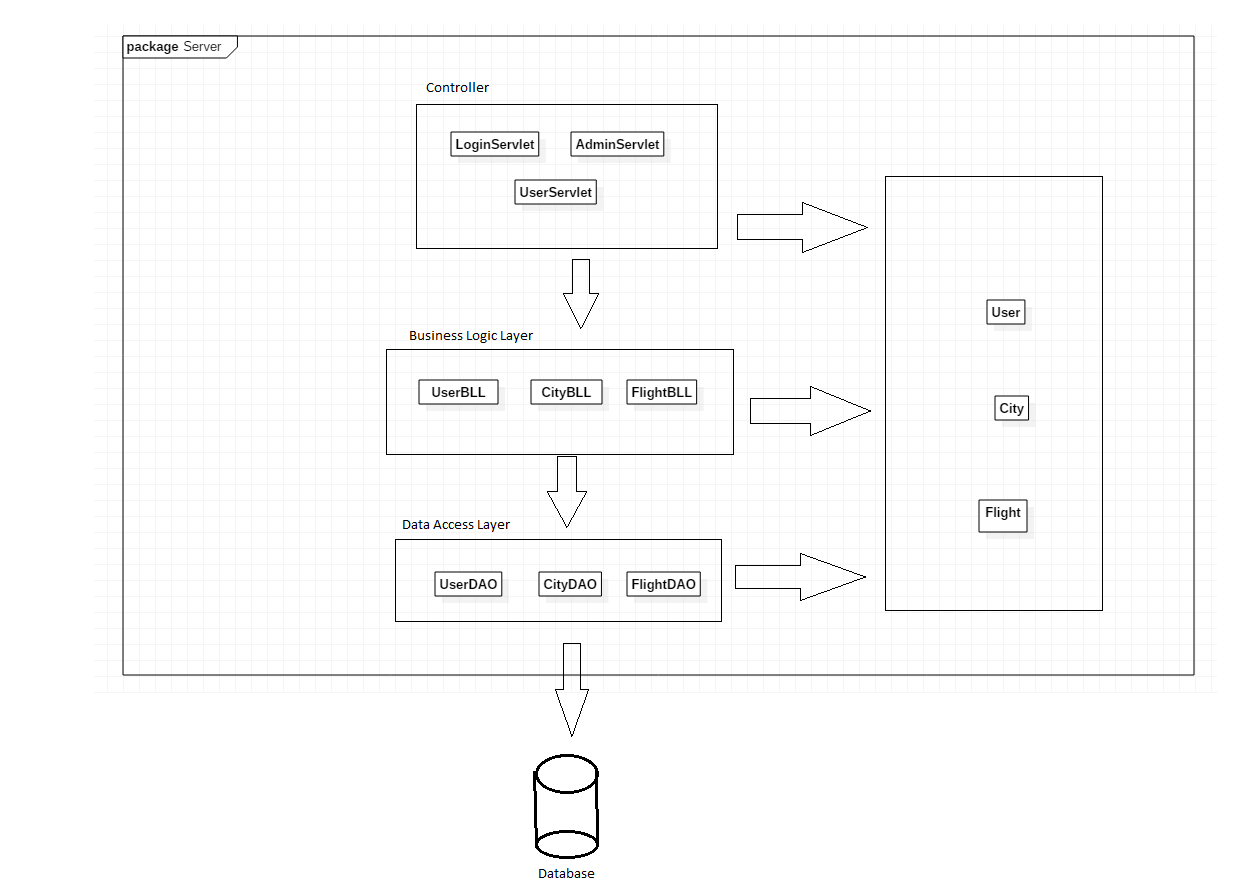
DISTRIBUTED SYSTEMS

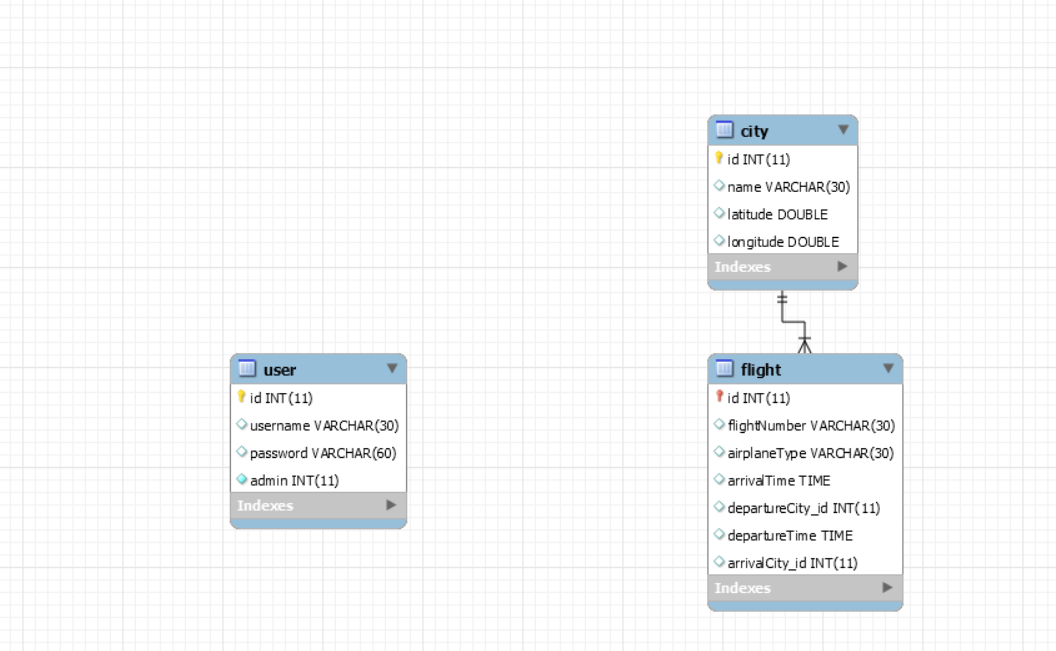
Assignment 1

Request-Reply Communication Paradigm

A1.2: Web app using Request-Reply

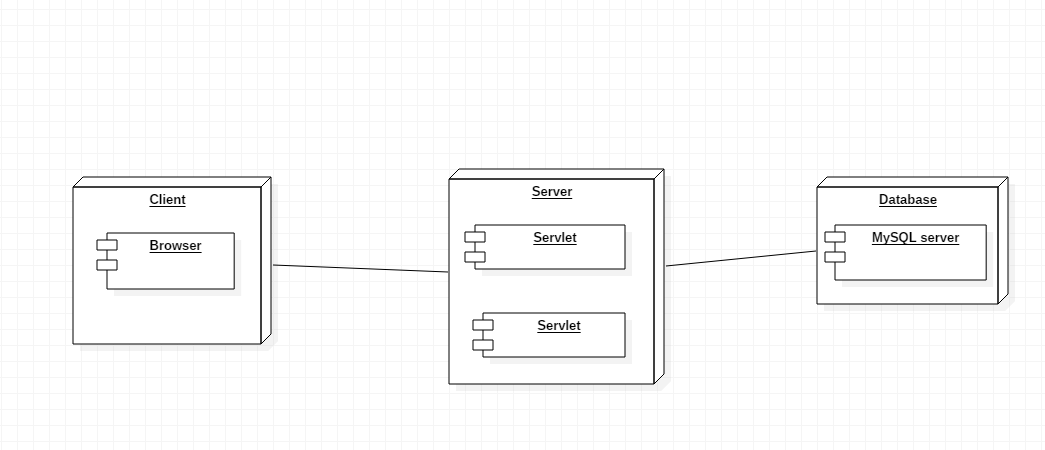
* Conceptual architecture of the distributed system:

The distributed system is organized on three logical levels: Client, Server and Database. The client is represented by the internet browser and is responsible for displaying the interface and registering the user’s actions. This is done through the use of HTTP requests (mainly GET and POST). The requests are sent to the server side where they are intercepted by certain Servlet classes, depending on the URL of the request. Each servlet is mapped to a different URL and is able to intercept and handle GET and POST requests, sending back responses to the web browser in order to be displayed. In order to achieve their functionality, servlets call a lower layer of the server application (business logic layer), which handles validation and business-related logic by calling, in turn, the lower application layer (data access layer). The data access layer interacts directly with the database through the Hibernate ORM framework. The database tables are mapped to the model classes such that each database table entry corresponds to an instance of a model class. Model classes are used to communicate between layers and with the database. There is also a number of other classes that are used to support the

* Database design:

The database design is fairly simple as the application does not require a large number of tables with complex relationships among them. The user table holds the registered users of the web application (and whether they are admins or not). It is only accessed at login when a user’s credentials are verified against the database contents. The password is stored encrypted using the BCrypt algorithm. The city table stores data about the possible destinations of the flights, namely the latitude and longitude of the cities the company flies to. This data is used to compute the local time at arrival for flights. The flight table holds data regarding the company’s flights, including the departure city and destination city, which are foreign keys from the city table. There is a one-to-many relationship between the city and the flight tables.

* UML Deployment diagram:

The deployment diagram is pretty simple as there are not many components in our project. In the current setup, all the components are expected to be on the same machine, however, with the minimal appropriate changes, the setup presented in the diagram can be achieved.

* Build and execution considerations: This project was built using IntelliJ IDEA Ultimate 2017.3 and is run using the Apache Tomcat 8.5.34 server.
  + Opening the project:
    1. Open IntelliJ IDEA
    2. File | Open
    3. Browse and select the directory containing the project
    4. Click “Open”
    5. Go to pom.xml file, double click to open it and right click on the editor
    6. Select Maven | Reimport to make sure that all the required libraries are included
    7. Open the hibernate.cfg.xml file from src/main/resources
    8. Modify the appropriate fields with the data required to access the database schema
  + Running the project:
    1. Go to Run | Edit Configurations
    2. Click on the Green plus (+)
    3. From the dropdown list, select Tomcat Server | Local
    4. (Optional) Assign a name to the run configuration
    5. Click on the “Fix” button from bottom right
    6. Select (Project name):war
    7. Click “Ok”
    8. Click Run | Run (Configuration name) – first option
    9. A browser page should automatically open with the default URL (<http://localhost:8080/> is default). If not, open a browser and go to the aforementioned URL.