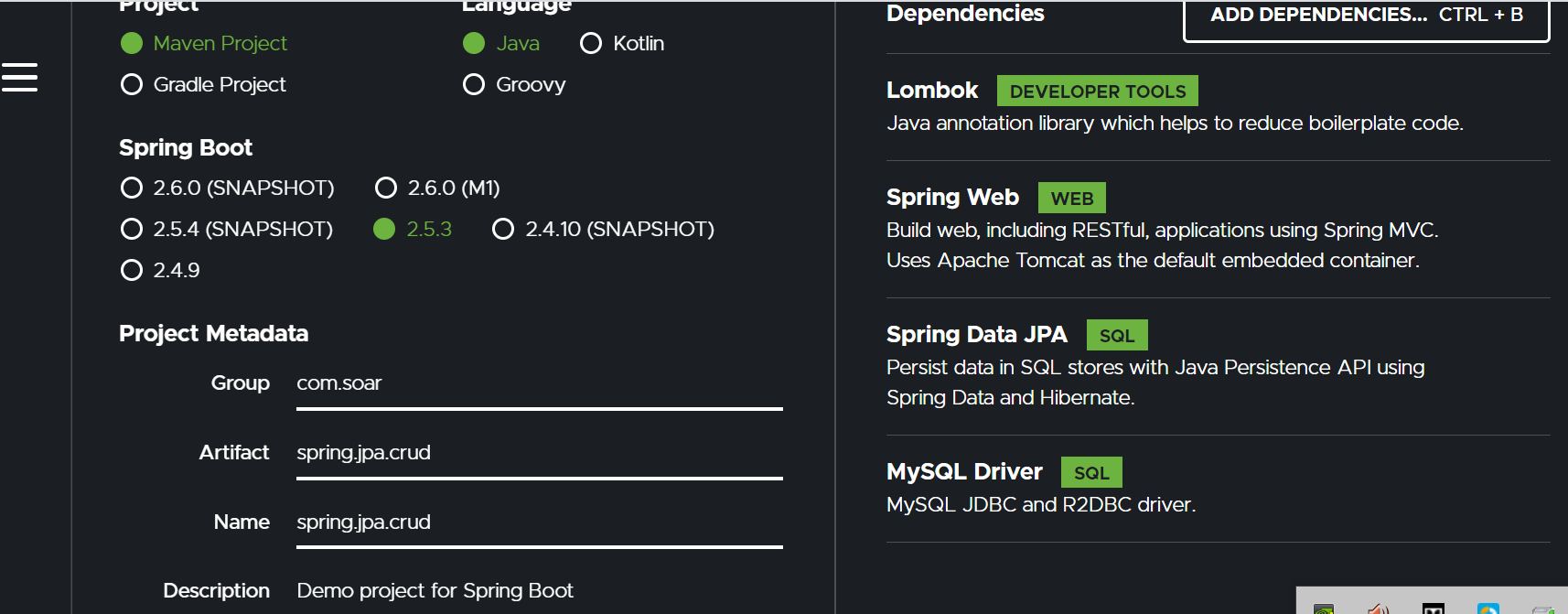
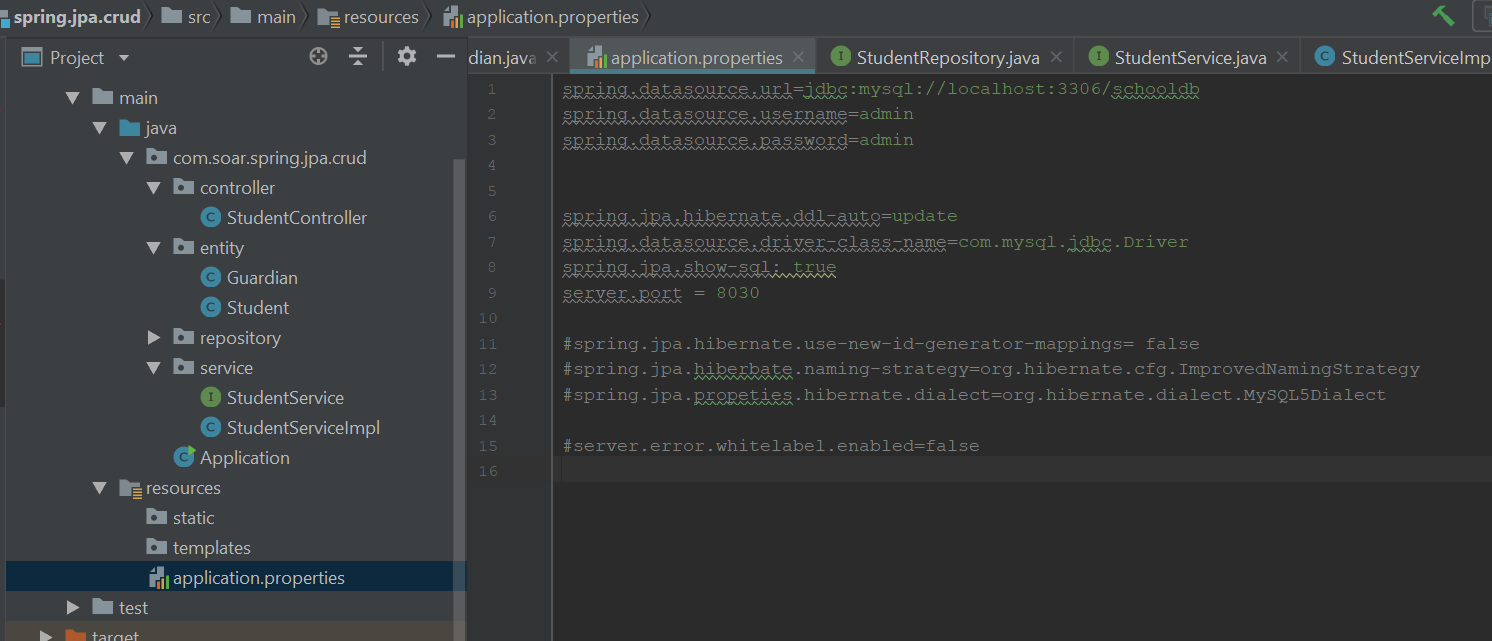
**JPA EntityManager example using Spring Boot:**

1. Create Spring Boot project with Spring Initializr (<https://start.spring.io/>) :

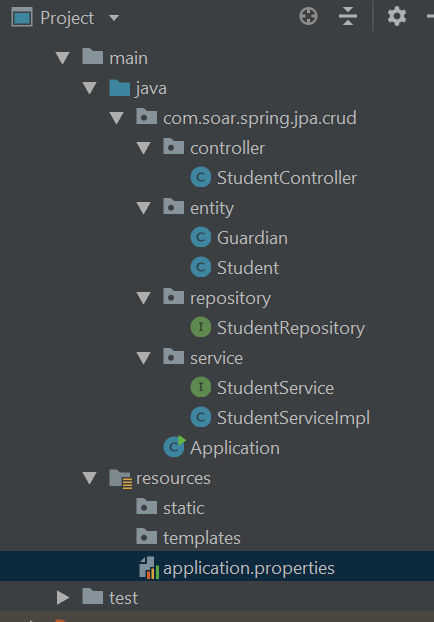


1. Download and open it in IntelliJ/Eclipse.
2. Add content to /resources/application.properties:



spring.datasource.url=jdbc:mysql://localhost:3306/schooldb  
spring.datasource.username=admin  
spring.datasource.password=admin  
  
  
spring.jpa.hibernate.ddl-auto=update  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
spring.jpa.show-sql: true  
server.port = 8030  
  
#spring.jpa.hibernate.use-new-id-generator-mappings= false  
#spring.jpa.hiberbate.naming-strategy=org.hibernate.cfg.ImprovedNamingStrategy  
#spring.jpa.propeties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect  
  
#server.error.whitelabel.enabled=false

1. Create packages and classes for the project



4.1 Package “entity”:

Student class:

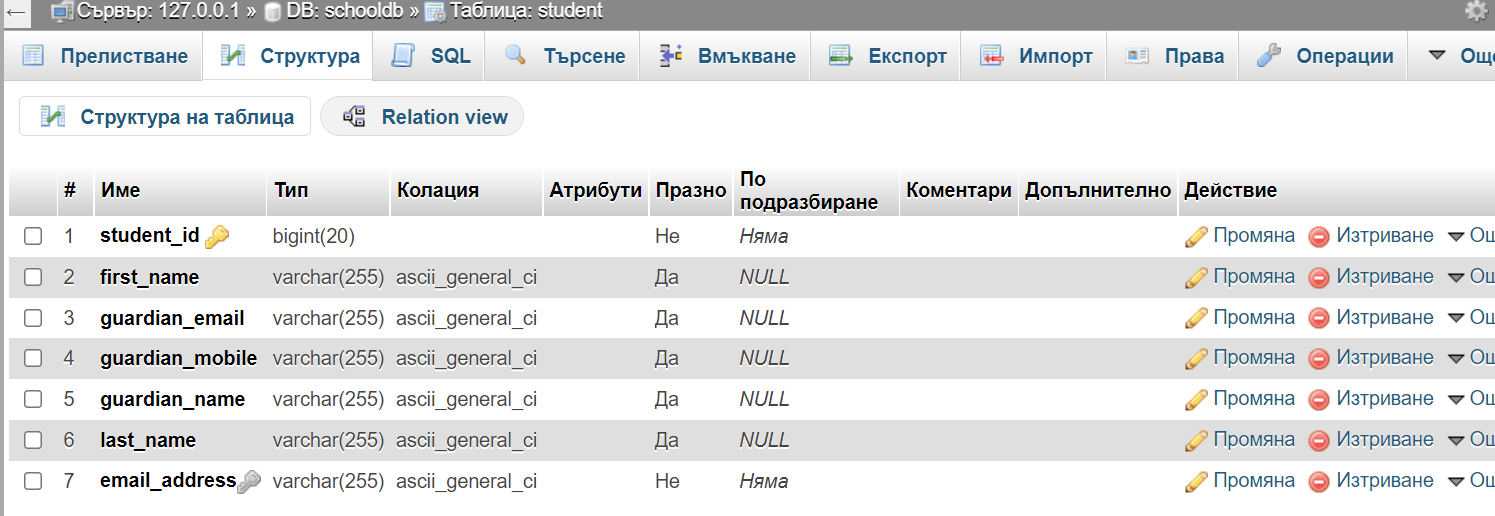
package com.soar.spring.jpa.crud.entity;  
  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
  
@Entity  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder  
@Table(name="student",  
 uniqueConstraints = @UniqueConstraint(  
 name = "emailid\_unique",  
 columnNames = "email\_address"  
 )  
)  
public class Student {  
  
 @Id  
 @SequenceGenerator(  
 name="student\_sequence",  
 sequenceName = "student\_sequence",  
 allocationSize=1  
 )  
 @GeneratedValue(  
 strategy = GenerationType.*SEQUENCE*,  
 generator="student\_sequence"  
 )  
 private Long studentId;  
 private String firstName;  
 private String lastName;  
  
 @Column(  
 name="email\_address",  
 nullable = false  
 )  
 private String emailId;  
  
 @Embedded  
 private Guardian guardian;  
  
}

If we don’t want to create a new table for this class - Guardian, we make it @Embeddable, and in Student entity we reference it through @Embedded annotation:

Guardian class:

package com.soar.spring.jpa.crud.entity;  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.AttributeOverride;  
import javax.persistence.AttributeOverrides;  
import javax.persistence.Column;  
import javax.persistence.Embeddable;  
  
@Embeddable  
@Data  
@Builder  
@AllArgsConstructor  
@NoArgsConstructor  
@AttributeOverrides(  
 {  
 @AttributeOverride(  
 name = "name",  
 column = @Column(name = "guardian\_name")  
 ),  
 @AttributeOverride(  
 name = "email",  
 column = @Column(name = "guardian\_email")  
 ),  
 @AttributeOverride(  
 name = "mobile",  
 column = @Column(name = "guardian\_mobile")  
 )  
 }  
)  
public class Guardian {  
  
 private String name;  
 private String email;  
 private String mobile;  
  
}

If we don’t have a table in or database, after running the spring project, we will have a table “student”:



* 1. Package “repository”, providing access to data:

package com.soar.spring.jpa.crud.repository;  
  
import com.soar.spring.jpa.crud.entity.Student;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface StudentRepository extends JpaRepository<Student, Long> {  
  
}

* 1. Package “service”, providing main service between data model and controller:

package com.soar.spring.jpa.crud.service;  
  
import com.soar.spring.jpa.crud.entity.Student;  
import org.springframework.stereotype.Component;  
  
@Component  
public interface StudentService {  
 public Student saveStudent(Student student);  
  
 public Student deleteStudent(Student student);  
  
 public Student updateStudent(Student student);  
  
 public Student getStudent(Long id);  
  
}

and its implementation:

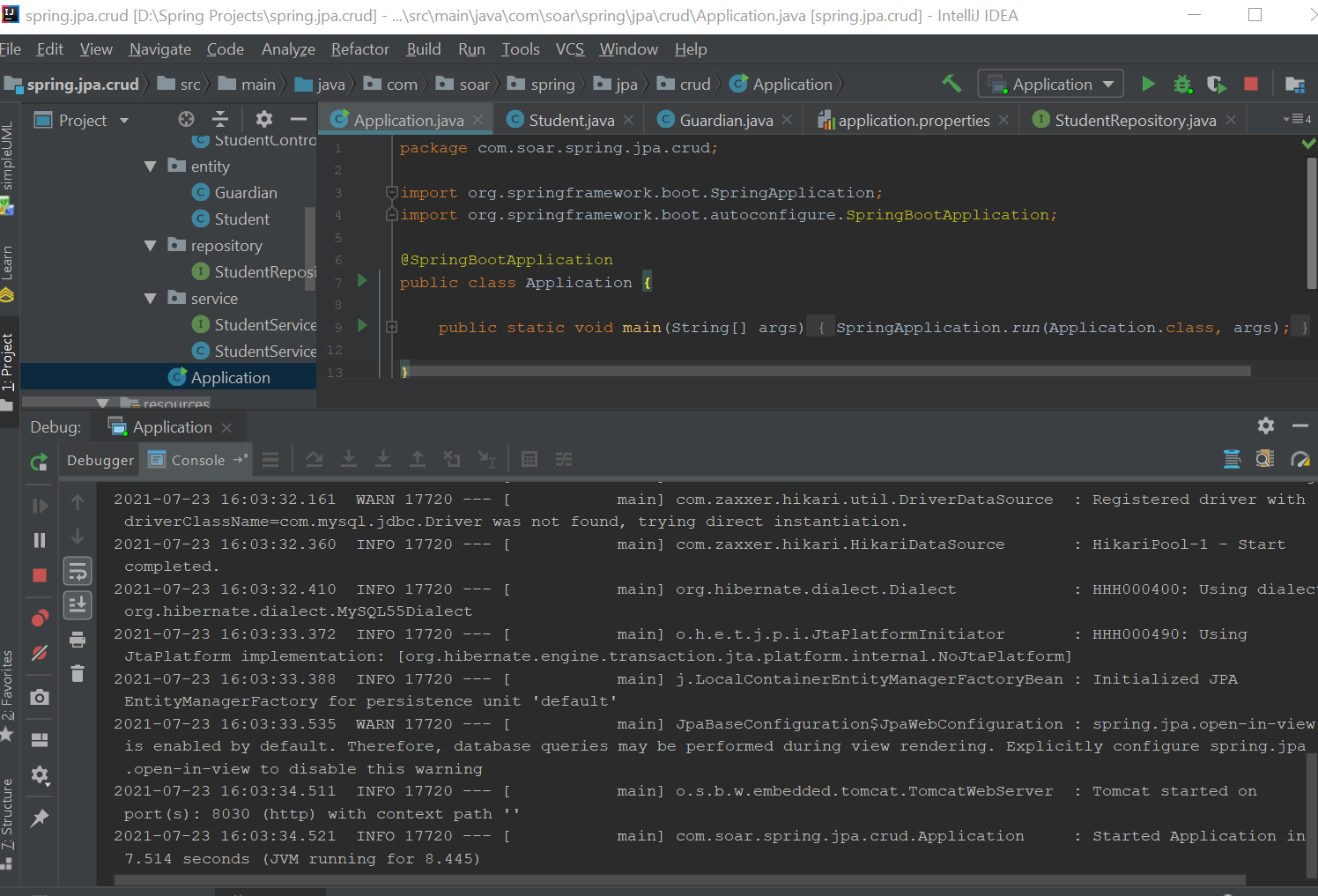
package com.soar.spring.jpa.crud.service;  
  
import com.soar.spring.jpa.crud.entity.Student;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import javax.persistence.EntityManager;  
  
@Service("StudentServiceImpl")  
public class StudentServiceImpl implements StudentService {  
  
 @Autowired  
 private EntityManager entityManager;  
  
 @Override  
 @Transactional  
 public Student saveStudent(Student student) {  
 entityManager.persist(student);  
 return student;  
 }  
  
 @Override  
 @Transactional  
 public Student deleteStudent(Student student) {  
 if(entityManager.contains(student)){  
 entityManager.remove(student);  
 }else{  
 entityManager.remove(entityManager.merge(student));  
 }  
 return student;  
 }  
  
 @Override  
 @Transactional  
 public Student updateStudent(Student student) {  
 entityManager.merge(student);  
 return student;  
 }  
  
 @Override   
 public Student getStudent(Long id) {  
 return entityManager.find(Student.class, id);  
 }  
}

* 1. Package “controller”- the main manager of whole process:

package com.soar.spring.jpa.crud.controller;  
  
import com.soar.spring.jpa.crud.entity.Student;  
import com.soar.spring.jpa.crud.service.StudentService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping(value = "/student")  
 public class StudentController {  
  
 @Autowired  
 private StudentService studentService;  
  
 @RequestMapping(value = "/save", method = RequestMethod.*POST*)  
 @ResponseBody  
 public Student saveStudent(@RequestBody Student student) {  
 return studentService.saveStudent(student);   
 }  
  
 @RequestMapping(value = "/update", method = RequestMethod.*PUT*)  
 @ResponseBody  
 public Student updateStudent(@RequestBody Student student) {  
 return studentService.updateStudent(student);  
 }  
  
 @RequestMapping(value = "/delete", method = RequestMethod.*DELETE*)  
 @ResponseBody  
 public Student deleteStudent(@RequestBody Student student) {  
 return studentService.deleteStudent(student);   
 }  
  
  
 @RequestMapping(value = "/find/{id}", method = RequestMethod.*GET*)  
 @ResponseBody  
 public Student getStudent(@PathVariable Long id) {  
  
 return studentService.getStudent(id);   
 }  
  
  
}

1. Run project with Spring application main class:

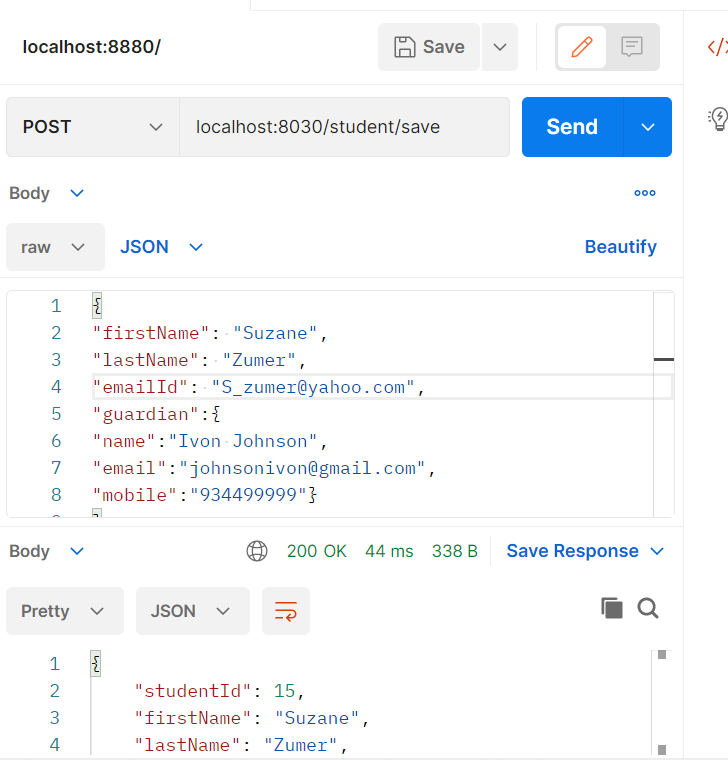
package com.soar.spring.jpa.crud;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Application {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Application.class, args);  
 }  
  
}



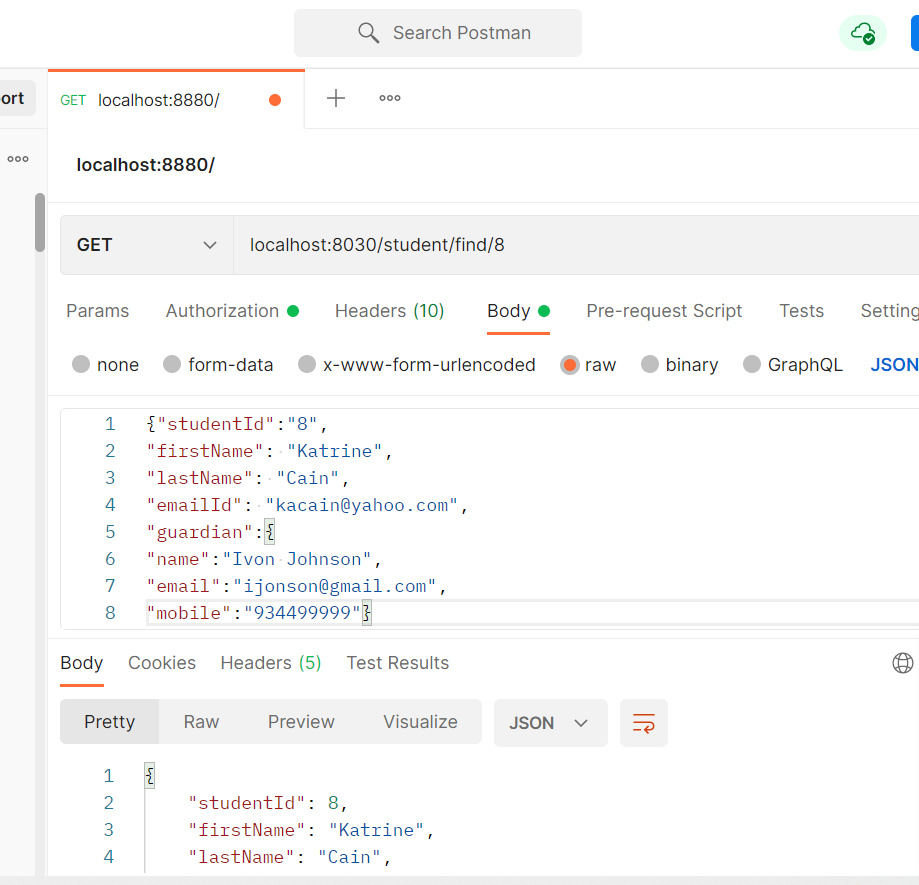
And…

1. Testing with Postman server:

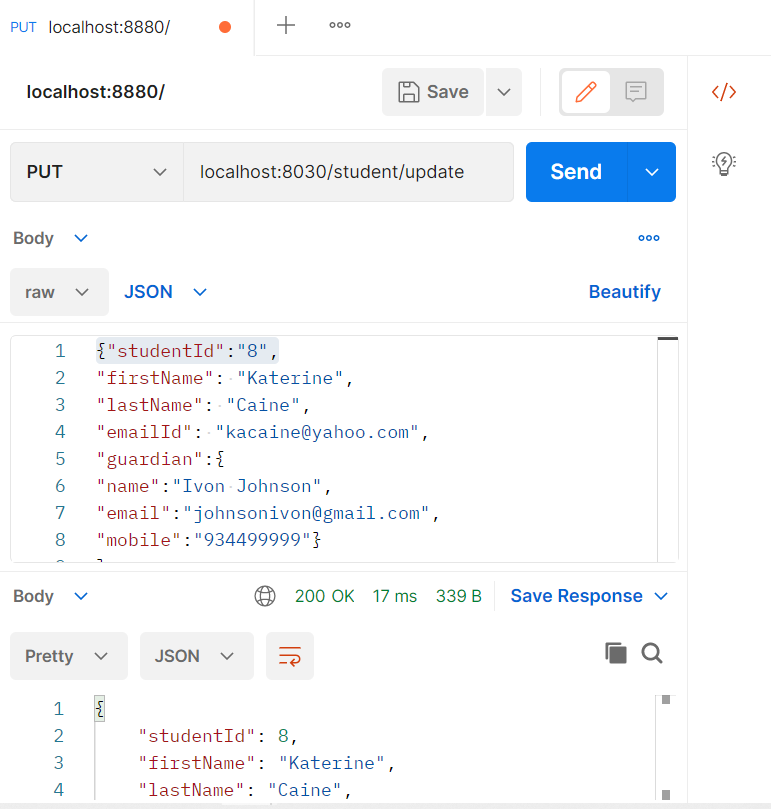
POST:



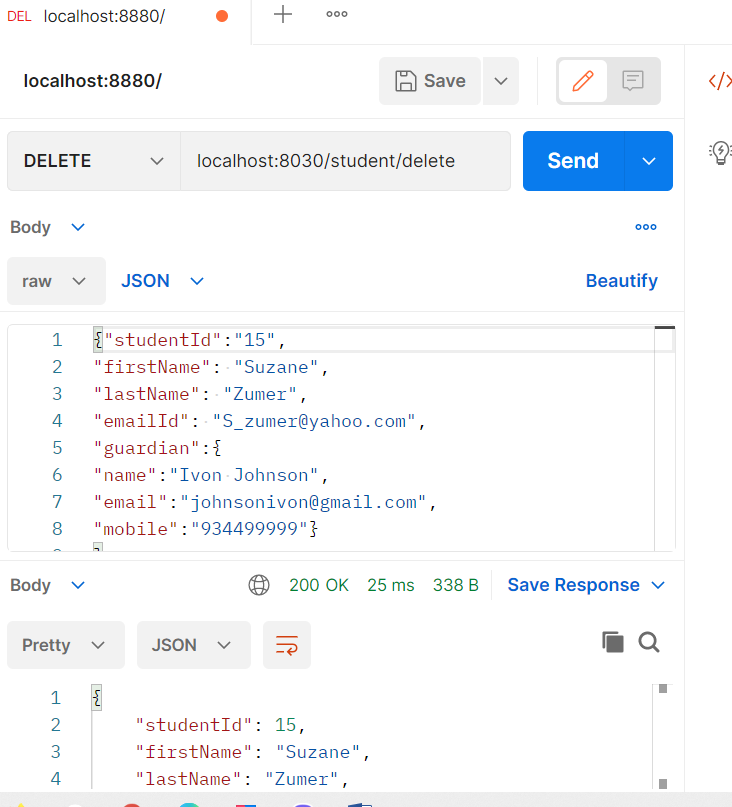
GET:



UPDATE:



DELETE:



The end!

Enjoy codding!