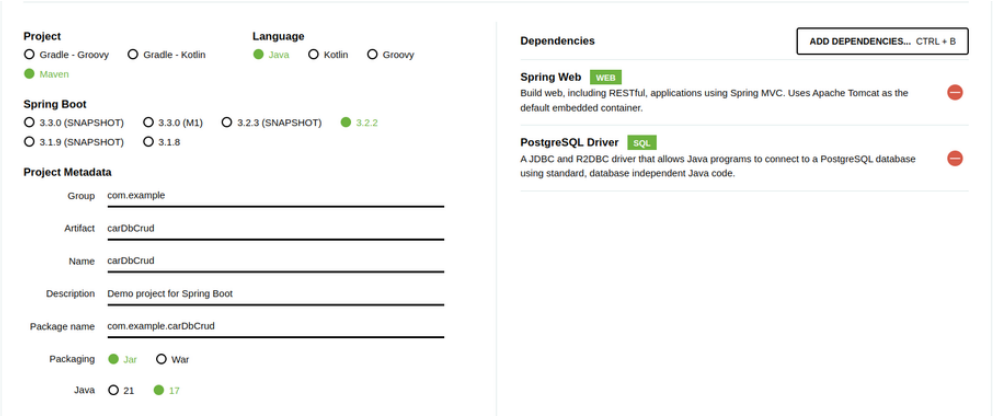
CRUD alkalmazás



- Letöltjük, kizippeljük
- Importáljuk (Open) Intellijbe
- ki lehet próbálni http://localhost:8080/ -on whitepage errort dob ha jó.
- Ahol az application file van, azon a packagen belül létrehozunk egy új packaget
- Controller package/ `ApiControllers`

```java
package car.db.carDbCrud.Controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ApiControllers {
    @GetMapping(value="/")
    public String getPage(){
        return "Welcome" ;
    }
}

```

- Ez kiírja localhoston, hogy Welcome
- felvesszük a jpa-t dependenciának és az sql connectort (ha még nincs ott), és az új Springnél kell a jakarta is

```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>

        <dependency>
            <groupId>jakarta.persistence</groupId>
            <artifactId>jakarta.persistence-api</artifactId>
            <version>3.1.0</version>
        </dependency>
```

- Létrehozzuk a POJO-t, és annotáljuk
- ```java
  package car.db.carDbCrud.Models;

  import jakarta.persistence.*;

  @Entity
  public class Car {
      @Id
      @GeneratedValue(strategy = GenerationType.IDENTITY)
      private Long id;
      @Column
      private String number_plate;
      @Column
      private String manufacturer;
      @Column
      private String model;
      @Column
      private int manufacture_year;

      public Long getId() {
          return id;
      }

      public void setId(Long id) {
          this.id = id;
      }

      public String getNumber_plate() {
          return number_plate;
      }

      public void setNumber_plate(String number_plate) {
          this.number_plate = number_plate;
      }

      public String getManufacturer() {
          return manufacturer;
      }

      public void setManufacturer(String manufacturer) {
          this.manufacturer = manufacturer;
      }

      public String getModel() {
          return model;
      }

      public void setModel(String model) {
          this.model = model;
      }

      public int getManufacture_year() {
          return manufacture_year;
      }

      public void setManufacture_year(int manufacture_year) {
          this.manufacture_year = manufacture_year;
      }
  ```
  ```

```
58
59      @Override
60      public String toString() {
61          return "Car{" +
62                  "id=" + id +
63                  ", number_plate='" + number_plate + '\'' +
64                  ", manufacturer='" + manufacturer + '\'' +
65                  ", model='" + model + '\'' +
66                  ", manufacture_year=" + manufacture_year +
67                  '}';
68      }
69  }
70
```

- Bekonfigoljuk a db kapcsolatot:
- Resources/application.properies
- ```
  1  spring.datasource.url=jdbc:postgresql://localhost:5432/CarDb
  2  spring.datasource.username=attilakixx
  3  spring.datasource.password=admin
  4  spring.jpa.hibernate.ddl-auto=update
  ```
- Hozzuk létre
  - Repo package létrehoz
  - CarRepo interface létrehoz

    ```
    1  package car.db.carDbCrud.Repo;
    2
    3  import car.db.carDbCrud.Models.Car;
    4  import org.springframework.data.jpa.repository.JpaRepository;
    5
    6  public interface CarRepo extends JpaRepository<Car, Long> {
    7  }
    8
    ```
- Ha most futtatjuk, akkor a POJO nevén (car) létrehoz egy új táblát a DB-ben, a megfelelő oszlopokkal.
- Visszamegyünk az ApiControllersbe-be
  - felvesszük autowired annotációval fieldnek a CarRepot
  - létrehozzuk a getCars metódust, ez listával tér vissza, a carRepo.findAll() methodja segít ebben.
  - A GetMappingban megadjuk, hogy az URL-ben hol legyen ez a hívás:
  - ```
    1  package car.db.carDbCrud.Controller;
    2
    3  import car.db.carDbCrud.Models.Car;
    4  import car.db.carDbCrud.Repo.CarRepo;
    5  import org.springframework.beans.factory.annotation.Autowired;
    6  import org.springframework.web.bind.annotation.GetMapping;
    7  import org.springframework.web.bind.annotation.RestController;
    8
    9  import java.util.List;
    10
    11 @RestController
    12 public class ApiControllers {
    13
    14     @Autowired
    15     private CarRepo carRepo;
    ```

```
16
17      @GetMapping(value="/")
18      public String getPage(){
19          return "Welcome" ;
20      }
21
22      @GetMapping(value="/cars")
23      public List<Car> getCars() {
24          return carRepo.findAll();
25      }
26  }
27
```

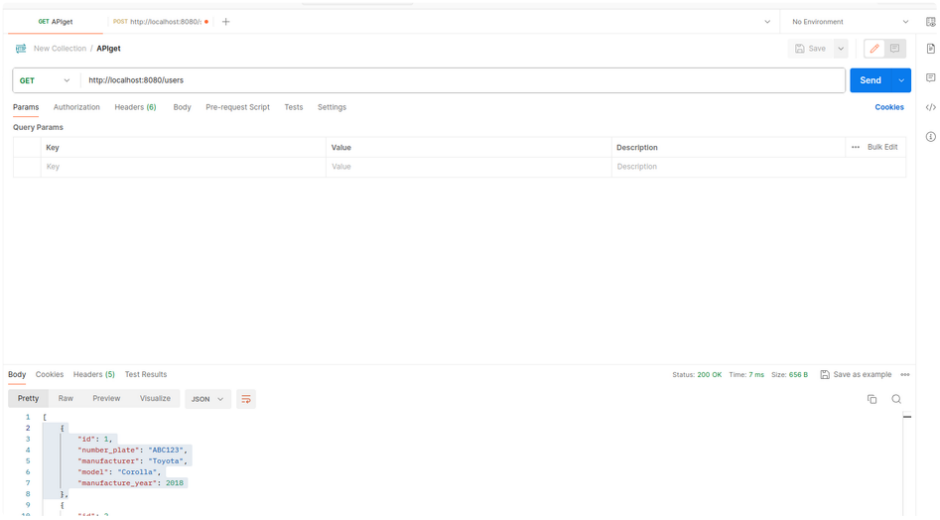
- Így ha felnézünk a http://localhost:8080/users címre, látjuk Json formátumban a DB összes elemét. (A db-t közben feltöltöttem mock datával postgreSqlben:

```
1  INSERT INTO Car (number_plate, manufacturer, model, manufacture_year) VALUES
2  ('ABC123', 'Toyota', 'Corolla', 2018),
3  ('XYZ456', 'Honda', 'Civic', 2019),
4  ('DEF789', 'Ford', 'Mustang', 2020),
5  ('GHI012', 'Chevrolet', 'Camaro', 2021),
6  ('JKL345', 'Tesla', 'Model S', 2022);
```

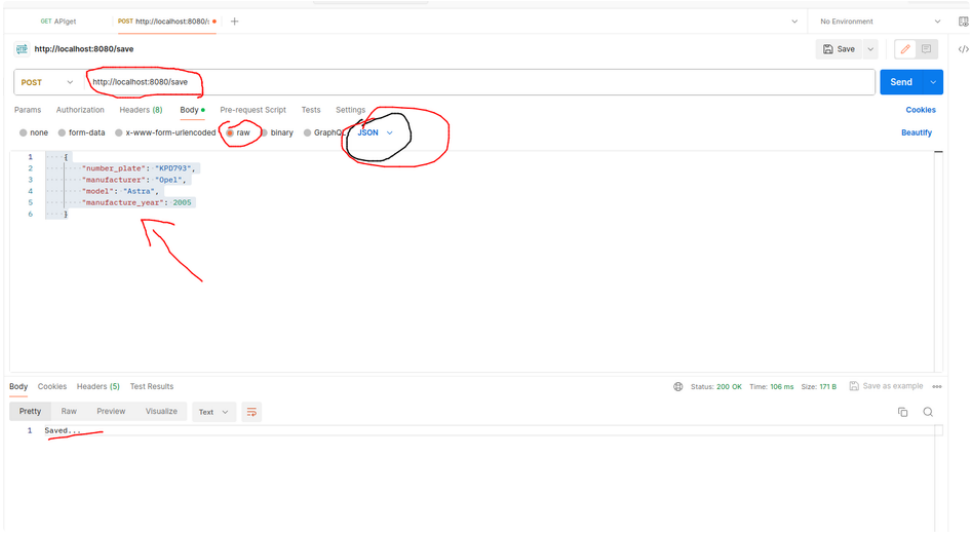
- Most megcsináljuk a save-et még mindig az ApiControllersben:
-

```
1
2      @PostMapping(value="/save")
3      public String saveCar(@RequestBody Car car){
4          carRepo.save(car);
5          return "Saved...";
6      }
```

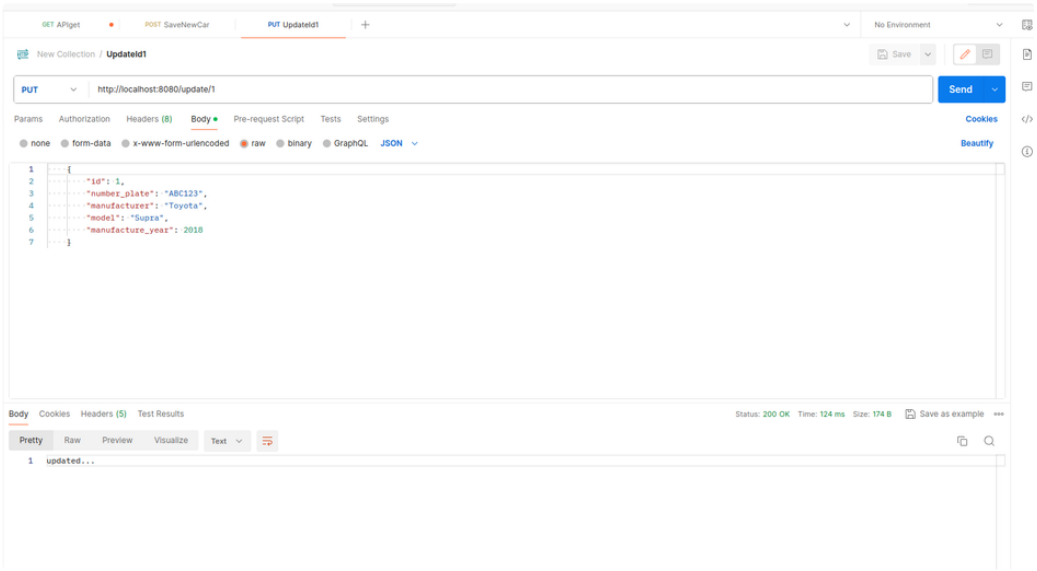
  - A lekérés és a post kipróbálásához postmant használunk.
    - file/new/HTTP/get:



  - file/new/HTTP/Post

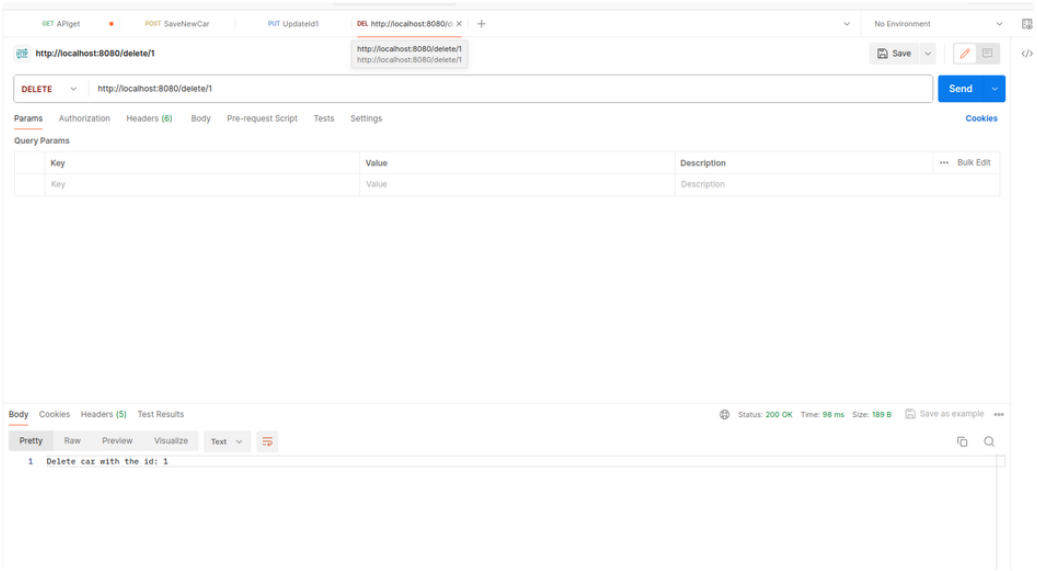


-
  - Ne felejtsük el a próba előtt újraindítani az appot
- Csináljuk meg az update methodot is:
  - ```
    @PutMapping(value = "update/{id}")
    public String updateCar(@PathVariable Long id, @RequestBody Car car) {
        Car updatedCar = carRepo.findById(id).get();
        updatedCar.setNumber_plate(car.getNumber_plate());
        updatedCar.setManufacturer(car.getManufacturer());
        updatedCar.setModel(car.getModel());
        updatedCar.setManufacture_year(car.getManufacture_year());
        carRepo.save(updatedCar);
        return "updated...";
    }
    ```



- És végül a delete method:
  - ```
    @DeleteMapping(value="delete/{id}")
    public String deleteCar(@PathVariable long id){
        Car deleteCar = carRepo.findById(id).get();
        carRepo.delete(deleteCar);
    ```

```
        return "Delete car with the id: "+id;
    }
```



ApiControllers

```java
package car.db.carDbCrud.Controller;

import car.db.carDbCrud.Models.Car;
import car.db.carDbCrud.Repo.CarRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class ApiControllers {

    @Autowired
    private CarRepo carRepo;

    @GetMapping(value = "/")
    public String getPage() {
        return "Welcome";
    }

    @GetMapping(value = "/cars")
    public List<Car> getCars() {
        return carRepo.findAll();
    }

    @PostMapping(value = "/save")
    public String saveCar(@RequestBody Car car) {
        carRepo.save(car);
        return "Saved...";
    }

    @PutMapping(value = "update/{id}")
    public String updateCar(@PathVariable Long id, @RequestBody Car car) {
        Car updatedCar = carRepo.findById(id).get();
        updatedCar.setNumber_plate(car.getNumber_plate());
```

```
        updatedCar.setManufacturer(car.getManufacturer());
        updatedCar.setModel(car.getModel());
        updatedCar.setManufacture_year(car.getManufacture_year());
        carRepo.save(updatedCar);
        return "updated...";

    }

    @DeleteMapping(value="delete/{id}")
    public String deleteCar(@PathVariable long id){
        Car deleteCar = carRepo.findById(id).get();
        carRepo.delete(deleteCar);
        return "Delete car with the id: "+id;
    }
}
```