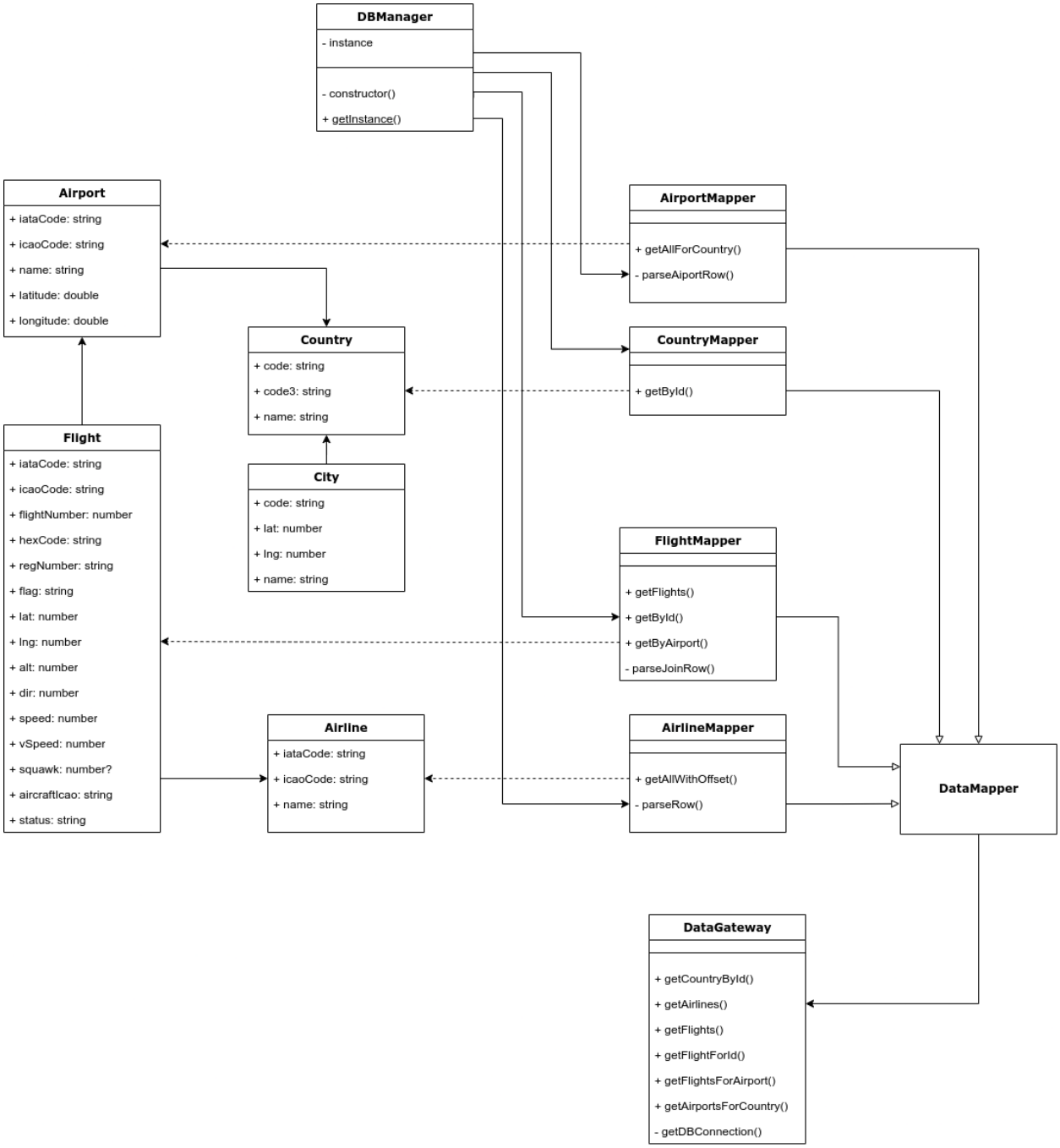
3.2 ER-Model

Based on the above diagram, we developed the domain classes into the following ER-Diagram, to model how the data will be stored in the database.



4. Backend Architecture

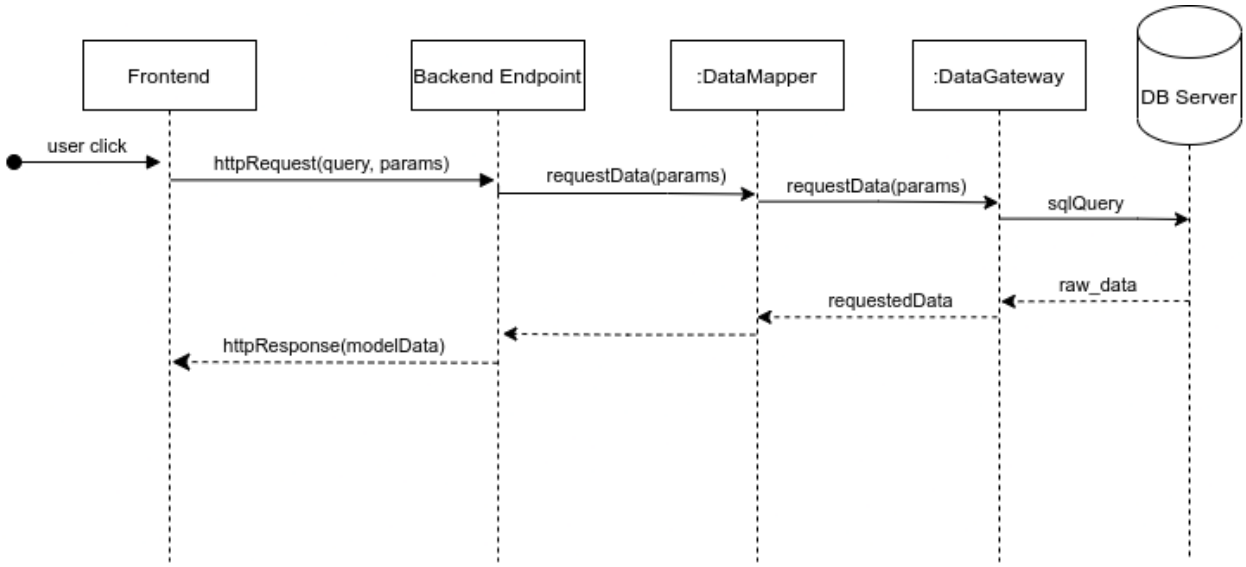
4.1 Class Diagram



After constructing the domain model, we designed the above class diagram for the backend system. The DataGateway class acts as the **Table Data Gateway**, and is the only class that contains any SQL query code. The four classes that inherit the DataMapper class act as the

Data Mappers, and use the results from the DataGateway to map the results to the corresponding domain classes.
The DBManager class, which uses the **Singleton** pattern, keeps and manages instances of the mappers for use in the endpoints.

4.2 Interaction Sequence Diagram



The above sequence diagram illustrates how the backend will execute a received request from the frontend.