Hospital Web App-admin version

Author: Jacek Kaczmarek

Email: kaczmarek.jacek10@gmail.com

This is web application for admin user. This application allows to change everything in project, all database. I created methods to add, delete, update, list objects. Below i will describe all project.

I will create also application for patient and doctor in the future.

So there will be permission to access data e.g each patient might see only own visits.

Bu this is only app for admin just to have control over data.

Database

I created database in mySQL. Name of the db is `web_hospital_tracker`. My database contains 3 tables.

Table doctor

Field Types

#	Field	Schema	Table	Type
1	id	web_hospital_tracker	doctor	INT
2	first_name	web_hospital_tracker	doctor	VARCHAR
3	last_name	web_hospital_tracker	doctor	VARCHAR
4	specialty	web_hospital_tracker	doctor	VARCHAR
5	email	web_hospital_tracker	doctor	VARCHAR

Table doctor contains some basic information about doctor. Each doctor has own id number which identifies doctor. Doctor also has first name, last name, specialty and email.

Below there is some example data of table doctor.

id	first_name	last_name	specialty	email
1	Iza	Kaczmarek	dentist	iza@gmail.com
6	Zuzia	Relewicz	dentist	zuzia@gmail.com
7	Weronika	Sobaczak	orthopaedist	weronika@gmail.com
9	Piotr	Kaczmarek	orthopaedist	puiotras@gmail.com
10	Pawel	Wroclawska	psychiatrist	pawel@gmail.com
13	Wieslaw	Kaczmarek	orthopaedist	wieslaw@gmail.com
NULL	NULL	NULL	NULL	NULL

Id numbers are not in order because i was deleting doctors and adding them multiple times and in this case e.g if you want to add a new doctor then doctor id will be 14. So that is why after doctor with id 1 there is doctor with id 6, i simply deleted all doctors beetween.

Table Patient

Field Types

#	Field	Schema	Table
1	id	web_hospital_tracker	patient
2	first_name	web_hospital_tracker	patient
3	last_name	web_hospital_tracker	patient
4	email	web_hospital_tracker	patient

Table patient contains some basic information about patient. Each patient has own id number which identifies patient. Patient has also first name, last name and email.

Below there is some example data of table patient.

	id	first_name	last_name	email
▶	9	Zuzia	Olszewska	zuzia@gmail.com
	10	Lil	Kaczmarek	lil@gmail.com
	17	Weronika	Mazurek	weronika@gmail.com
	22	Bartek	Strugarek	bartek@gmail.com
	23	Artur	Kowalski	artur@gmail.com
	25	Jakub	Kaczmarek	losek@gmail.com
	27	Beata	Rozanek	beata@gmail.com
	28	Iza	Piechowiak	iza@gmail.com
*	NULL	NULL	NULL	NULL

Id numbers are not in order because i was deleting patients and adding them multiple times and in this case e.g if you want to add a new patient then doctor id will be 29. So that is why after doctor with id 10 there is doctor with id 17, i simply deleted all doctors beetween.

Table visit

I created many-to-many mapping beetwen tables doctor and patient.

Table visit contains id number from table doctor and id number from table patient.

On diagram below you can see that dependency.

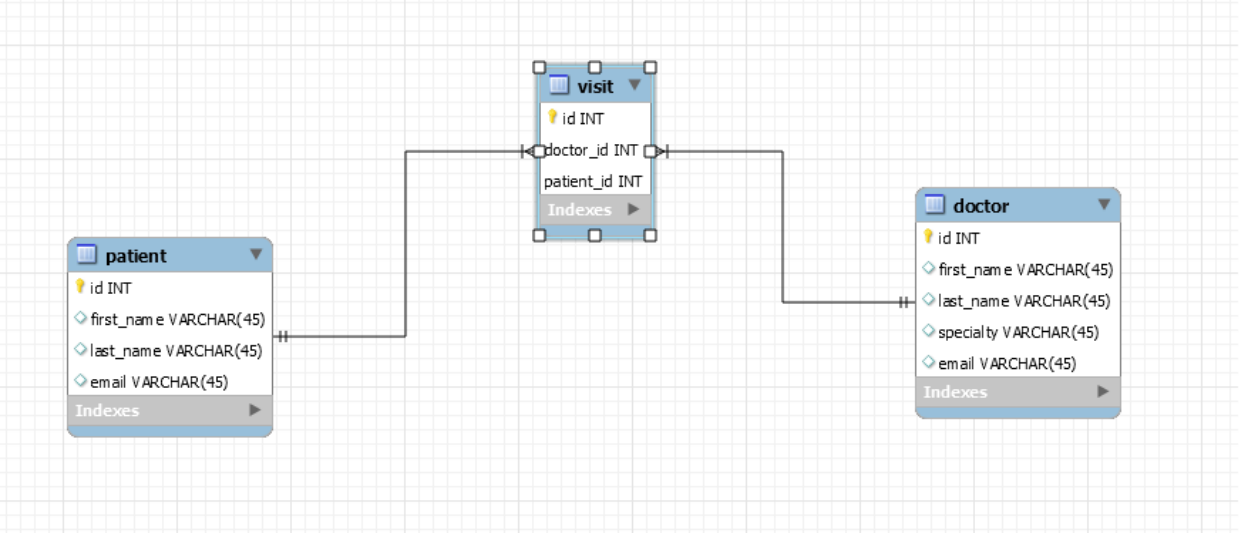


Table visit conatins own id number and id of doctor and patient.

Field Types

#	Field	Schema	Table	Type
1	id	web_hospital_tracker	visit	INT
2	doctor_id	web_hospital_tracker	visit	INT
3	patient_id	web_hospital_tracker	visit	INT

Some example data of table visit.

	id	doctor_id	patient_id
▶	2	1	9
	1	1	10
*	NULL	NULL	NULL

Maven Project

I created maven project to modify database data.

In package com.hospital.springdemo.entity there are connected classes with db tables.

So class Doctor is connected to table doctor.

```
@Entity
@Table(name="doctor")
public class Doctor {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="first_name")
    private String firstName;

    @Column(name="last_name")
    private String lastName;

    @Column(name="specialty")
    private String specialty;

    @Column(name="email")
    private String email;
```

All fields are conntected together.

Table doctor has also many-to many mapping.

```
@ManyToMany(fetch=FetchType.LAZY,
        cascade= {CascadeType.PERSIST, CascadeType.MERGE,
        CascadeType.DETACH,CascadeType.REFRESH})
@JoinTable(
        name="visit",
        joinColumns=@JoinColumn(name="doctor_id"),
        inverseJoinColumns=@JoinColumn(name="patient_id")
        )
@JsonIgnore
private List<Patient> patients;
```

Each doctor has list of patients that he cures.

Class Patient

Class Patient is connected to table patient.

```
@Entity
@Table(name="patient")
public class Patient {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="first_name")
    private String firstName;

    @Column(name="last_name")
    private String lastName;

    @Column(name="email")
    private String email;
```

All fields are conntected together.

Table patient has also many-to many mapping.

```
@ManyToMany(fetch=FetchType.LAZY,
        cascade= {CascadeType.PERSIST, CascadeType.MERGE,
        CascadeType.DETACH,CascadeType.REFRESH})
@JoinTable(
        name="visit",
        joinColumns=@JoinColumn(name="patient_id"),
        inverseJoinColumns=@JoinColumn(name="doctor_id")
        )
@JsonIgnore
private List<Doctor> doctors;
```

Each patient has list of doctors, so as in real life we can treat with many doctors.

Class Visit

```java
package com.hospital.springdemo.entity;

import javax.persistence.Column;

@Entity
@Table(name="visit")
public class Visit {


    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;


    @Column(name="doctor_id")
    private int docId;

    @Column(name="patient_id")
    private int patId;
```

This class conntects doctors and patients.

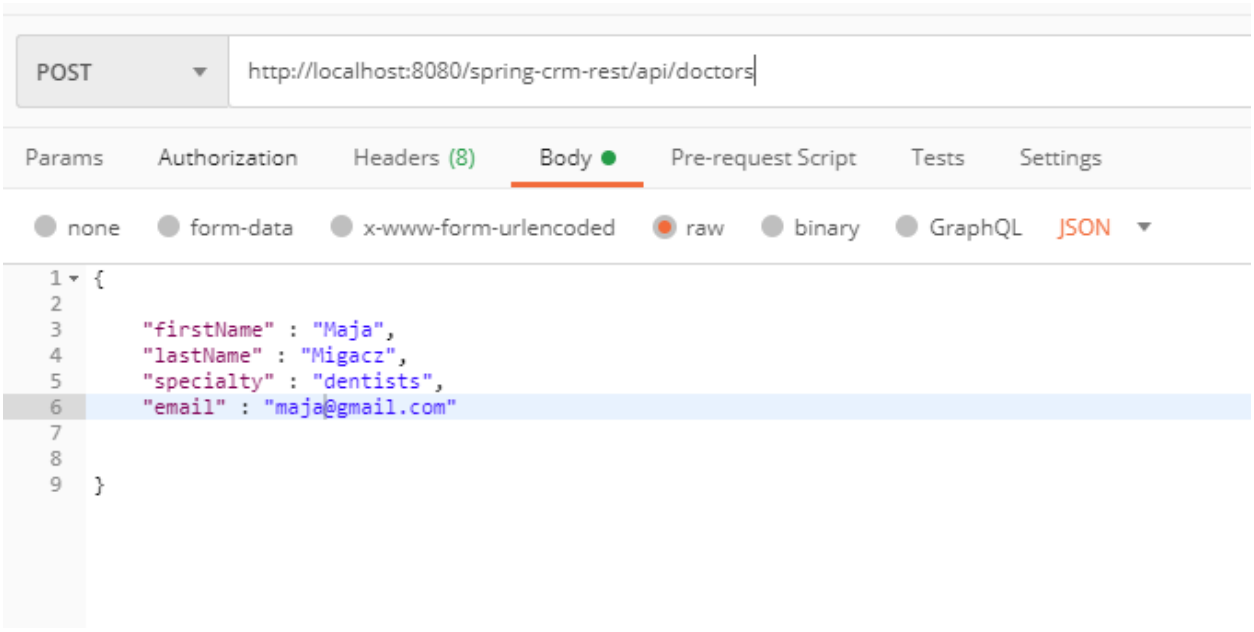
All visits are stored in database.

Each visit has own id number, id of doctor and id of patient.

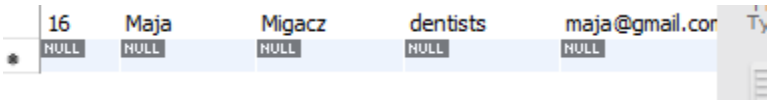
Testing REST API in postman

I will test CRUD on Doctor class.

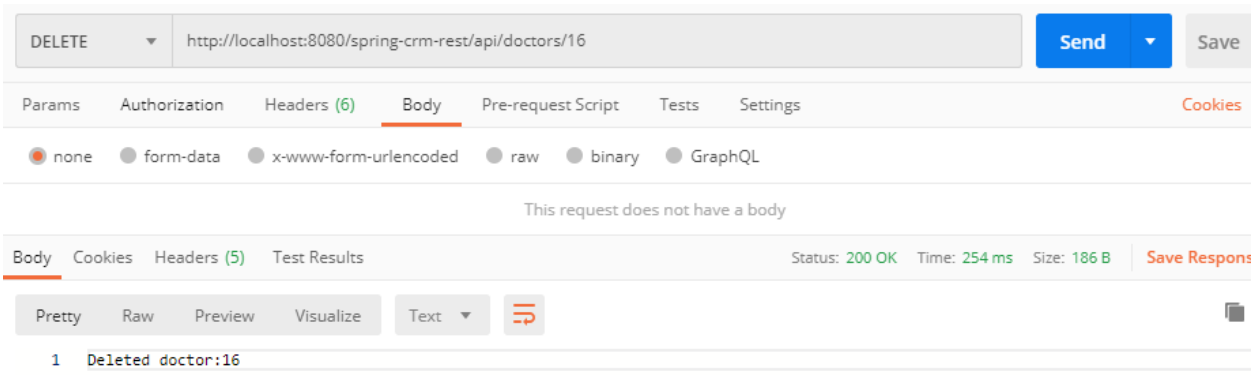
Adding doctor:



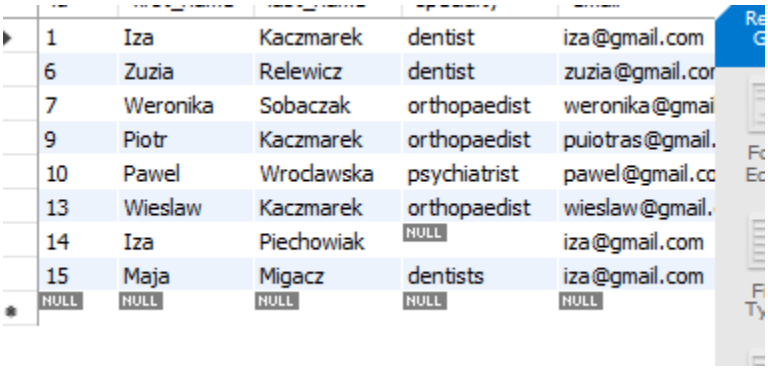
Result in database:



Deleting doctor:

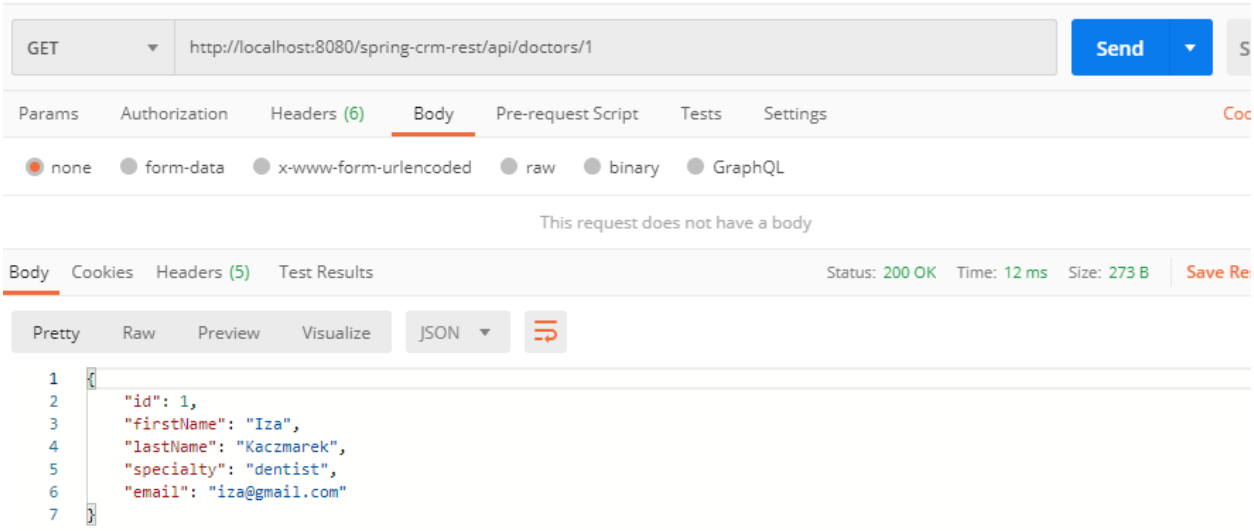


Result:

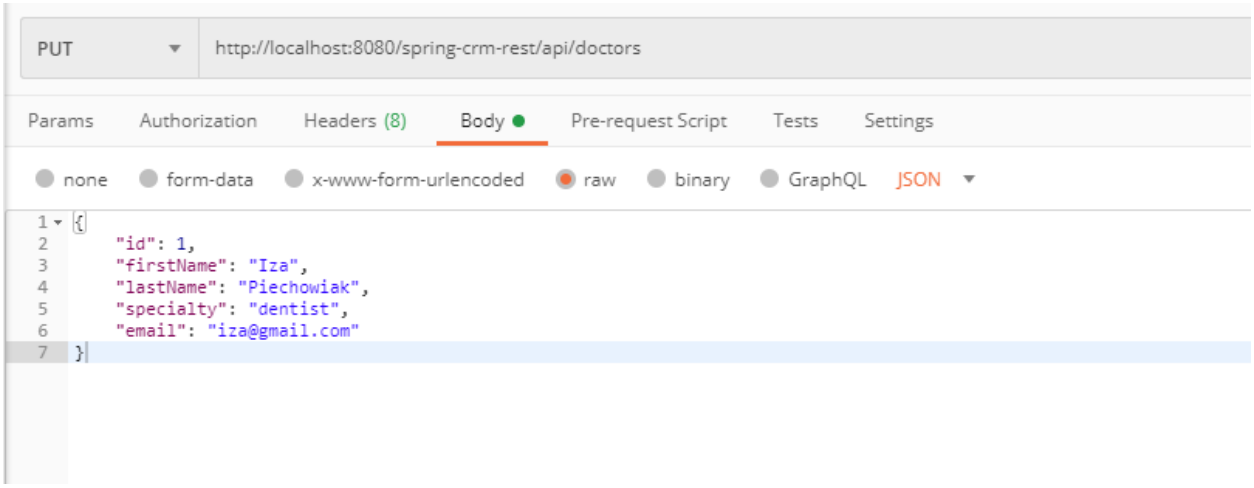


1	Iza	Kaczmarek	dentist	iza@gmail.com
6	Zuzia	Relewicz	dentist	zuzia@gmail.cor
7	Weronika	Sobaczak	orthopaedist	weronika@gmai
9	Piotr	Kaczmarek	orthopaedist	puiotras@gmail.
10	Pawel	Wroclawska	psychiatrist	pawel@gmail.cc
13	Wieslaw	Kaczmarek	orthopaedist	wieslaw@gmail.
14	Iza	Piechowiak	NULL	iza@gmail.com
15	Maja	Migacz	dentists	iza@gmail.com
NULL	NULL	NULL	NULL	NULL

Reading doctor:

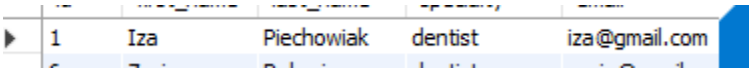


Updating doctor:



lastName was changed from value Kaczmarek to Piechowiak

Result:



So there are also same methods for Patients and Visits

Methods to update patients are in package:

Com.hospital.springdemo.rest

In file PatientRestController

Methods to update visits are in package:

Com.hospital.springdemo.rest

In file VisitRestController