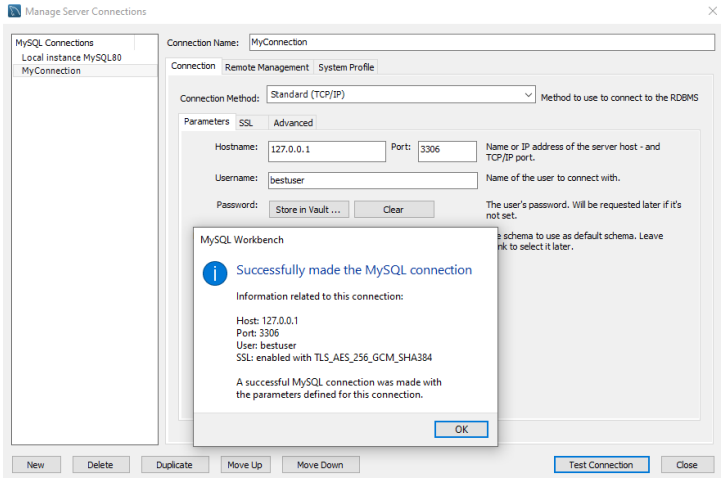
23. Spring Framework (Spring Boot, Rest applications)

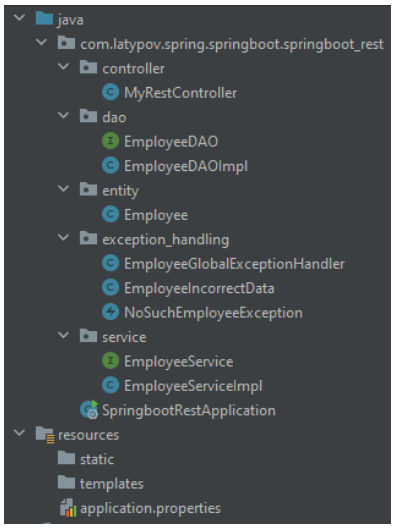
- Конфигурация приложений с помощью Spring Boot
- Создание Rest Api сервиса

Сценарий: Создание структуры БД в среде MySQL, реализация Rest-методов работы с данными, отправка запросов на сервер в среде Postman.

Конфигурация подключения в среде MySQL:



Структура проекта:



application.properties (параметры подключения приложения к БД):

```
spring.datasource.url=jdbc:mysql://localhost:3306/my_db?useSSL=false&serverTimezone=UTC
spring.datasource.username=bestuser
spring.datasource.password=bestuser
server.port=3333
server.servlet.context-path=/springboot-rest
```

Создание таблицы Employees:

```
CREATE TABLE my_db.employees (
  id int NOT NULL AUTO_INCREMENT,
  name varchar(15),
  surname varchar(25),
  department varchar(20),
  salary int,
  PRIMARY KEY (id);
```

entity.Employee (связь с таблицей Employees):

```
@Entity
@Table(name = "employees")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;
    @Column(name = "name")
    private String name;
    @Column(name = "surname")
    private String surname;
    @Column(name = "department")
    private String department;
    @Column(name = "salary")
    private int salary;
```

service.EmployeeService (сервисная часть, интерфейс с методами):

```
public interface EmployeeService {
    public List<Employee> getAllEmployees();
    public void saveEmployee(Employee employee);
    public Employee getEmployee(int id);
    public void deleteEmployee(int id);
    public List<Employee> findAllByName(String name);

}
```

service.EmployeeServiceImpl (имплементация методов сервиса):

```
@Service
public class EmployeeServiceImpl implements EmployeeService {

    @Autowired
    private EmployeeDAO employeeDAO;

    @Override
    @Transactional
    public List<Employee> getAllEmployees() {
        return employeeDAO.getAllEmployees();
    }

    @Override
    @Transactional
    public void saveEmployee(Employee employee) {
        employeeDAO.saveEmployee(employee);
    }

    @Override
    @Transactional
    public Employee getEmployee(int id) {
        return employeeDAO.getEmployee(id);
    }

    @Override
    @Transactional
    public void deleteEmployee(int id) {
        employeeDAO.deleteEmployee(id);
    }
}
```

dao.EmployeeDAO (методы обработки данных):

```
public interface EmployeeDAO {
    public List<Employee> getAllEmployees();
    public void saveEmployee(Employee employee);
    public Employee getEmployee(int id);
    public void deleteEmployee(int id);
}
```

dao.EmployeeDAOImpl (реализация методов обработки данных, использование EntityManager, методы merge(), find(), executeUpdate()):

```
@Repository
public class EmployeeDAOImpl implements EmployeeDAO {
    @Autowired
    private EntityManager entityManager;
    @Override

    public List<Employee> getAllEmployees() {
        Query query = entityManager.createQuery("from Employee");
        List<Employee> allEmployees = query.getResultList();
```

```
        return allEmployees;
    }

    @Override
    public void saveEmployee(Employee employee) {
        Employee newEmp = entityManager.merge(employee);
        employee.setId(newEmp.getId());
    }

    @Override
    public Employee getEmployee(int id) {
        Employee employee = entityManager.find(Employee.class, id);
        return employee ;
    }

    @Override
    public void deleteEmployee(int id) {
        Query query = entityManager.createQuery("delete from Employee where
id=:employeeId");
        query.setParameter("employeeId", id);
        query.executeUpdate();
    }
}
```

exception_handling.EmployeeGlobalExceptionHandler (создание исключений, аннотации @ControllerAdvice, @ExceptionHandler, ResponseEntity):

```
@ControllerAdvice
public class EmployeeGlobalExceptionHandler {
    @ExceptionHandler
    public ResponseEntity<EmployeeIncorrectData> handleException(
            NoSuchEmployeeException exception){
        EmployeeIncorrectData data = new EmployeeIncorrectData();
        data.setInfo(exception.getMessage());
        return new ResponseEntity<>(data, HttpStatus.NOT_FOUND);
    }
    @ExceptionHandler
    public ResponseEntity <EmployeeIncorrectData> handleException(
            Exception exception){
        EmployeeIncorrectData data = new EmployeeIncorrectData();
        data.setInfo(exception.getMessage());
        return new ResponseEntity<>(data, HttpStatus.BAD_REQUEST);
    }
}
```

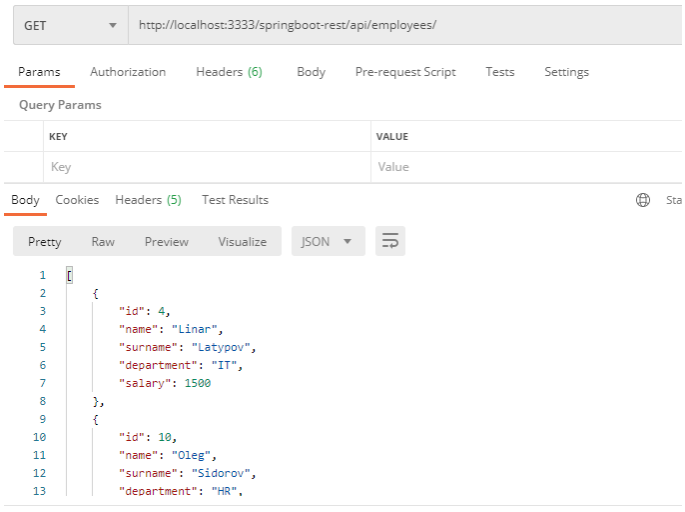
controller.MyRestController (реализация котроллера, аннотация @RestController):

```
@RestController
@RequestMapping("/api")
public class MyRestController {
    @Autowired
    private EmployeeService employeeService;

    @GetMapping("/employees")
    public List<Employee> showAllEmployees(){
```

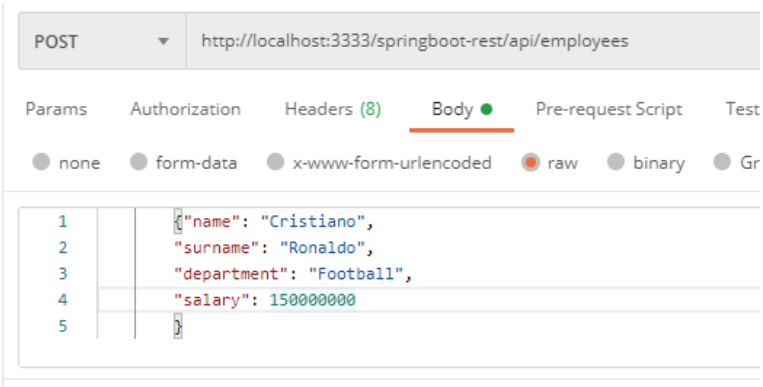
```
        List<Employee> allEmployees = employeeService.getAllEmployees();
        return allEmployees;
    }

    @GetMapping("/employees/{id}")
    public Employee getEmployee(@PathVariable int id){
        Employee employee = employeeService.getEmployee(id);
        if (employee == null){
            throw new NoSuchEmployeeException("There is no employee with ID =
" + id + " in Database");
        }
        return employee;
    }

    @PostMapping("/employees")
    public Employee addNewEmployee(@RequestBody Employee employee){
        employeeService.saveEmployee(employee);
        return employee;
    }

    @PutMapping("/employees")
    public Employee updateEmployee(@RequestBody Employee employee){
        employeeService.saveEmployee(employee);
        return employee;
    }

    @DeleteMapping("/employees/{id}")
    public String deleteEMployee(@PathVariable int id) {
        Employee employee = employeeService.getEmployee(id);
        if (employee == null){
            throw new NoSuchEmployeeException("There is no employee with ID =
" + id + " in Database");
        }
        employeeService.deleteEmployee(id);
        return "Employee with ID= " + id + " was deleted";
    }

}
```

Запуск приложения и отправка запросов через Postman

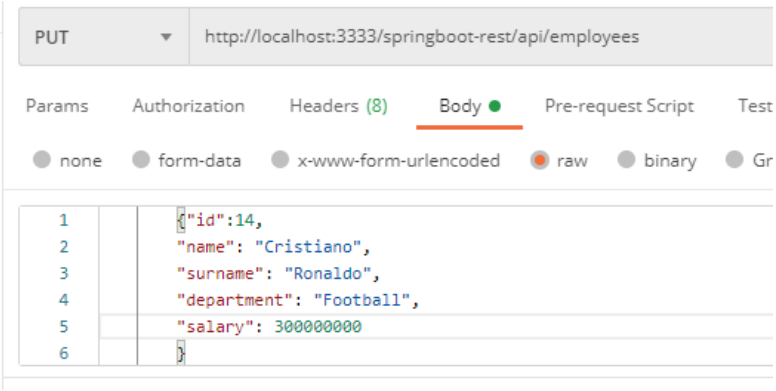
Get-запрос:



Post-запрос:



Put-запрос:



Delete-запрос: