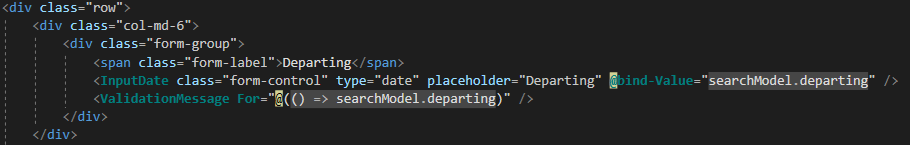
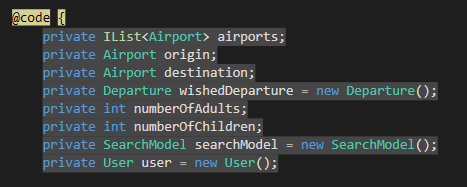
**Presentation Layer**

To communicate with user, blazor implements UI which is in form of razor pages, which hold html content and c# code.





For getting input for user blazor form is used, with validation. Values from those forms are binded onto c# objects and then used to perform some action, usually communicating with rest of the system.

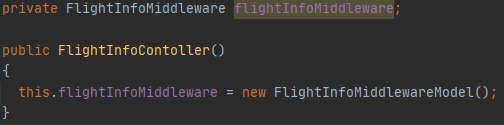
**Web Services**

To establish web service communication there must be client and server and both of these need to be connected to the internet, so they can communicate over it. Server, running on some port is exposing the services to the client, so that the client can invoke and call the service. Platform for our web service is HTTP protocol, meaning that mainly methods like GET,DELETE,PUT and POST are used. For security HTTPS or SSL was introduced, so that data are encrypted and decrypted, so that only sender and receiver can see it.

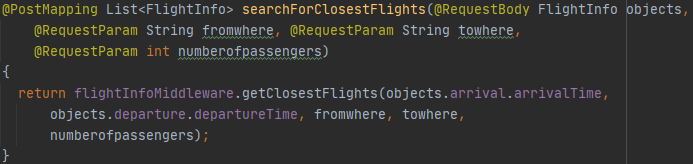
**Server side**



Classes on server side,in this case it is Middleware layer(implemented in Java), which are responsible for handling clients requests are called controllers, they are mapped, so that when there are more controllers, you can easily distinguish between them and call the right one.



Each controller holds and implements in constructor instance of middleware interface, so that they can call methods and perform logic.



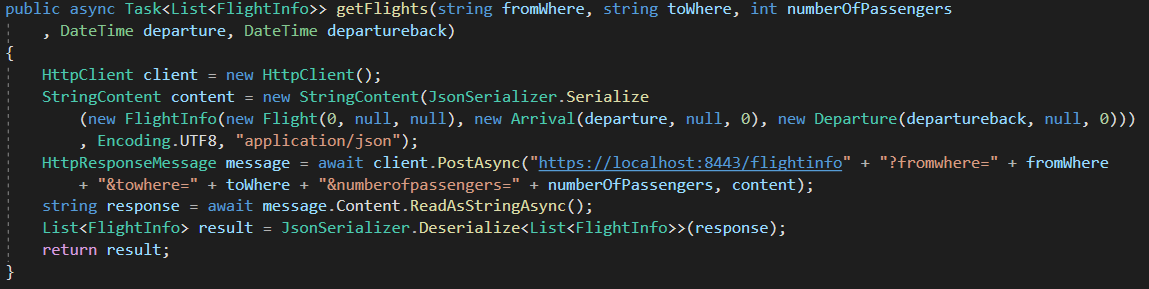
Example of controller method will be searchForClosestFlights method. With beginning annotation, it is set as a Post method, which returns a list of Flights. In argument section it requests body from http message, and three parameters from URI(Uniform resource identifier). Finally controller calls middleware interface, which holds all the logic and returns list of desired flights.

**Client Side**

Client is implemented in C#, it calls HTTP methods to be performed in Server and awaits answer.



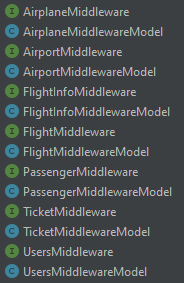
Somewhere in the code, method for searching flights is called and the result from server is awaited.



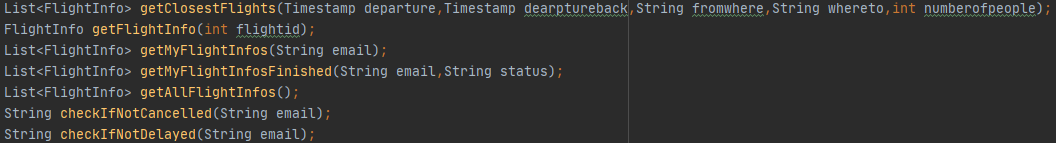
Then, requested service is called and method invoked. HttpClient is created, content of http message serialized into JSON and send as a content in invoked POST method. It is supplied with full uri, which contains port and controller mapping plus arguments. Then a response is awaited, deserialized from JSON and returned back to the client.

**Middleware Model**

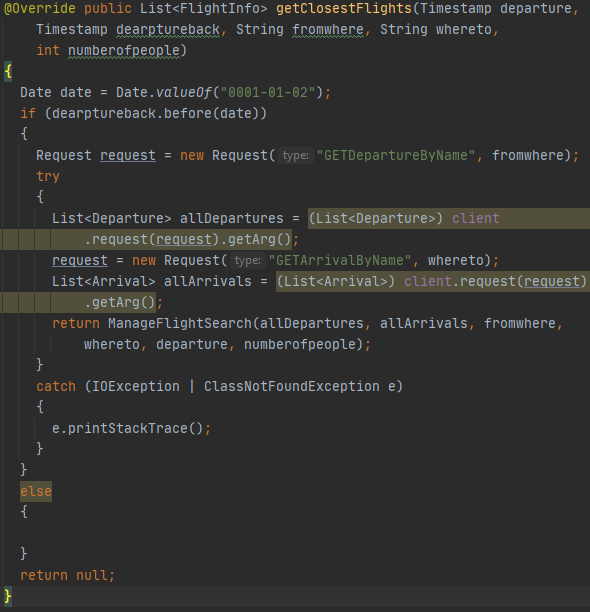
Models (Middlewares) are interfaces, which hold all the logic for the program. They are implemented by models and divided into fields, by objects they are working with.



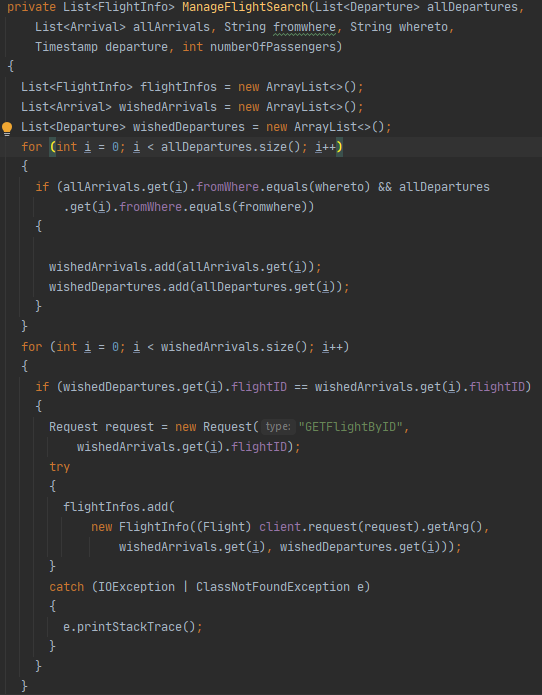
For example FlightInfoMiddleware holds these methods.



They are implemented in FlightInfoMiddlewareModel, so that method getClosestFlights look like this.



In this method, it is checked that the flight is one-way and then Departures and Arrival are obtained from persistence layer. Finally there is returned a result from ManageFlightSearch method.



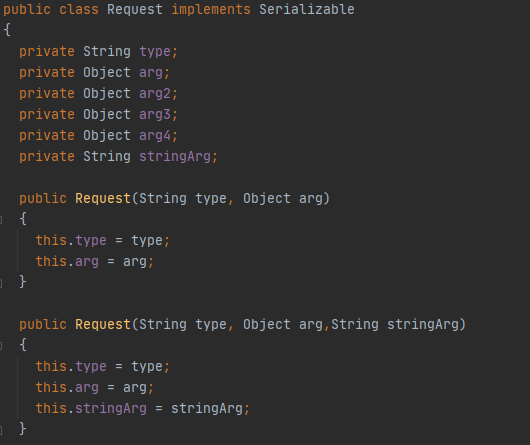
ManageFlightSerch method has two parts in first part only those flights that match departure location and arrival location are chosen.



Then dates are checked, with tolerance of three days around clients chosen date. The result is then returned and displayed on client side.

**Socket Communication**

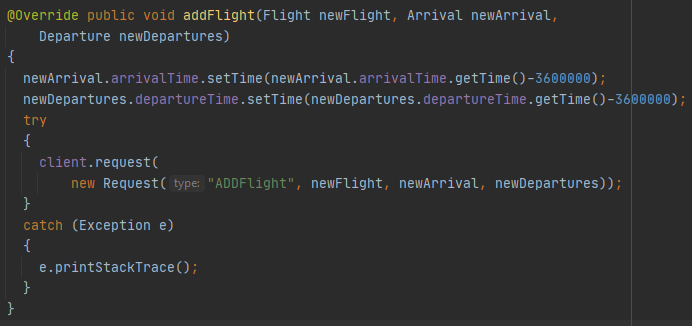
This communication occurs in our system between middleware(client) and persistence(server) layer. They communicate over TCP and custom made protocol, to ensure that data are safely transfered between those two and that both sides understand each other. To communicate DTO is used, in our case it is Request class.



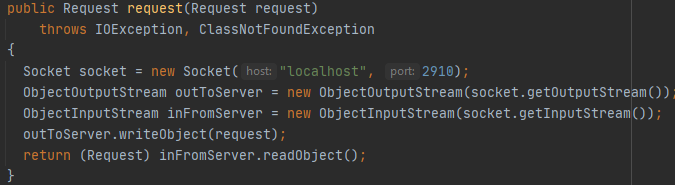
This class has Sting type which determines, what action is requested and many arguments, in which objects can be inputed. It of course implements serializable so that it could be converted to bytes and send over network.

**Client**

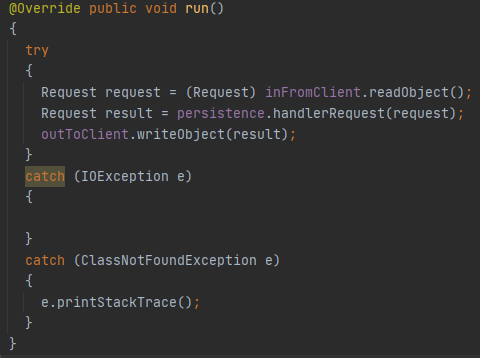
In middleware requests are created, with wished type and objects.



For example in method add flight request with type „ADDFlight“ is created and then is supplied with three objects, newFlight,newArrival and newDeparture. Then client is called and request method is invoked with request argument.



The client socket is then sending the request to persistence and then is waiting for an answer in shape of another request.

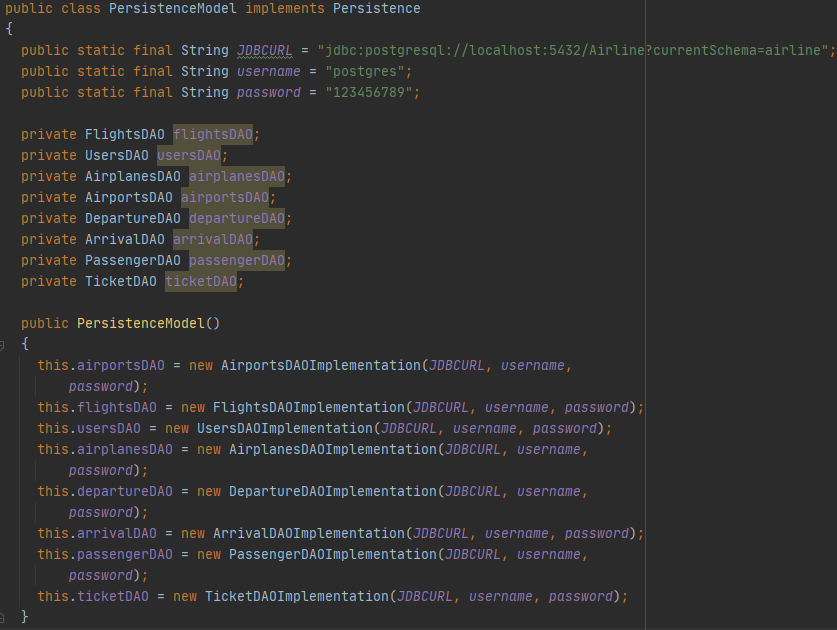


**Server**

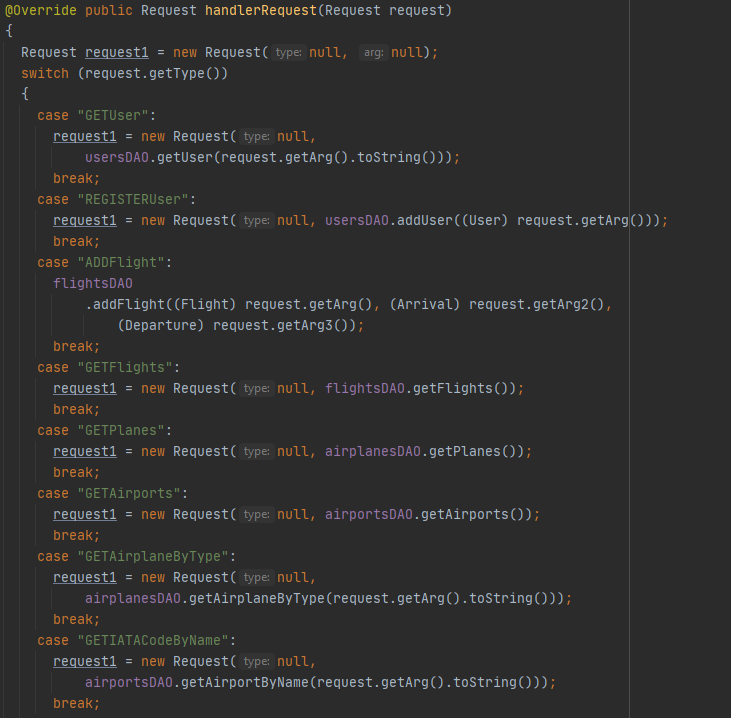
On server side (persistance), the socket is waiting and accepting client sockets. Then if the request arrives persistance is called with method handlerRequest.



Persistence is interface with one method, which is implemented in Persistence Model.



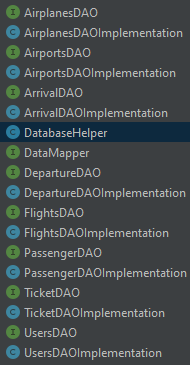
In PersistenceModel JDBCURL for database is set, same for username and password. Then it has instances of all the Data Access Objects, initializing them with all the information.



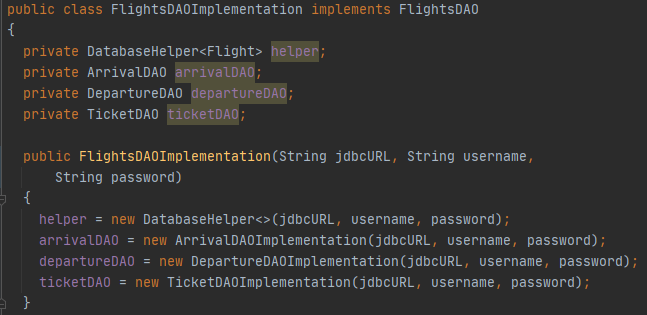
The final part of custom made protocol is the method handlerRequest, which reads all request types and does things invoked by client. Calling proper dao and performing action. In some cases returning something in form of Request.

**Database access**

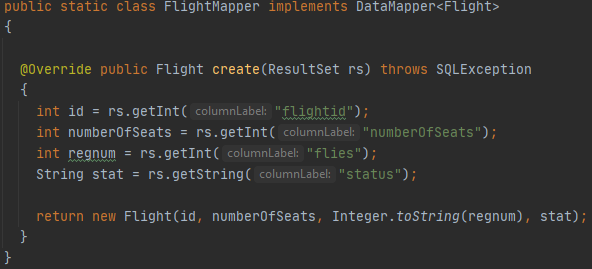
For a access to database from Java we are using jdbc. There are two important objects. First is databaseHelper, which prepares statements and holds general logic to map object from database to java objects and dataMapper, which is used to map these objects. For access to database Data Access Objects are used, divided by each table of database.



Each DAO holds methods, to be called by persistanceModel.



For example FlightsDAOImplementation has DatabaseHelper instance and some other DAO instances, which are initialized in constructor and is implementing its interface FlightsDAO.



In each DAO there is also an class which is implementing DataMapper and code to map that object properly.



Then, the methods are implemented with DatabaseHelper being called,sql command written and parameters supplied. In this example biggest id already in database is calculated and new flight with the biggest id plus one is inserted into Flight table and also departure and arrival is inserted into their respective tables.