Assignment 1.2

Flights Management System

## Student: IEREMIAȘ Viorel

## Group: 30442

# **Requirements analysis**

## **Assignment specification**

The application is used in the management of flights from multiple locations across the world in a unique platform that offers a complete set of functionalities. Each flight consists of the following information: flight number, airplane type, departure city, departure date and hour, arrival city, arrival date and hour. Each city has associated its geographical coordinates: latitude and longitude. The application has two types of users: clients and administrators.

Both types of users can list flights, using filters and ranges on the important characteristics of the flight, like location and time, either of departure, or of arrival, airplane type, flight duration, etc.

Additionally, for the regular clients, it is relevant to find the local date and time of both the departure, so clients can query the system for this information, in the context in which normally dates and times are represented according to UTC for the purpose of consistency. In order to display the local time, the geographical coordinates of the city are passed to an external web service which returns the actual time value.

The administrators have extended functionality. Besides the functions already mentioned, they can also add new flight in the system, or perform modifications, including deletion of the current flights.

Both clients and administrations have to provide a username and a password in order to use the application. Although they use the same platform, the two components are disjunct, so a client cannot access the area dedicated to administrators and vice-versa.

## **Functional requirements**

All users can perform the following operations:

• authenticate into the platform

• list all flights

• search and filter flights

Clients are also able to:

• query for the local time of the departure and the arrival

Administrators also have the ability to:

• add a new flight

• modify/update existing flight

• delete a flight

# **Design and architecture**

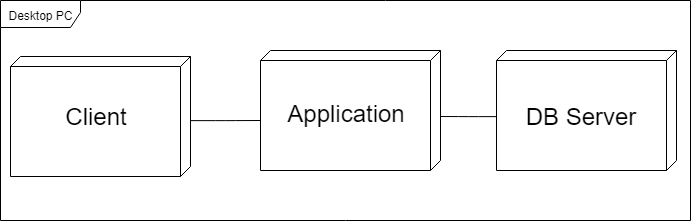
## **Conceptual architecture**

The solution is a three-tiered distributed system, consisting of the following tiers: Data Access, Business Layer and Presentation.

Data access tier manages the access and modification of persistent data that is retrieved, processed and written back by the business processes. A database server is responsible for the data management, while a connector manages the connection and the communication with the next tier.

The business layer is a program that accomplishes the processing involved by the business flows initiated by user requests. The program is layered, on isolation levels, which abstract out different tasks, decouple concerns and increase coherency inside a layer. The layers communicate in a single direction, from the interaction with the user towards the data access layer, in terms of contracts set out by interfaces.

The presentation tier is represented by the client that makes requests to the server, to obtain the information it needs. In this case, the user interacts with the system through a browser which renders the responses that come from the server as result of requests being submitted.

**The application server layer is an important component that connects the processing and presentation tiers. It is represented by an instance of Apache Tomcat, an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies which acts as a web container for the application. A web container is responsible for managing the lifecycle of servlets (creation of servlet instances, loading and unloading servlets), creating and managing request and response objects and performing other servlet-management tasks, like mapping a URL to a particular servlet and ensuring that the URL requester has the correct access-rights.

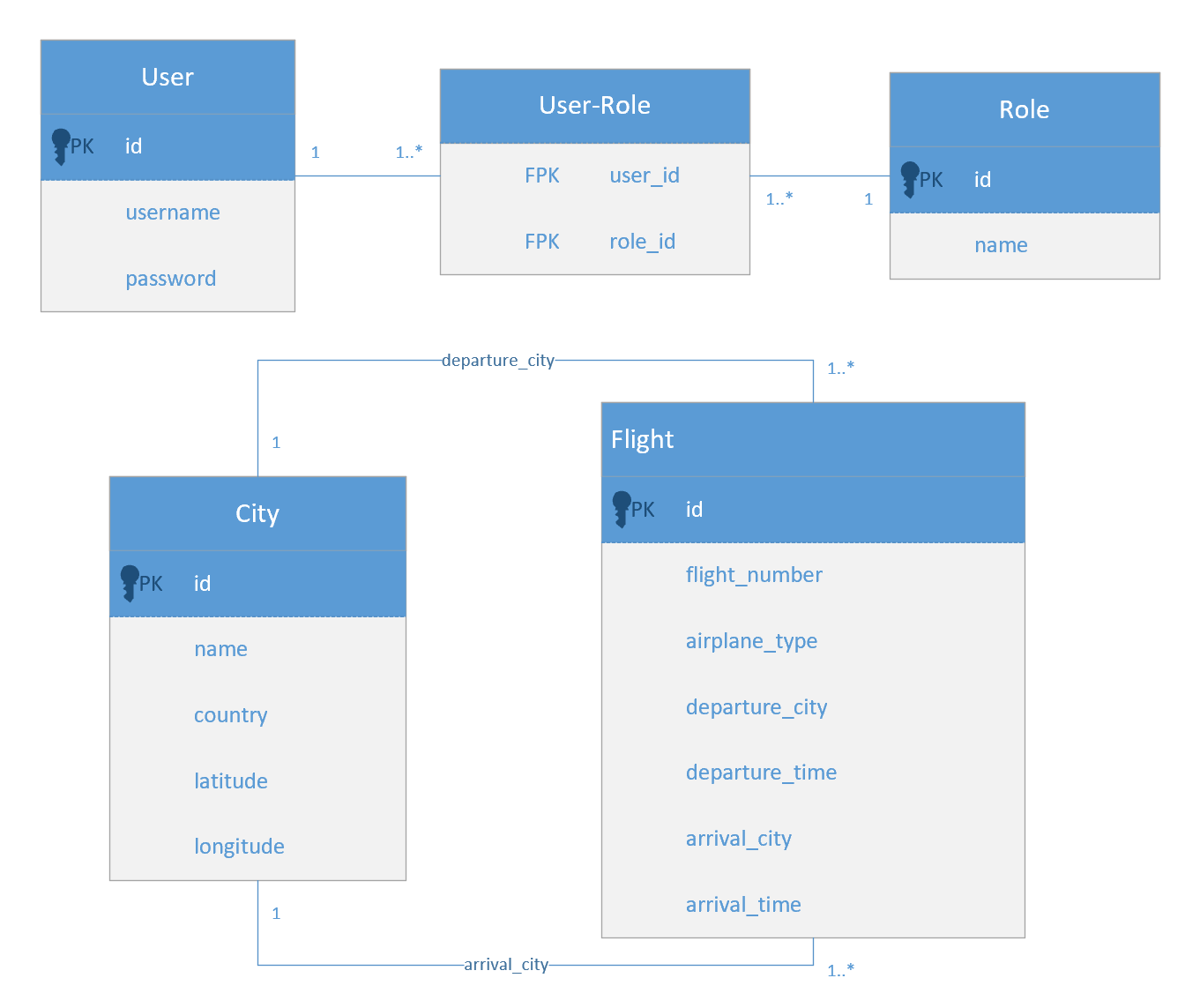
Deployment diagram

## **Database design**

The design of the database is highly dependent on the requirements of the application, because they influence both the entities and the relationships between them.

The following entities are included in the database:

* User: stores basic authentication information for the users of the application
* Role: used to define the access rights for a user
* User-Role: maps users to roles in a Many-To-Many relationship, which offers the flexibility to assign multiple roles to a user, in case of implemented a more fine-grained role-based access control policy
* City: stores geographical information (country, name, latitude, longitude) about a city which hosts an airport
* Flight: main business entity, storing the following information – flight number, airplane type, departure location, departure date and time, arrival location and arrival date and time

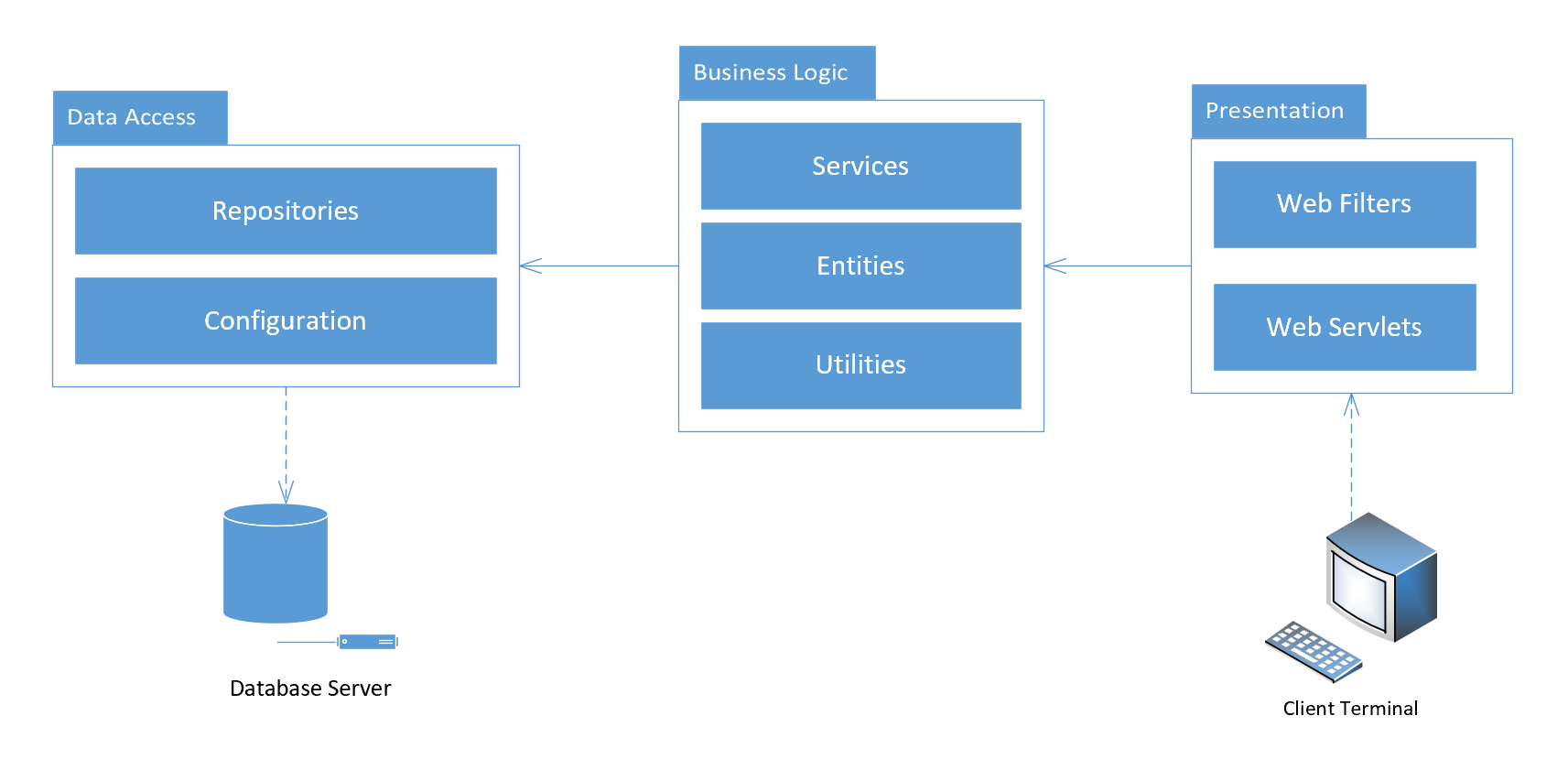
E-R diagram

## **Package architecture**

As previously mentioned, the application is layered. Each layer abstracts out one type of process and communicates with the layer directly below it in terms of some contracts established by a set of interfaces. The main layers are Data Access, Business Logic and Presentation.

The Presentation layer manages the communication with the client, from whom it receives requests and to which it sends back responses. The web servlets map the endpoints of the application and parse the requests, before delegating the work to the services, which process the request. The web filters have different functions, ranging from logging to application access rights management for the two types of users.

The Business layer is the most complex part of the system. It processes the requests that are parsed and forwarded to it by the Servlets. The Entities package contains the classes for the business objects, User, Role, Flight and City. The Services package contains the classes that process the data, generate the result and send the data that needs to be persisted to the data access layer. The Utilities package contains additional classes, that are used by the Services, like data transfer objects or filter objects.

Package diagram