Employee Shift Scheduler

Student: Toma Joksimović

Professor: Nebojša Bačanin Džakula

Course: Internet Software Architectures

**1.Introduction**

This software is a simulation of business software, concretly for management. It’s purpose is to manipulate with data for employees and shifts, and most important purpose is to make schedule for those shifts with employees. In this app is a very simple example of the algorithm that is used for scheduling shifts, which I have constructed, only already existable part is sorting with Insertion Sort algorithm.

This application architecture is based on 2 REST Spring Boot microservices, and one client app made in Spring which is tinny connected to those REST web services.

I have used REST web services because REST is much flexible than SOAP, mainly because it supports more standards such as JSON, YAML, XML etc. In this app, I am using JSON format for transfering the data, and it is much faster than XML format, which is better for modeling data, as we can see in configuration files for Spring and Spring Boot, or Maven, Android, Java FX etc.

This project is 2 layer architecture, because there is no database, inspite of that we have simulated database with Array Lists in Java.

Client App is CLI, without GUI, but user is driven with clear sentences and menus throw the application.

There is 10 mapping methods which are mapping to REST resources (URI-s), such as GET, POST, PUT, DELETE requests.

Also in both services we have Controller and Model. Model is implemented as DAO (Data Access Object) service which is Autowired into the controller for using DAO methods in mapping methods.

With this app, managers of employees won’t waste so much time by scheduling employees.

**2.Review of Used Technologies**

First for both web services we have used Spring Boot preconfigured framework with REST, Hibernate and Maven frameworks also, but Hibernate is not used because we don’t have real database where we can use Hibernate. Maven is great for managing whole project structure, for example in downloading external libraries.

Spring Boot is great preconfigured framework and more easier to work than casual Spring framework. Spring Boot is based on MVC (Model View Controller) application architecture but we will Model use as DAO (Data Access Object) layer. In this case Controller is implemented as Rest Controller and DAO class as Spring Boot Component so it can be Autowired.

In both web services (Employee Service and Shift Service) there is one of the most important xml files which is named **pom.xml**.

That file is used for injecting all dependencies into the project, storing information about SDK, project description, repositories etc.

Because those two web services are targeted on the same IP address (localhost -> 127.0.0.1), their socket (IP address + port) needs to be different, but because they are localy developed, they have only 2 different ports

Default port is 8080, and it is however used for **Employee Service** because there is not app that taked this port, so file **application.properties** in this web service is empty.

In other web service **Shift Service** we put into **application.properties** file server.port=1313, so this service will communicate throw port 1313 with client application.

Client application is developed with Spring framework so it has spring.xml file as configuration file but it is empty because there is no need for creating concrete objects in this app, because its main perpose is exchanging data with those 2 web services.

Also in this Client Application I have used apache http client external libraries for connecting application with web services.

There is special class with static methods for sending requests and receiving responses from the web services.

In one we can use either GET or DELETE method, depending on that what is sent by argument that is used as type of method, while in other static method we have same situation with POST or PUT HTTP methods for requests.

**3.Implementation**

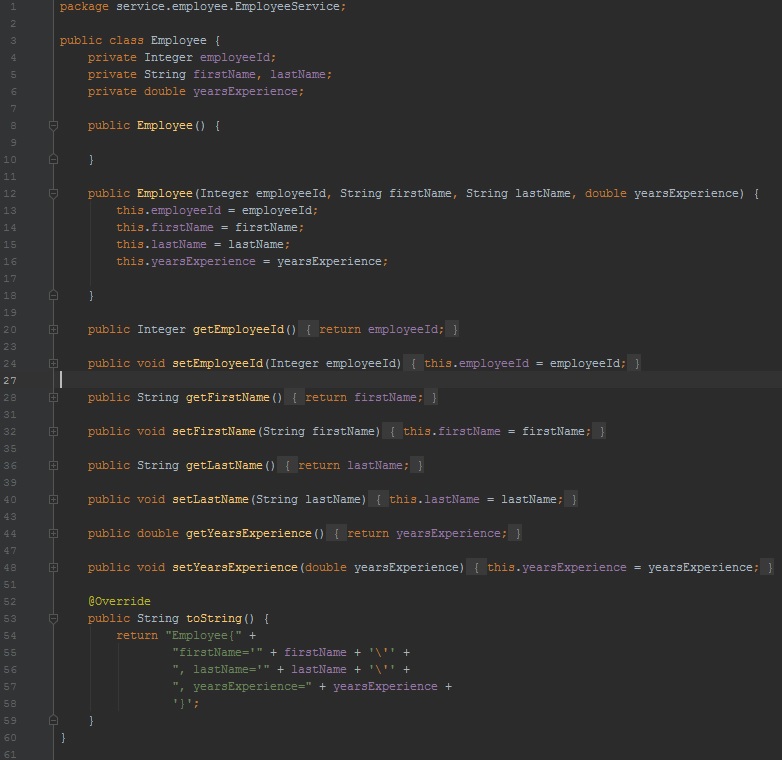
Now we will show implementation for Employee Service, then Shift Service and at the end Manager Client application.

**3.1 EmployeeService**

Service has internal Spring Boot environment and its own web server that uses port 8080 as default port

**3.1.1 Employee.java**

This class just shows basic informations about one employee that needs to be inserted in required shift. Class have private attributes, default and constructor with arguments, getters and setters and **toString()** overridden method.

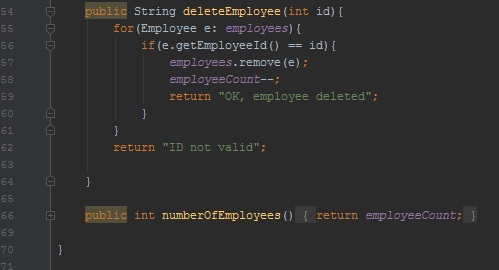
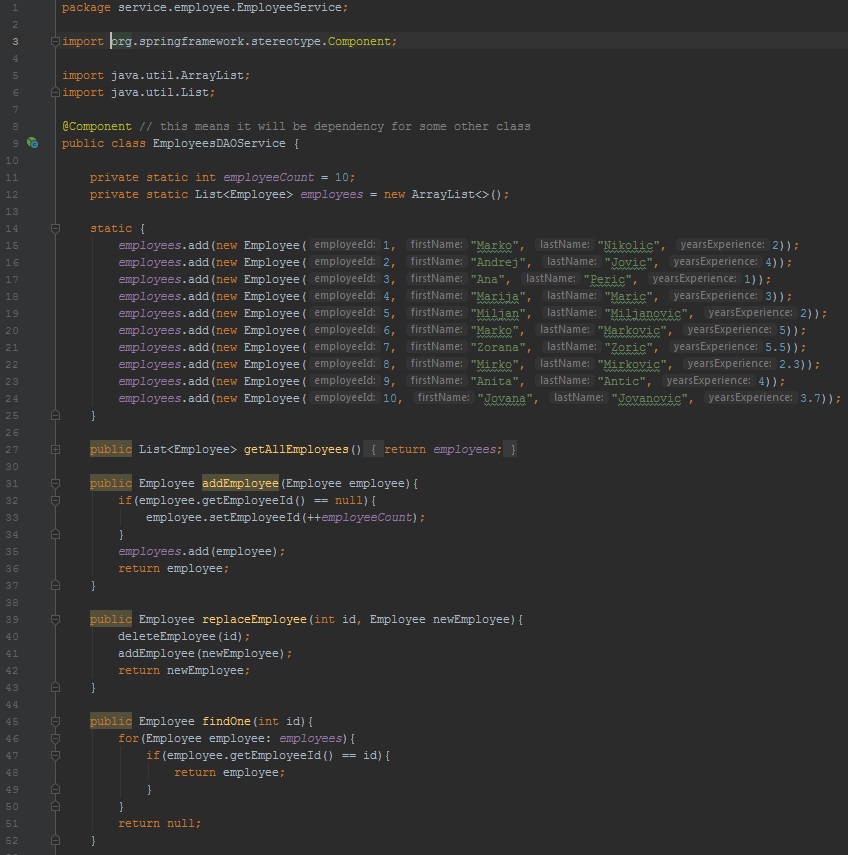


**3.1.2 EmployeesDAOService.java**

Class is Annotated as Spring Boot Component that will be automatically injected into Controller.

This class have 2 static fields. One is data about number of employees, other is simulation of database table as Array List of employees.

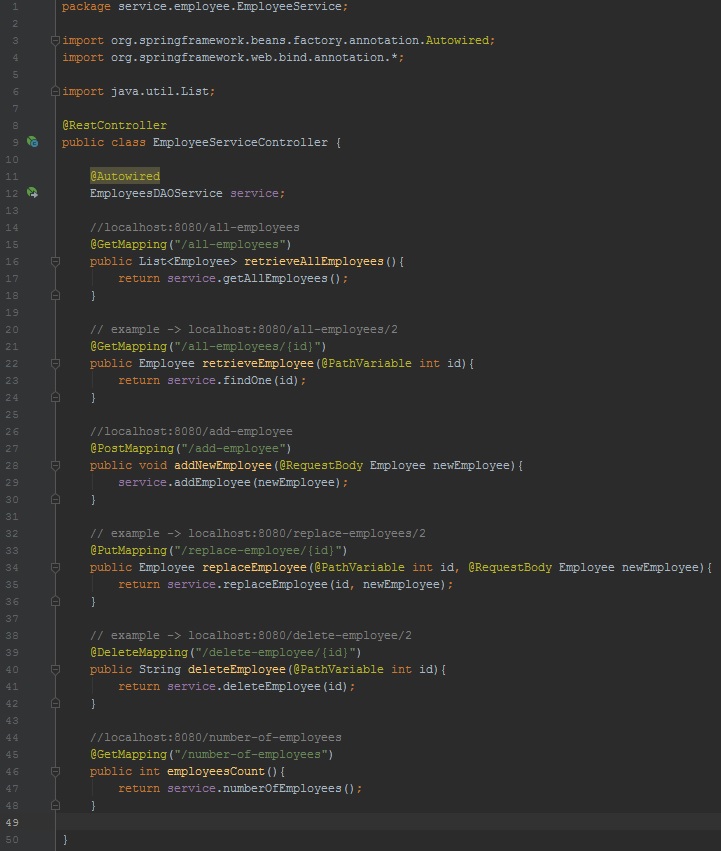
Also it has static block for adding initial employees into the list. There are also methods that will be called into Controller mapping methods which are displaying all employees, only one specified employee, deleting employee, replacing one employee with another, showing total number of employees, and for adding one employee into the list.



**3.1.3 EmployeeServiceController.java**

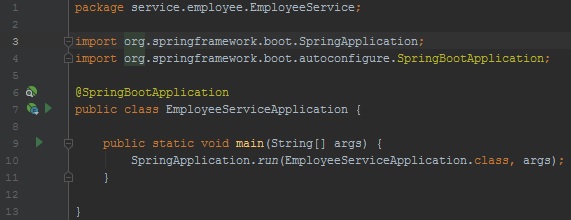
This class is implemented as Rest Controller and in this class Spring Boot automatically inject DAO service object with **Autowired** annotation.

Methods in this class are all annotated with HTTP methods which will be used for those resources mapped on specific URI. This annotations have URI routes on which we are calling this method defined below annotation. **@PathVariable** is here an integer that is taken from the address path and used in the method and **@RequestBody** is data taken from body of the client request which data will be used for creating object in services as new Employee.



**3.1.4 EmployeeServiceApplication.java**

This class is the main class which is calling file for this class in Java bytecode via run method of Spring Boot internal environment, so by that our service will be up if there is no errors.

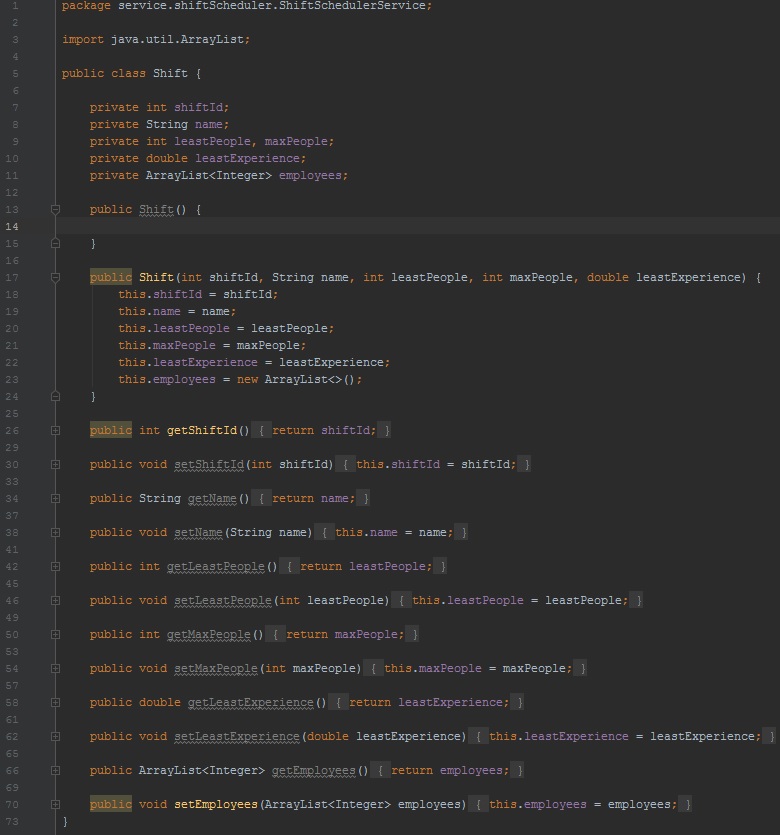


**3.2 ShiftService**

This service uses port 1313 which is added in application.properties file in resurces folder.C:\Users\Korisnik\Desktop\Zavrsni projekat\docs\Untitled.jpg . This service is used for manipulating with data for shifts and scheduling employees into shifts by special algorithm in the client application.

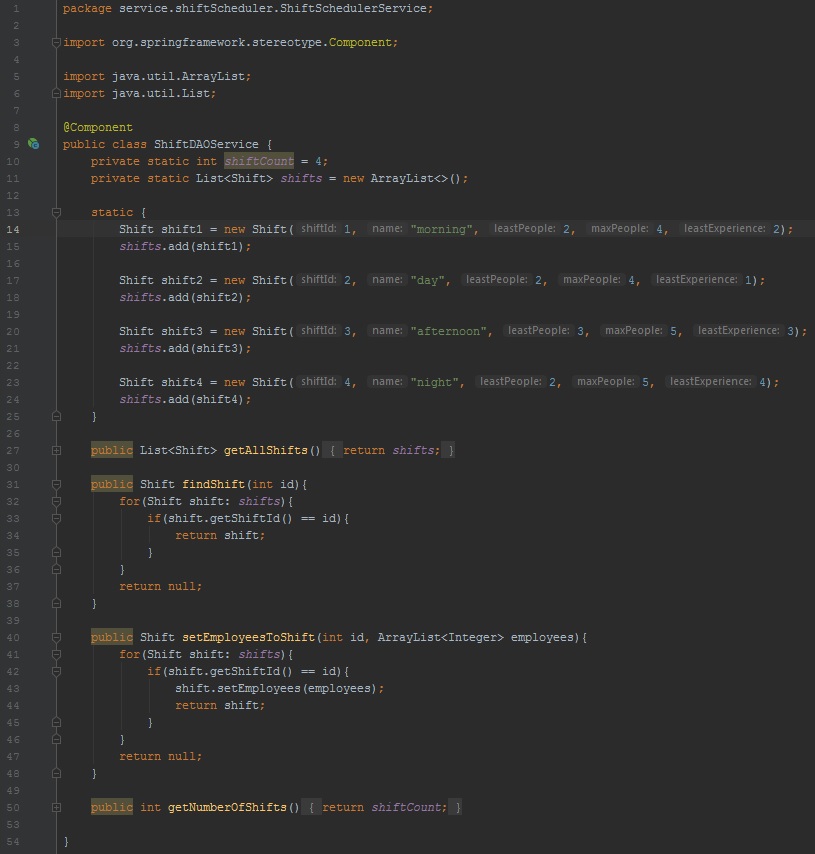
**3.2.1 Shift.java**

This class is used for creating shift objects and it has all necessary data and **constraints about it as leastPeople, maxPeople and leastExperience**. Everyting else is just constructors, getters and setters that are easy to understand. List of employees is empty at the beginning but letter the algorithm will insert employees in it. This class is simulated as template for one record in the database table which is not created here.

****

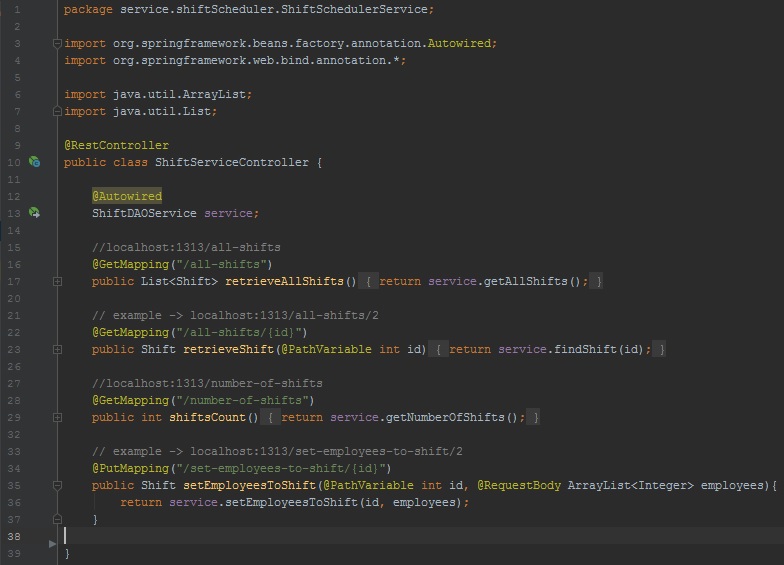
**3.2.2 ShiftDAOService.java**

This class is implemented as Spring Boot component with **@Component** annotation for it at the beginning for Autowiring as Spring Bean in Controller class. It has static fields for number of shifts and static List of shifts defined and added in static block. Also there are 4 methods which are called for sending requests on specific URIs defined in controller. Those methods are used for modifying data in shifts or showing shifts. One important method is **setEmployeesToShift()** which will add employees ordered by algorithm on the client for concrete shift.

****

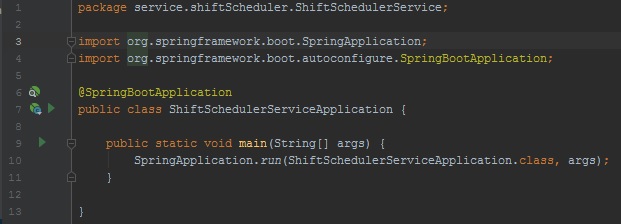
**3.2.3 ShiftServiceController.java**

This is REST Controller so we assigned annotation for that so Spring Boot will recognize it. DAO service is automatically injected by **@Autowired** annotation. We have 4 mapping methods with HTTP methods annotations and paths to specifis resources which are readed or posted. There are 3 **GET** methods which are triggered and which are showing data for all shifts, specific shift by id, number of shifts and one the most important method which uses **HTTP PUT** method for setting list of employees given by algorithm for specifis shift.



**3.2.4 ShiftSchedulerServiceApplication.java**

This is main class for Shift Service, as previous Employee Service, same procedure is for this compiling. Run method from Spring Boot will compile main class which will start service on port 1313 if every other component is started and all dependencies are injected without errors.

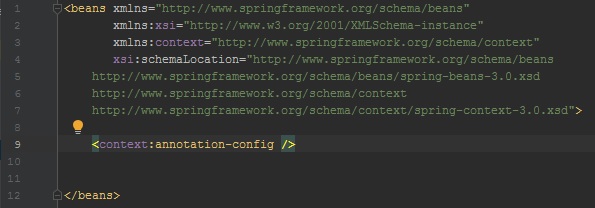
****

**3.3 ManagerClientApp**

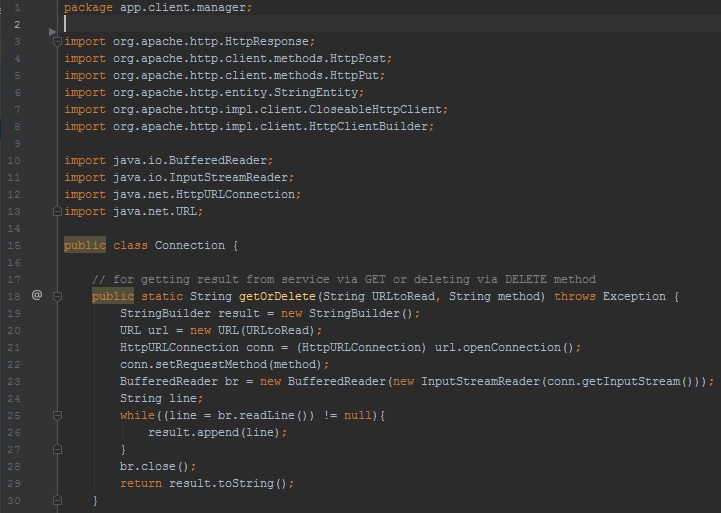
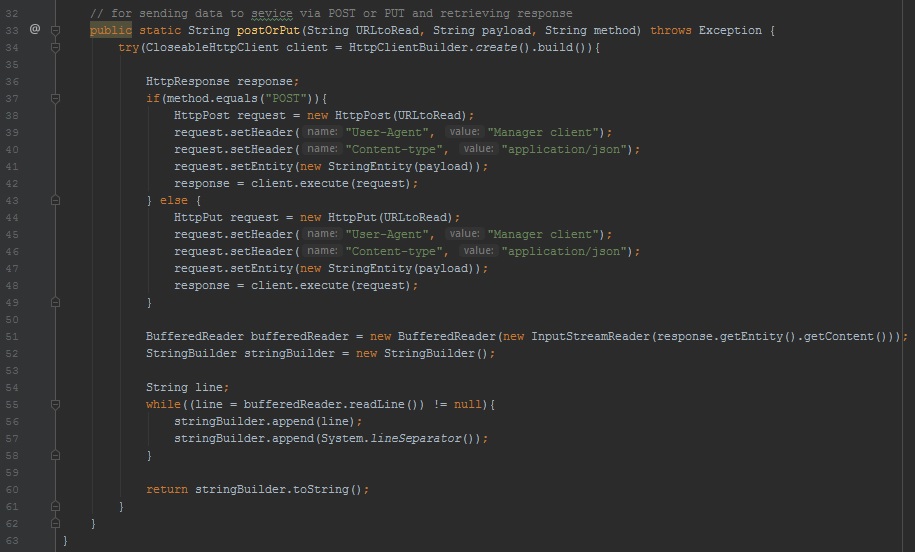
This Spring client application that communicates with 2 web services above via web and transfer data. It also includes class Conncetion for connecting to those services and important algorithm for scheduling employees which is the key part of this software.

**3.3.1 spring.xml**

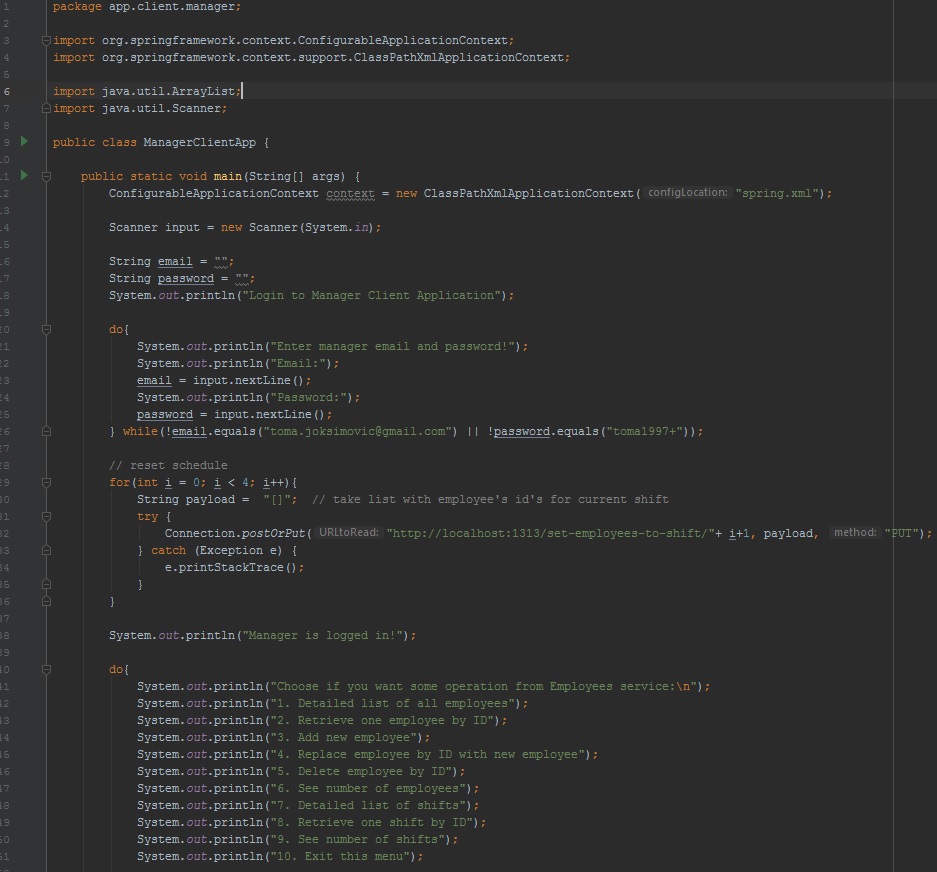
This is empty Spring xml configuration file, because at the end there wasn’t need for additional beans because everything is in transferring data between services and app, so we only have application context and links for Spring.



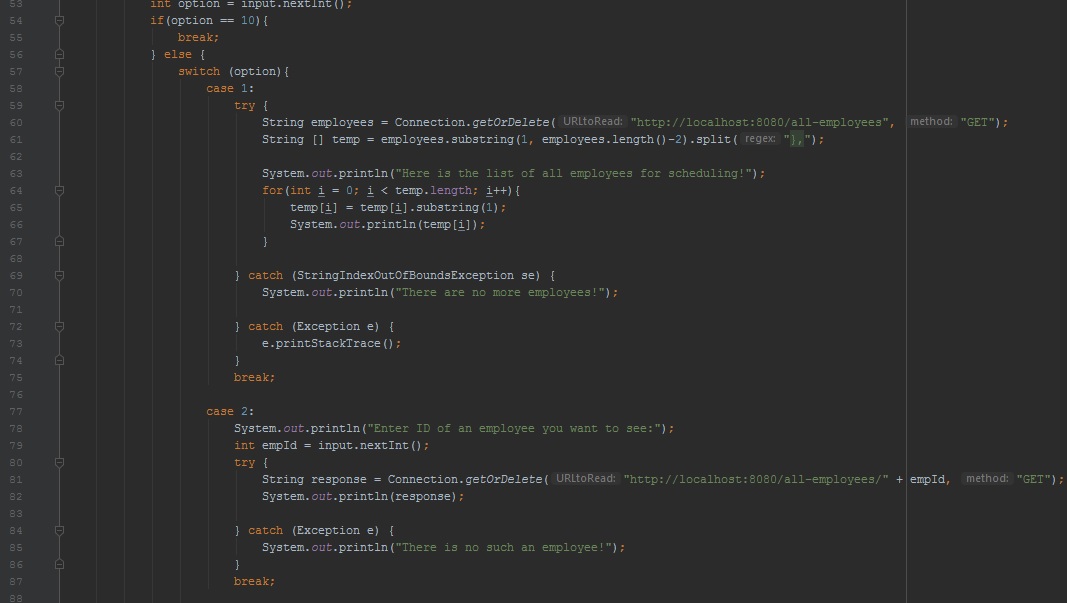
**3.3.2 Connection.java**

This class as mentiones is used for connecting to web services with 4 HTTP methods. In first static method we are using either GET or DELETE and by that we will send request to given URL with given method, and after we get response, we will just read it and return it. Similar is done in 2nd method except we are setting header in request and in body we set our payload data for sending to server. Important is that we use external org.apache.htttp libraries for connection.

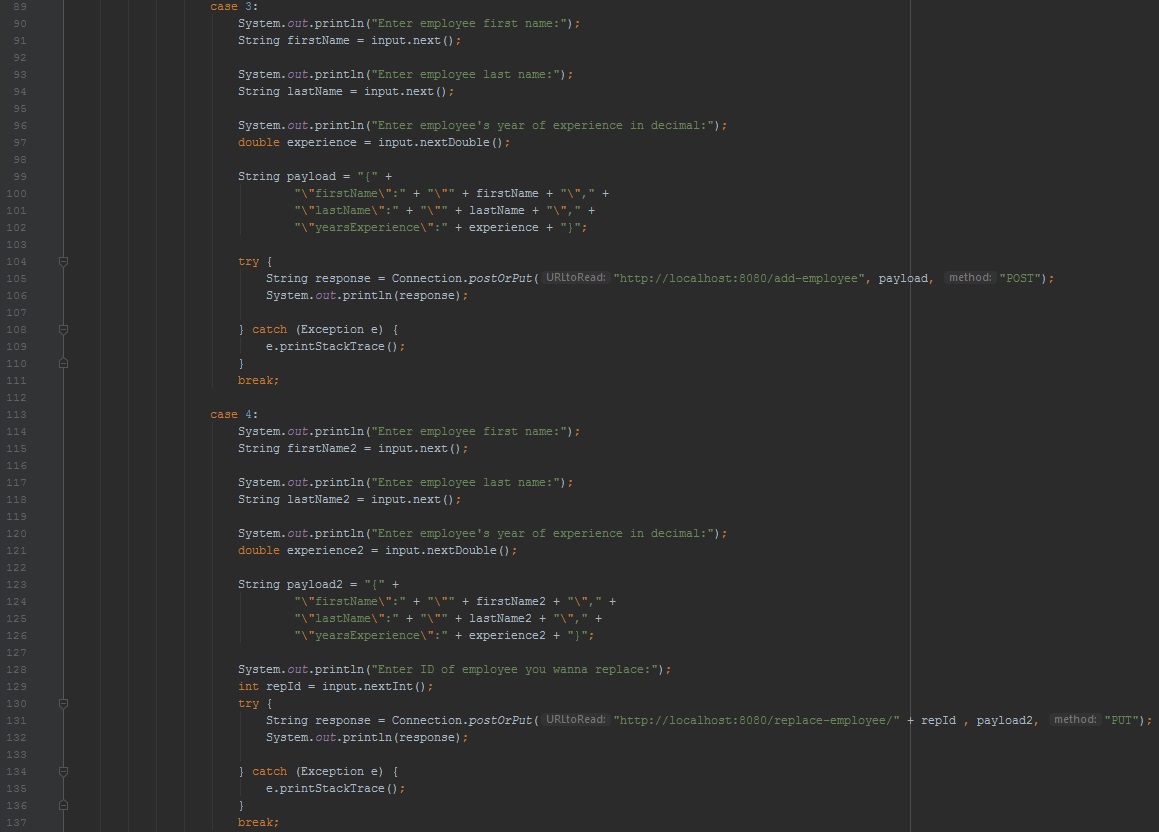
**3.3.3 ManagerClientApp.java**

****

In the main class first we will just make empty Spring context, but never used. Later manager should login with hardcoded parameters to the application. After that we will just reset schedule to empty by sending PUT request with empty list if algorithm was previously working so lists are not empty ( we do this because later in processing data with split.() command it won’t split by comma correctly if therea are elements in list). After that we are showing menu for manipulating data for 2 web services.

****

First manager will enter an option. It is check if option is 10 then break infinite loop. I have used switch clause for all options in menu. First is for sending GET request for all employees to retrieve, and lest them through the loop. Second is to retrieve data only for specified employee by giving id to the end of the path to resource and displaying his informations.

****

Case 3 and 4 are similar because we are entering data for new employee, only difference is that in case 3 we are making new employee so we are using POST method, and in case 4 we are replacing existing employee, so we additionally enter id of employee we want to replace and we are using PUT method. We are making a JSON object and storing in payload variable which we will send as request.

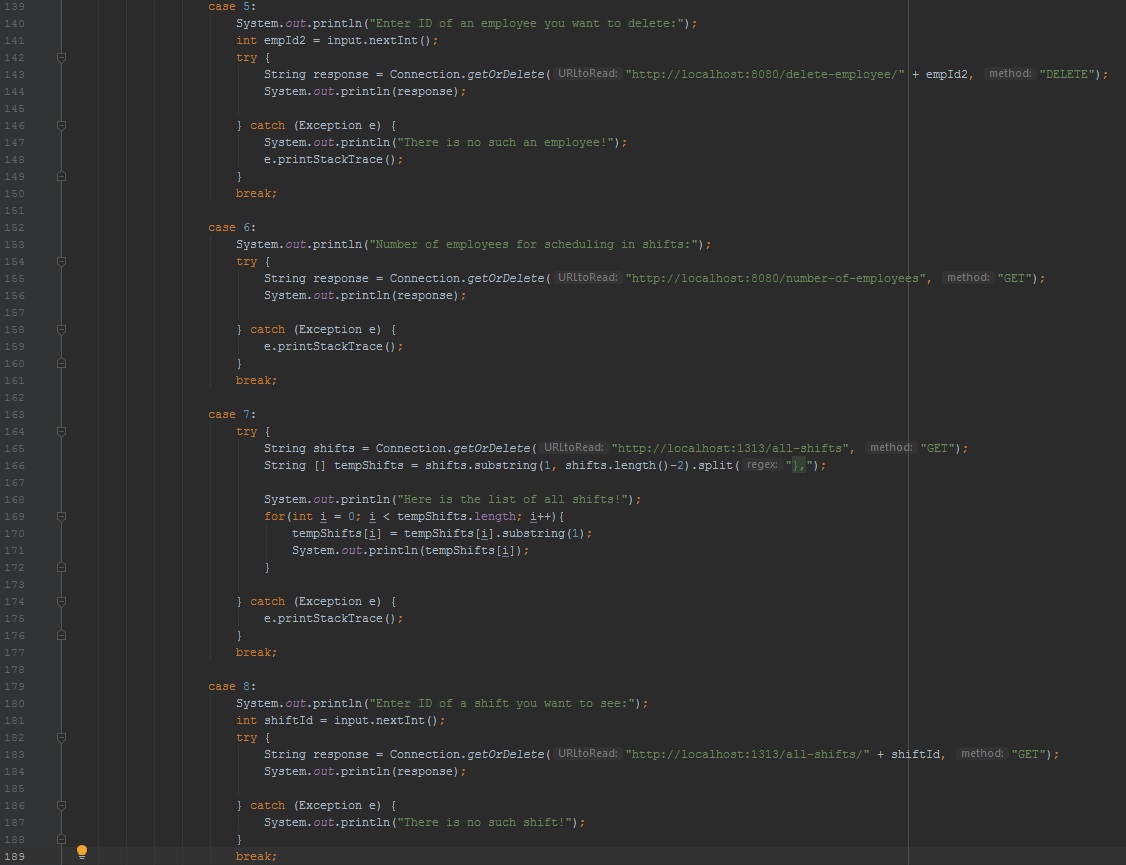
In case 5 we enter id of an employee we awant to delete and send as HTTP DELETE method, if Exception occurred, it means that there is no such employee we entered by id.

Case 6 is simple so we just send GET request method to see number of employees.

In case 7 we send request to other service to retrieve data about all shifts, similar as case 1.

Case 8 is similar as case 2 but instead we are entering id for specified shift and retrieve information about that via GET method.

If shift is not found, and response is some of 4xx, we well get Exception message.

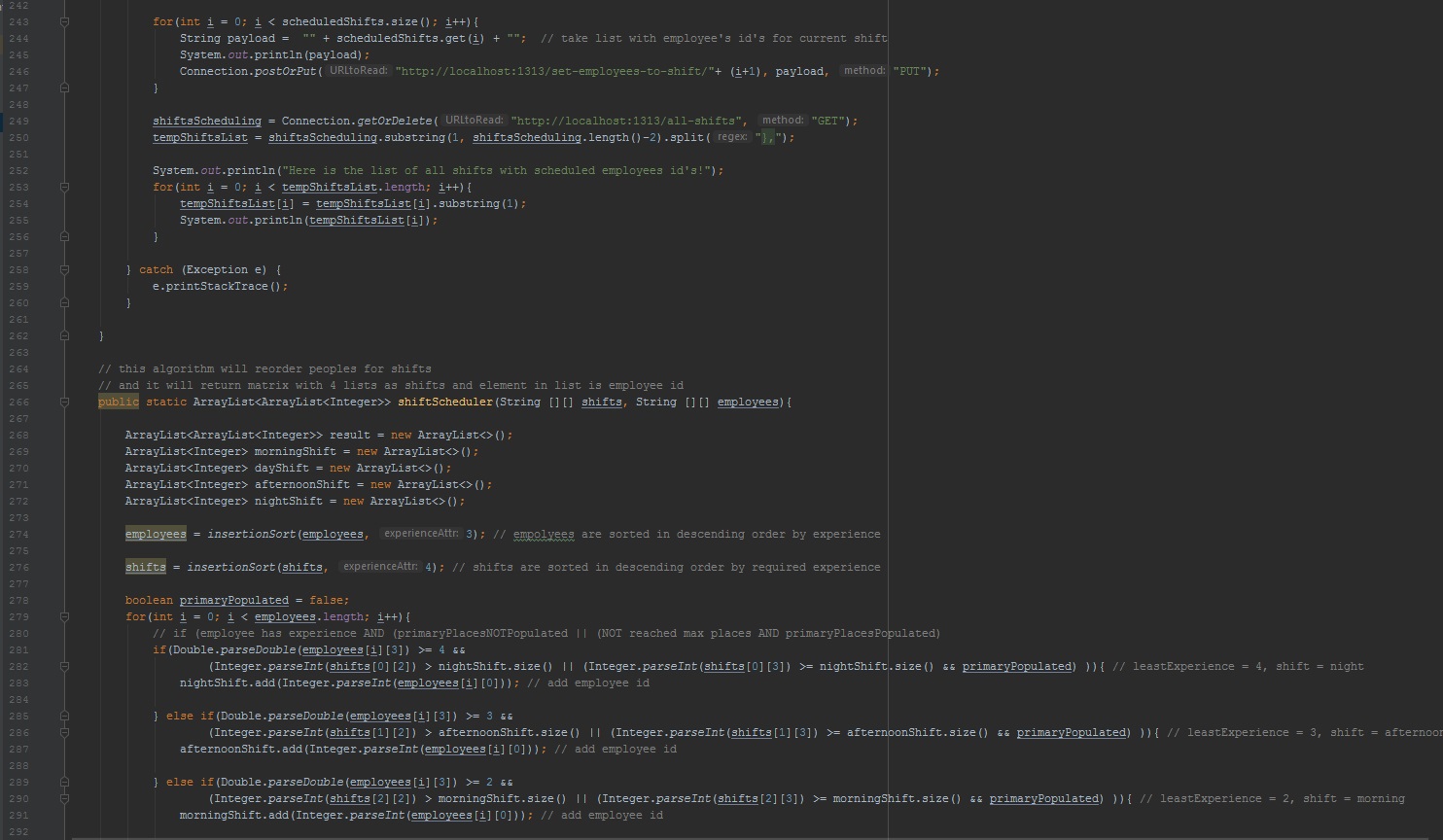
****

In case 9 we just send a GET request to see number of shifts. Default case is when user entered wrong option.

After that we take from both services all data from the list, then split them by elements and then store it in array then again we split by fields and put values in array so basically we make two matrices for employees and shifts.

Then we use that for input in shiftScheduler() algorithm. After algorithm give us back 2D ArrayList of Integers( for every shift there is a list of employees IDs, which will work in that shift. So in for loop we are putting for every shift list of employees in JSON and sending with PUT request to Shift Service, which will set for every shift an array list of employees. At the end of main class we just send again GET request for all shifts just to see updated shifts with employees lists.

Algorithm begins with initializing 4 lists for 4 shifts, and we sorts arrays of employees and shifts in descending order with Insertion Sort algorithm which is given at the end. We just modified it to sort by experience as criteria.

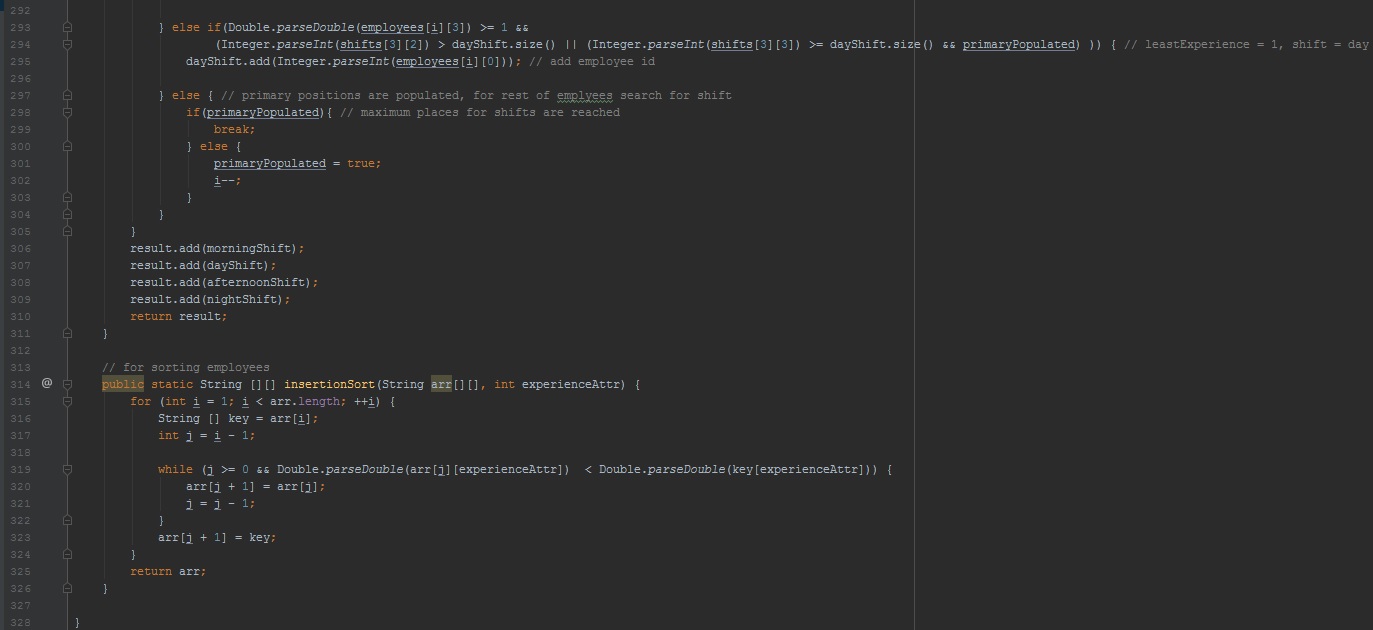
****

****

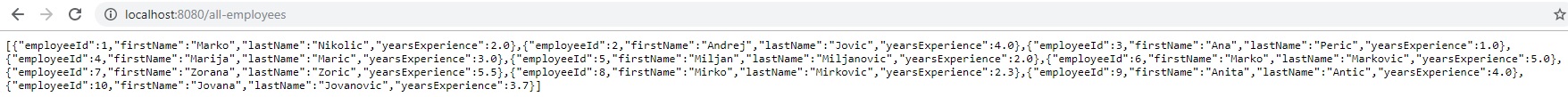
After we are looping through every employee and we order employee by his experience to specified shift and while least people for shift are not inserted.

Also if least number of employees is reached for every shift, now we order employees while maximum number of employees is reached or while there is no more employees.

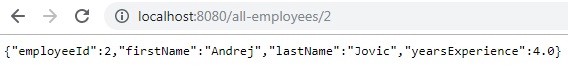
At the end we add to result 2D list all shift lists of employees and return that as result.

**4. Testing of HTTP methods**

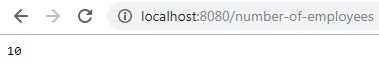
**4.1 Employee Service GET for all employees**



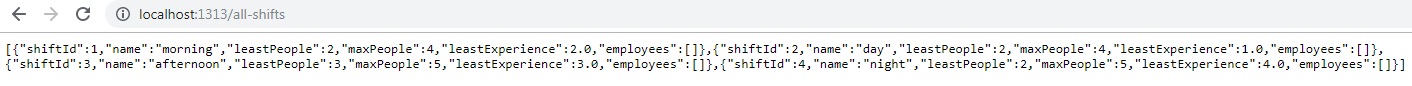
**4.2 Employee Service GET for one specified employee by ID**



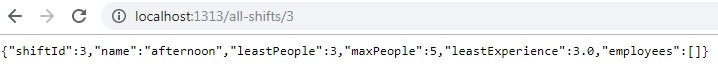
**4.3 Employee Service GET for number of employees**

****

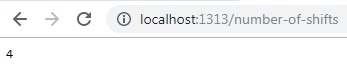
**4.4 Shift Service GET for all shifts**

****

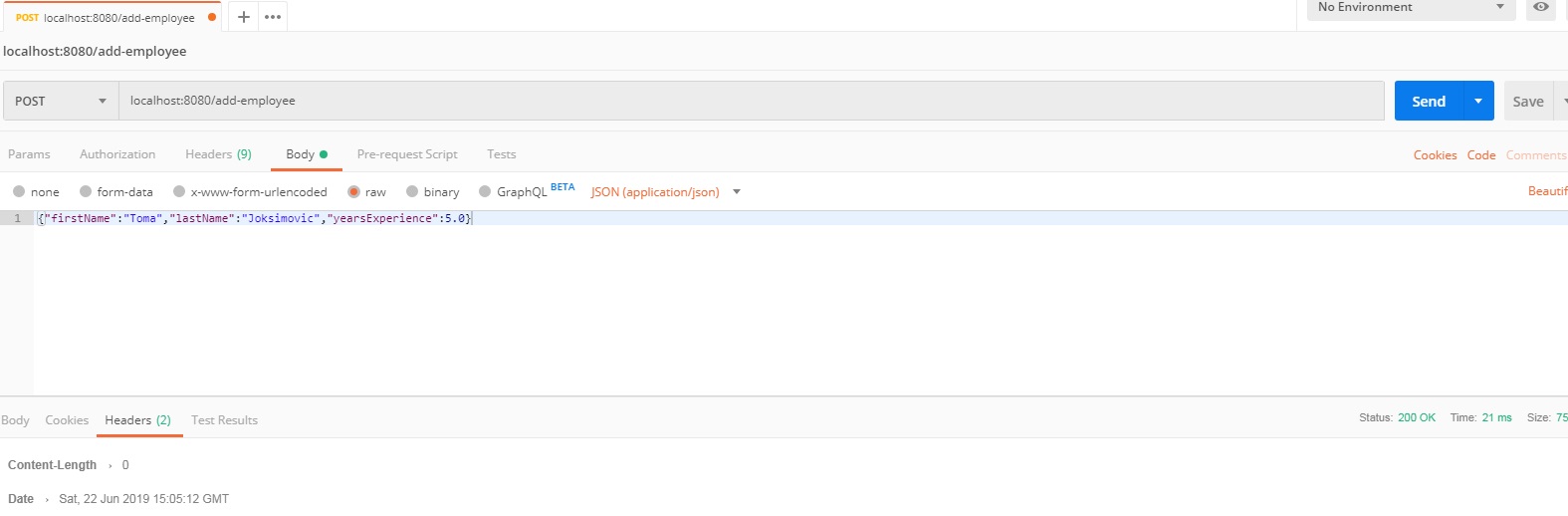
**4.5 Shift Service GET for one specified shift by ID**

****

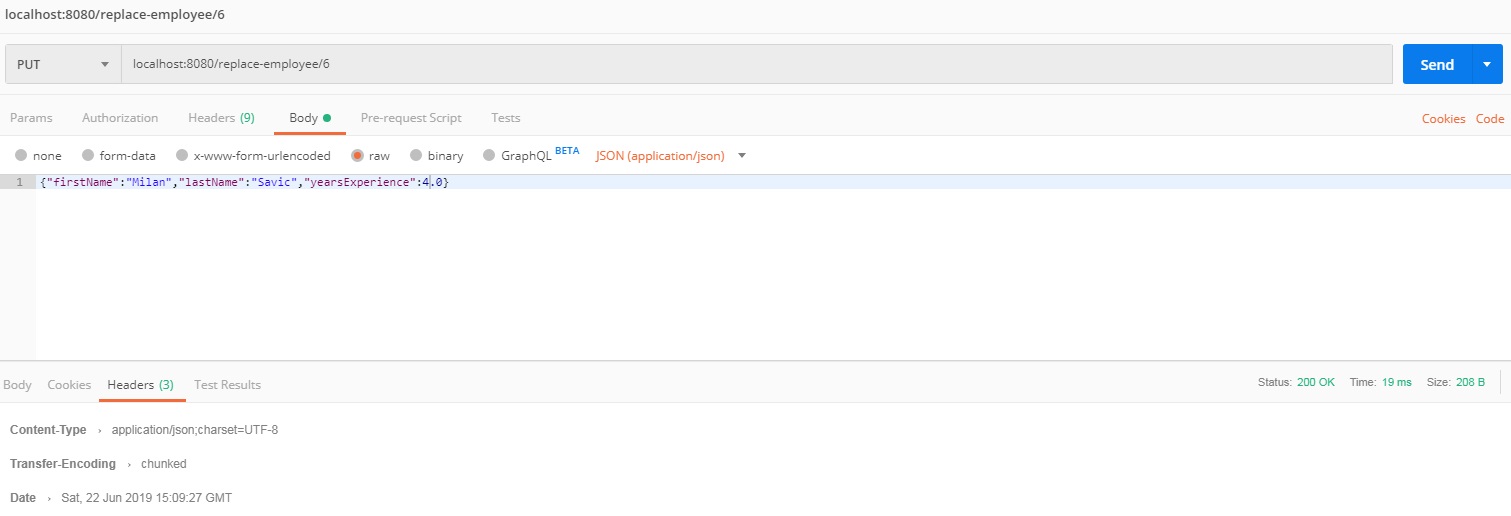
**4.6 Shift Service GET for number of shifts**

****

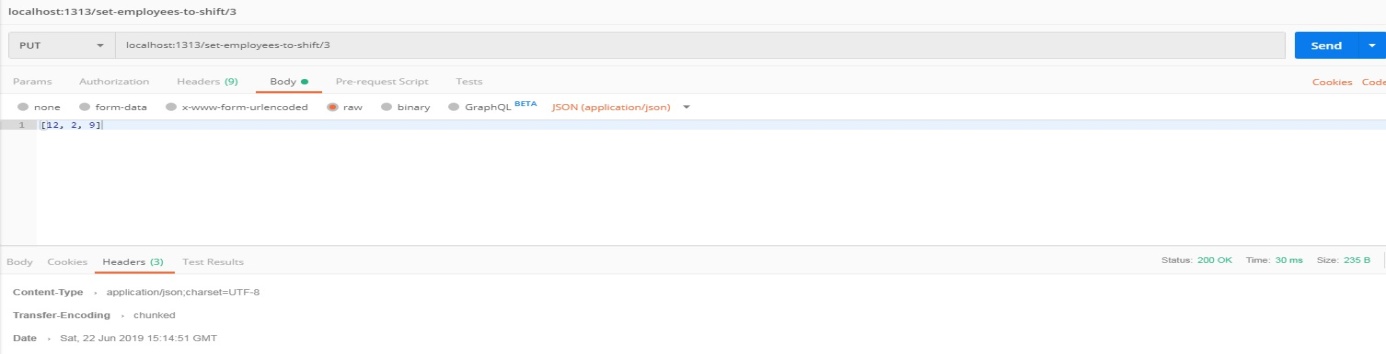
**4.7 Employee Service POST for adding new employee**

****

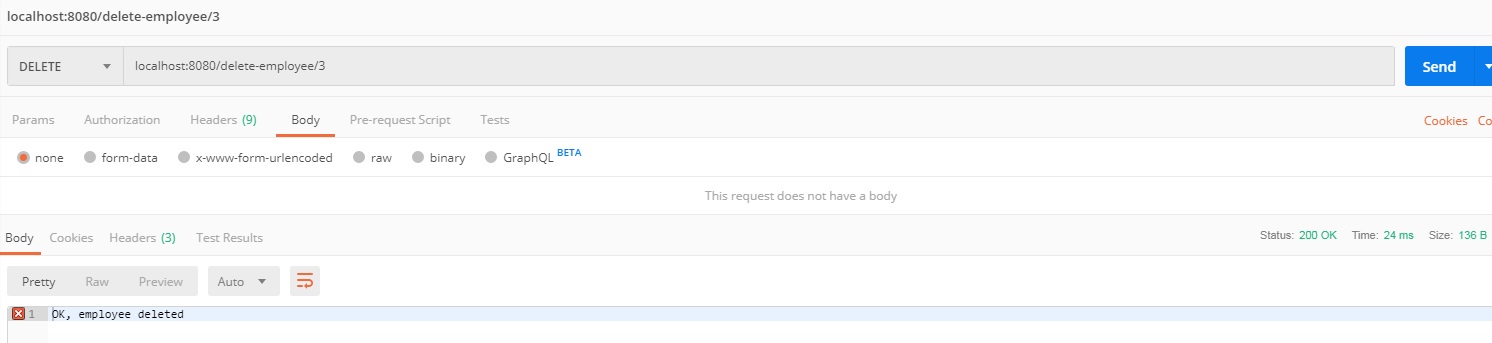
**4.8 Employee Service PUT for replacing one employee with other by specifying ID**

****

**4.9 Shift Service PUT for setting employees list for specified shift by ID**

****

**4.10 Employee Service DELETE for removing specified employee by ID**

****

**5. Conclusion**

This is just simple example of an application that can be used for easier management of employees shifts. There are a lot of harder algorithms that use stronger and optmizied heuristics because in real life there are more complicated shift planning problems with a lot of constraints for shifting by side of shift and also by side of employee. You can check for Local Search (**Tabu Search, Simulated Annealing, Late Acceptance**). Algorithm which I called ShiftScheduler in this application is designed by me, there are only one part that already exists as algorithm and it is trivial and not so optimal Insertion Sort O(n^2), and other parts are imagined for this project, with poor heuristics.

Generally software has microservice architecture where microservices are tinny connected to client application as **independent services**. Transfer of data is fast because it is via **JSON** format which **REST** supports.

There is a great open source solution for this Shift Scheduling problem which is called **OptaPlanner** and you can take a look on the Internet and even see code.