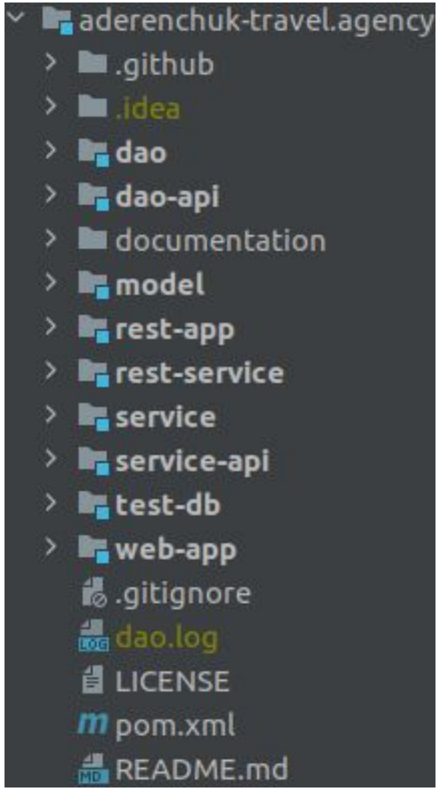
Web-application “Tour Agency” using Microservice Architecture

Creator: Dzeranchuk Artsiom

Project structure



```
aderenchuk-travel.agency
  .github
  .idea
  dao
  dao-api
  documentation
  model
  rest-app
  rest-service
  service
  service-api
  test-db
  web-app
  .gitignore
  dao.log
  LICENSE
  pom.xml
  README.md
```

Data access object (DAO) part. A pattern that provides an abstract interface dao-api to some type of **database** or other persistence mechanism.

Database part. Includes SQL scripts for initialization a configuration class that connect DB with Spring Boot

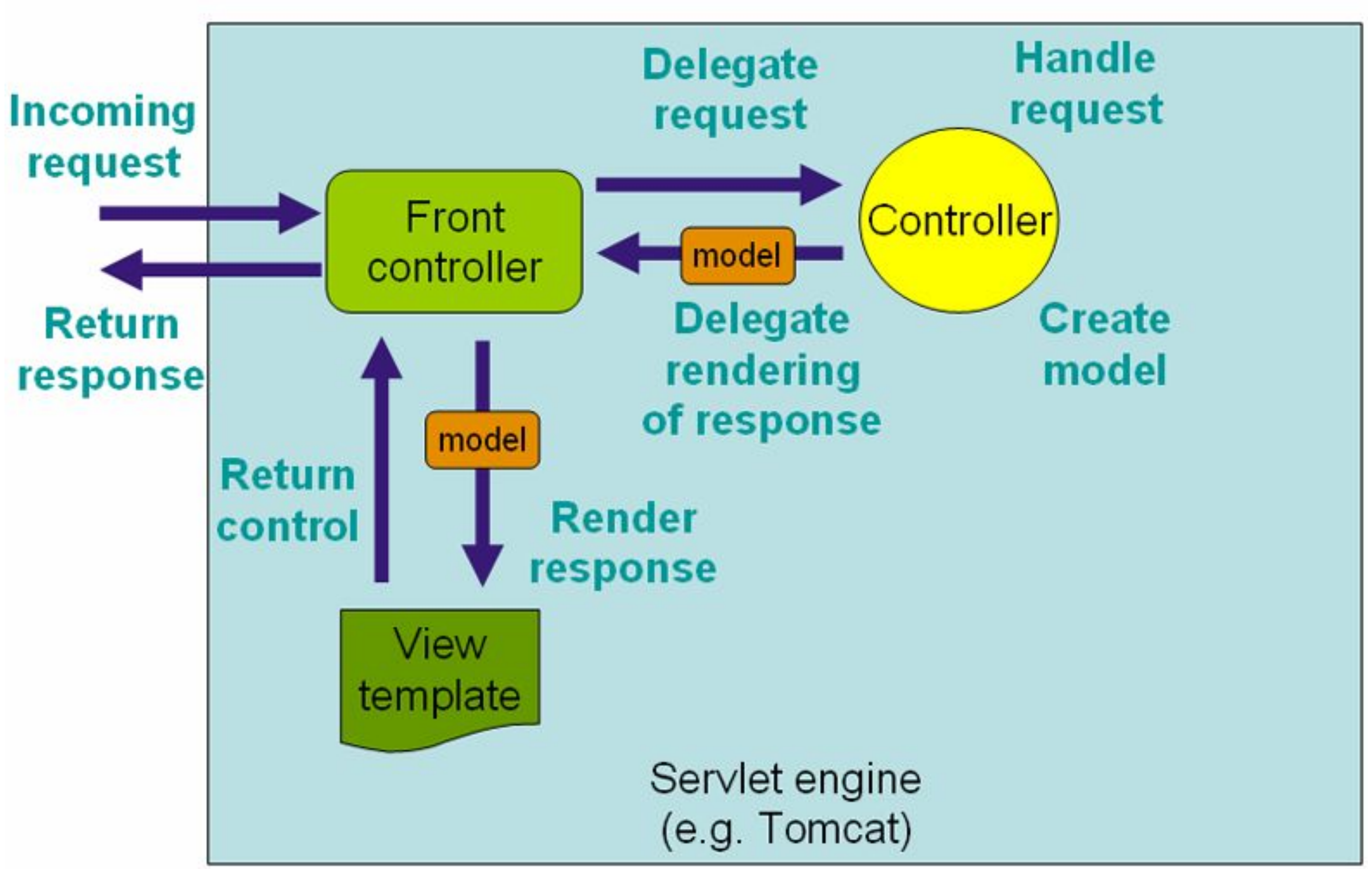
Model part. Contains a description of the main entities of the application.

“Server” module handle part. Consists of configuration class that is used to start REST-application (this is where “void main” is placed) and REST-controllers.

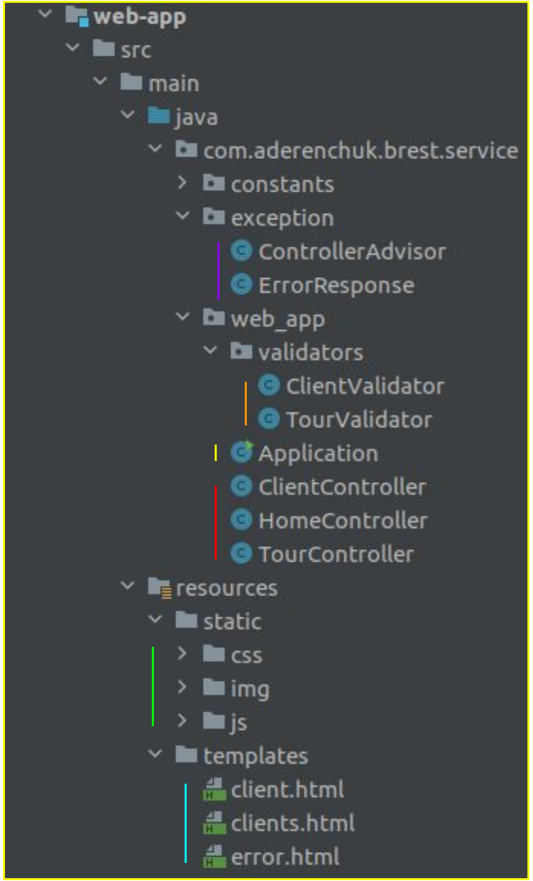
Service part. This module is made for feng shui application. Those modules are capable of coordinating web-app/rest-service modules and rest-app controllers/DAO modules interaction.

“Client” - web part. This module is responsible for deploying operations results through user-friendly UI. It refers directly to rest-service module to perform any kind of transactions, as it lacks business logics itself. This module contains configuration class.

Project architecture

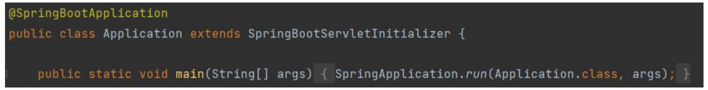


WEB-APP



```
web-app
  src
    main
      java
        com.aderenchuk.brest.service
          constants
          exception
            ControllerAdvisor
            ErrorResponse
          web_app
            validators
              ClientValidator
              TourValidator
            Application
            ClientController
            HomeController
            TourController
      resources
        static
          css
          img
          js
        templates
          client.html
          clients.html
          error.html
```

- Classes with errors. Generate error throws for incorrect actions.
- Validation classes. Check the correctness of the entered data.
- Web-app entry point.

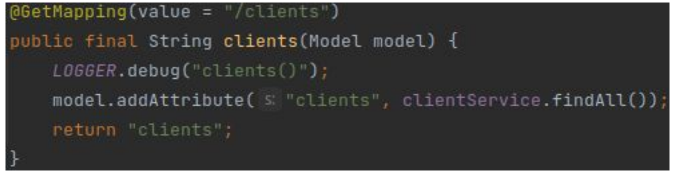


```
@SpringBootApplication
public class Application extends SpringBootServletInitializer {

    public static void main(String[] args) { SpringApplication.run(Application.class, args); }
```

Application class with its main method.

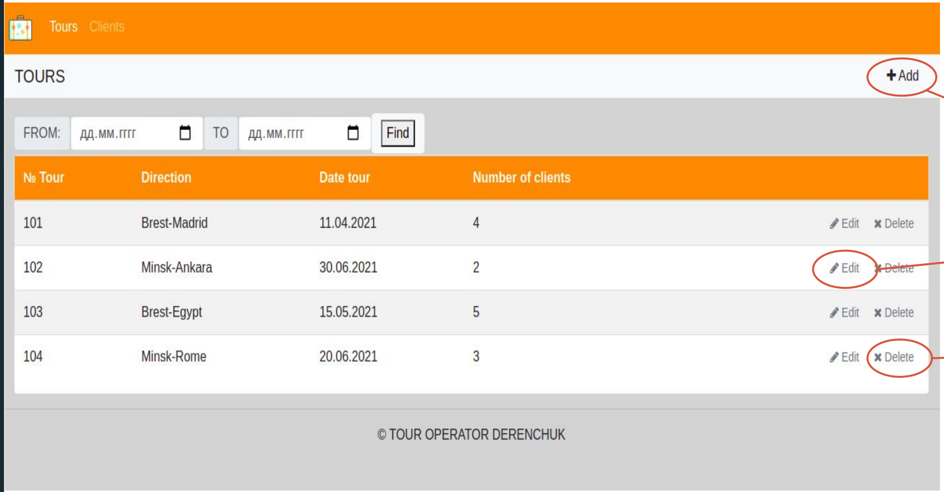
- Front-end controllers. Those are intercepts incoming requests, converts the payload of the request to the internal structure of the data and sends the data to Model for further processing.



```
@GetMapping(value = "/clients")
public final String clients(Model model) {
    LOGGER.debug("clients()");
    model.addAttribute( s: "clients", clientService.findAll());
    return "clients";
}
```

- Front-end resources. Images, css and js files.
- HTML templates, used to visualize received info. Data is taken from models and inserted directly into HTML components by Thymeleaf.

WEB-APP



Tours Clients

TOURS

+ Add

FROM: дд.мм.гггг TO дд.мм.гггг Find

№ Tour	Direction	Date tour	Number of clients	
101	Brest-Madrid	11.04.2021	4	Edit Delete
102	Minsk-Ankara	30.06.2021	2	Edit Delete
103	Brest-Egypt	15.05.2021	5	Edit Delete
104	Minsk-Rome	20.06.2021	3	Edit Delete

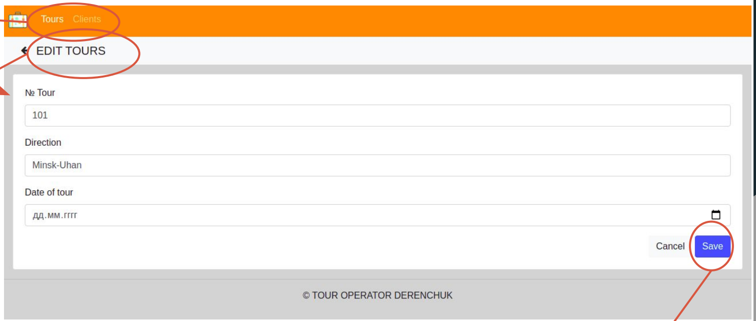
© TOUR OPERATOR DERENCHUK

Navigation bar

Current form unique name

edit form

delete form



Tours Clients

EDIT TOURS

№ Tour: 101

Direction: Minsk-Uhan

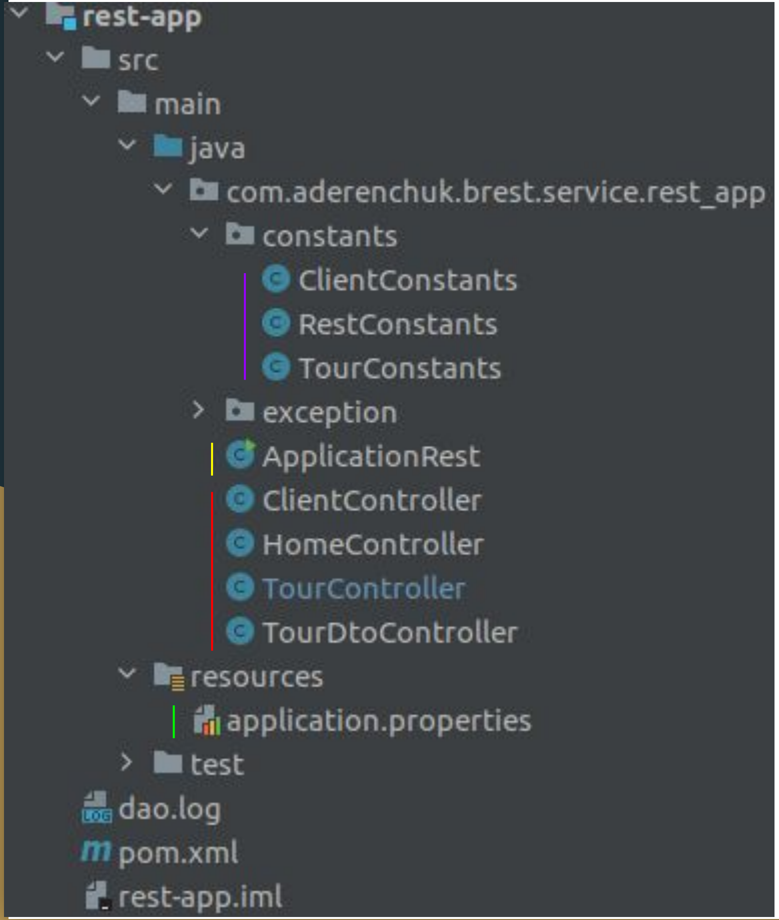
Date of tour: дд.мм.гггг

Cancel Save

© TOUR OPERATOR DERENCHUK

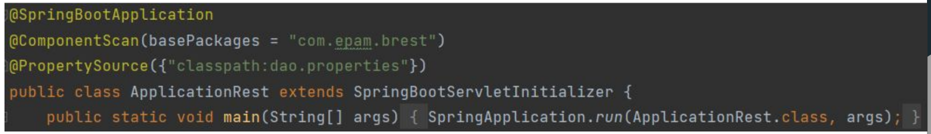
Submit form and send new/updated data

REST-APP



```
rest-app
  src
    main
      java
        com.aderenchuk.brest.service.rest_app
          constants
            ClientConstants
            RestConstants
            TourConstants
          exception
          ApplicationRest
          ClientController
          HomeController
          TourController
          TourDtoController
      resources
        application.properties
    test
  dao.log
  pom.xml
  rest-app.iml
```

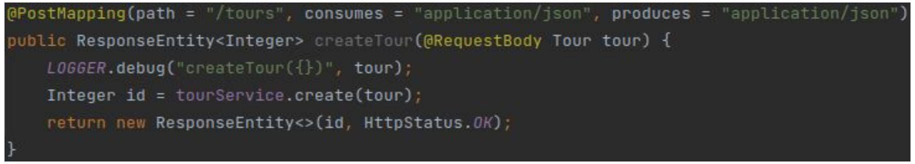
- Constants. Some data is packed in line files for better code readability.
- Rest-app entry point.



```java
@SpringBootApplication
@ComponentScan(basePackages = "com.epam.brest")
@PropertySource({"classpath:dao.properties"})
public class ApplicationRest extends SpringBootServletInitializer {
    public static void main(String[] args) { SpringApplication.run(ApplicationRest.class, args); }
```

ApplicationRest class with its main method.

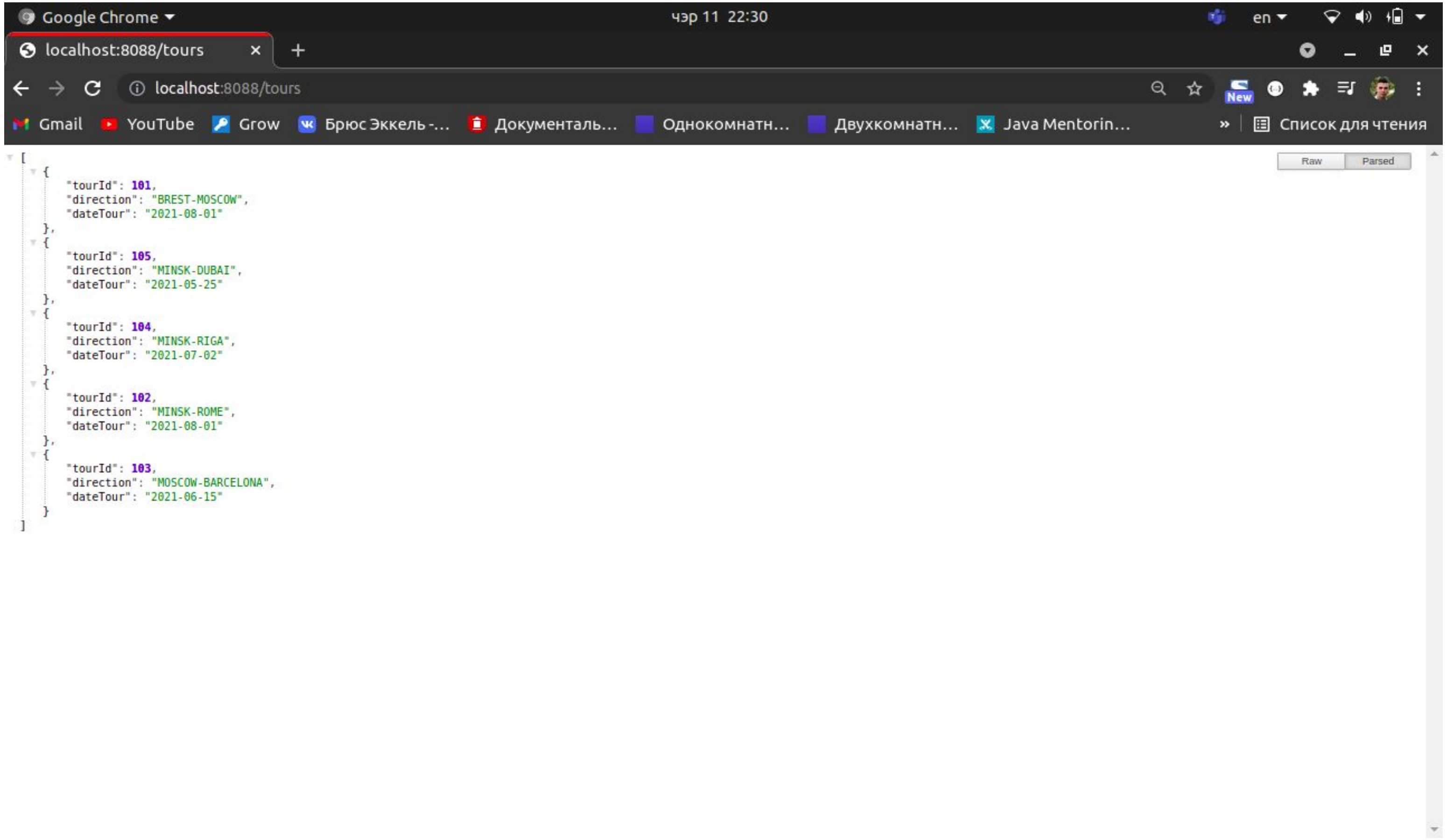
- Back-end controllers. Just like web-app ones, those are used to define which services to trigger to get/send data. Response entities are deployed on defined mapping.



```java
@PostMapping(path = "/tours", consumes = "application/json", produces = "application/json")
public ResponseEntity<Integer> createTour(@RequestBody Tour tour) {
    LOGGER.debug("createTour({})", tour);
    Integer id = tourService.create(tour);
    return new ResponseEntity<>(id, HttpStatus.OK);
}
```

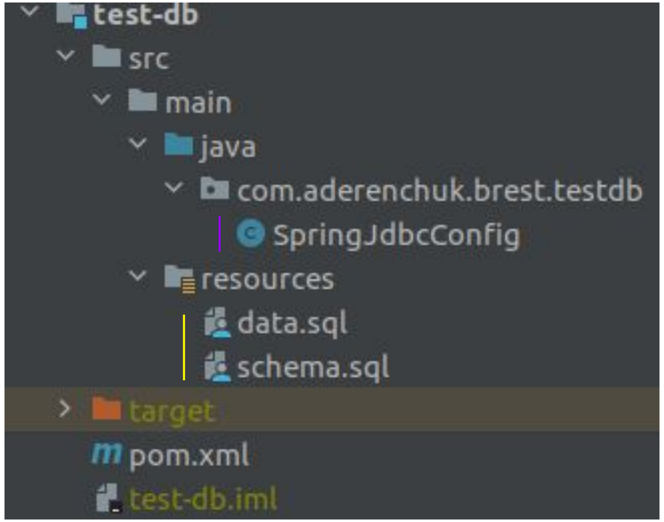
- Property file with REST-server configuration (server port).

REST-APP



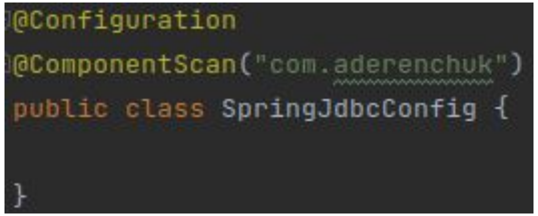
```
[
  {
    "tourId": 101,
    "direction": "BREST-MOSCOW",
    "dateTour": "2021-08-01"
  },
  {
    "tourId": 105,
    "direction": "MINSK-DUBAI",
    "dateTour": "2021-05-25"
  },
  {
    "tourId": 104,
    "direction": "MINSK-RIGA",
    "dateTour": "2021-07-02"
  },
  {
    "tourId": 102,
    "direction": "MINSK-ROME",
    "dateTour": "2021-08-01"
  },
  {
    "tourId": 103,
    "direction": "MOSCOW-BARCELONA",
    "dateTour": "2021-06-15"
  }
]
```

DATABASE



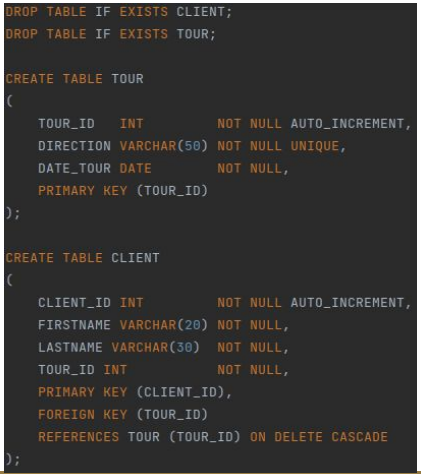
```
test-db
  src
    main
      java
        com.aderenchuk.brest.testdb
          SpringJdbcConfig
      resources
        data.sql
        schema.sql
  target
  pom.xml
  test-db.iml
```

- Database configuration file. Sets up all needed dependencies, points scripts, creates beans.



```java
@Configuration
@ComponentScan("com.aderenchuk")
public class SpringJdbcConfig {

}
```

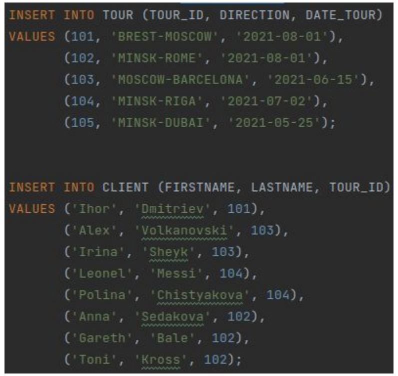


```sql
DROP TABLE IF EXISTS CLIENT;
DROP TABLE IF EXISTS TOUR;

CREATE TABLE TOUR
(
    TOUR_ID   INT         NOT NULL AUTO_INCREMENT,
    DIRECTION VARCHAR(50) NOT NULL UNIQUE,
    DATE_TOUR DATE        NOT NULL,
    PRIMARY KEY (TOUR_ID)
);

CREATE TABLE CLIENT
(
    CLIENT_ID INT         NOT NULL AUTO_INCREMENT,
    FIRSTNAME VARCHAR(20) NOT NULL,
    LASTNAME VARCHAR(30)  NOT NULL,
    TOUR_ID INT           NOT NULL,
    PRIMARY KEY (CLIENT_ID),
    FOREIGN KEY (TOUR_ID)
    REFERENCES TOUR (TOUR_ID) ON DELETE CASCADE
);
```

- SQL files.



```sql
INSERT INTO TOUR (TOUR_ID, DIRECTION, DATE_TOUR)
VALUES (101, 'BREST-MOSCOW', '2021-08-01'),
       (102, 'MINSK-ROME', '2021-08-01'),
       (103, 'MOSCOW-BARCELONA', '2021-06-15'),
       (104, 'MINSK-RIGA', '2021-07-02'),
       (105, 'MINSK-DUBAI', '2021-05-25');


INSERT INTO CLIENT (FIRSTNAME, LASTNAME, TOUR_ID)
VALUES ('Ihor', 'Dmitriev', 101),
       ('Alex', 'Volkanovski', 103),
       ('Irina', 'Sheyk', 103),
       ('Leonel', 'Messi', 104),
       ('Polina', 'Chistyakova', 104),
       ('Anna', 'Sedakova', 102),
       ('Gareth', 'Bale', 102),
       ('Toni', 'Kross', 102);
```