THE SPRING BOOT REST API APPLICATION
“introtelecom”

- provides an introduction to mobile phone service management.

- source code and more details on: https://github.com/petroneris/introtelecom

- mobile phone services nowadays require very complex software solutions.

- this demo application simplifies many details related to mobile services, to show and explain the mobile service basics - how to:

  - register the phones, grant a monthly package frame to each phone

  - register customers (owners of physical phones) and users (owners of software accounts).

  - provide add-ons (additional services during the month) to customers

  - create a record of mobile services (Service Detail Record -SDR, their type, duration and amount)

  - provide the user with details of an active service (what amount of service is left until the end of the month)

  - generate monthly bill facts for the previous month

- **Basic types of mobile services** used in this application:
    - calls (**cls**),
    - short message service (**sms**),
    - internet (**int**),
    - applications and social media (**asm**),
    - international calls (**icl**)
    - roaming (**rmg**)

- two **payment models** here:
    - prepaid – payment comes first and then the services become available
    - postpaid - payment is made after the delivered service, at the beginning of the following month

LIST OF TECHNOLOGIES

- **Spring Boot** – packaging and deployment for REST (REpresentational State Transfer) API
- **Spring Data JPA** – ORM and transaction-driven data handling
- **Hibernate ORM** – used together with Spring Data JPA
- **Spring Security framework** – for logging and access activities
- **Spring Boot Validation** (with **Hibernate Validator**) – for input and database data validation
- **PostgreSQL** – RDBMS data layer
- **Swagger** – UI for REST API and documentation
- **JWT** – JSON Web Token for authentication
- **MapStruct** – to map between different object models (e.g. entities and DTOs)
- **Lombok** – java library that eliminates boilerplate code
- **Slf4j-with-Logback** library for logging

A standard Java servlet container (**Tomcat**) is used as the server platform.
Maven is used as the build system.
The code is **Java 8** – compliant.

DESIGN CONSIDERATIONS

This is a classic REST API application with a Swagger client. It consists of:

- PostgreSQL database named "**intro_telecom**".
- Spring Boot application "**introtelecom**", designed as a Maven Project
- Swagger UI; receives user requests and forwards them to the Spring Boot application for further processing; also displays the REST response in JSON format.

APPLICATION FUNCTIONALITY AND USER ROLES

After running the project, open a browser and enter the following URL:

http://localhost:8080/swagger-ui/index.html

The entrance: Access Controller
- the login service grants a JWT token to each **valid User** - with a valid username and password; with JWT token they can access other resources in the application.

JWT - JSON Web Token

This application receives user requests through the Swagger UI. Requests are then sent to several REST controllers that forward data to services for further processing.

The application interacts with users - **User**. A User has two distinct roles:

- ***ADMIN***, whose role is to control the entire application,
- ***CUSTOMER***, who can get information related to his phone number and mobile services in general.

- Access Controller – permition to all
- **ADMIN role User** can send requests to almost all REST controllers except the Client Controller
- **CUSTOMER role User** can only send requests to the Client Controller

user

phone_number
username role_id = 1 . . .

(ADMIN role User)

user

phone_number
username role_id = 2 . . .

(CUSTOMER role User)

role

role_id	role_type
1	ADMIN
2	CUSTOMER

As mentioned above, this demo app simplifies many things to show and explain the basics of mobile services. So, for the sake of simplicity (1/2):

- there are only six package plans (packages); in reality, every mobile provider offers a huge number of packages
- there are two zones for international calls and roaming; in reality there are at least three zones
- for roaming services there is no price for incoming calls
- roaming and international calls are not included in prepaid packages
- the customer can demand add-on services during the current month, which are valid until the end of the month
- there are only six add-on services that correspond to basic types of mobile services used in this application
- customers can only demand add-on service types that are included in the monthly package frame
- customers cannot demand another add-on service of the same type during the current month
- the packages and add-ons have the same prices for international calls and roaming

... for simplicity (2/2):

- for both types of payments (prepaid and postpaid), package frames start at the very beginning of the month; package frames and add-on frames expire at the very end of the month; so, no leftovers are carried over to the next month
- package frames start automatically at the very beginning of the month; currently, package frames cannot be activate later in the month
- the app does not handle how customers demand the specific add-on service; for demo purposes, ADMIN role user grants add-on frames to CUSTOMER role users in an arbitrary way
- the app does not handle how customers pay for services
- additional services for the mobile phone (call identification, call waiting, call forwarding, phone number display ban, MMS (Multimedia Messaging Service), voicemail...) are not included
- additional internet services (hosting, wi-fi zone, web mail...) are not included
- service recording (SDR) is handled by the ADMIN role user via input data
- and other things ... ☺