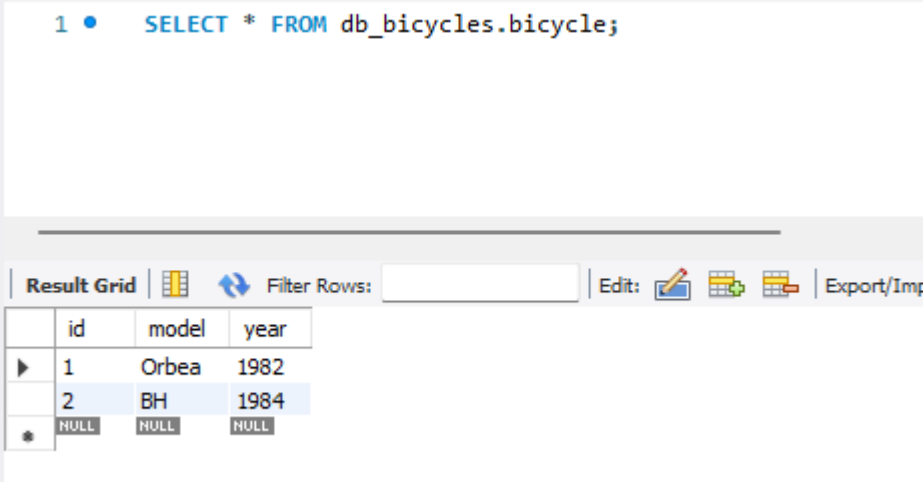




## Índice:

<b>Creación de la base de datos MySQL y mostrar lo que hay dentro:</b>	<b>2</b>
<b>Código de las clases:</b>	<b>2</b>
<b>Ahora probaremos todos los END-POINTS usando Postman:</b>	<b>8</b>
GET	8
POST	8
PUT	9
DELETE	11
<b>Gestión de errores</b>	<b>13</b>

Creación de la base de datos MySQL y mostrar lo que hay dentro:



The screenshot shows a MySQL query editor with a single query: `SELECT * FROM db_bicycles.bicycle;`. Below the query, a toolbar includes options for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Imp'. The 'Result Grid' is active, displaying a table with the following data:

	id	model	year
▶	1	Orbea	1982
	2	BH	1984
*	NULL	NULL	NULL

Código de las clases:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BicyclesApplication {

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

}

-----
import java.io.Serializable;
```

```

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "bicycle")
public class Bicycle implements Serializable{

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    private String model;

    private int year;

    public long getId() {
        return id;
    }

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

    public String getModel() {
        return model;
    }

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

    public int getYear() {
        return year;
    }
}

```

```

    public void setYear(int year) {
        this.year = year;
    }

    public Bicycle(long id, String model, int year) {
        super();
        this.model = model;
        this.year = year;
    }
    public Bicycle() {

    }

}

-----
import org.springframework.data.repository.CrudRepository;

import com.example.bicycles.entity.models.Bicycle;

public interface IBicycleDao extends CrudRepository <Bicycle,Long>{

}

-----
import java.util.List;

import com.example.bicycles.entity.models.Bicycle;

public interface IBicycleService {
    public Bicycle get(long id);
    public List<Bicycle> getAll();
    public void post(Bicycle bicycle);
    public void put(Bicycle bicycle, long id);
    public void delete(long id);
}

-----
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

```

```

import com.example.bicycles.entity.dao.IBicycleDao;
import com.example.bicycles.entity.models.Bicycle;

@Service
public class BicycleServiceImpl implements IBicycleService {

    @Autowired
    private IBicycleDao bicycleDao;

    @Override
    public Bicycle get(long id) {

        return bicycleDao.findById(id).get();
    }

    @Override
    public List<Bicycle> getAll() {
        return (List<Bicycle>) bicycleDao.findAll();
    }

    @Override
    public void post(Bicycle bicycle) {
        bicycleDao.save(bicycle);
    }

    @Override
    public void put(Bicycle bicycle, long id) {
        bicycleDao.findById(id).ifPresent((x)->{
            bicycle.setId(id);
            bicycleDao.save(bicycle);
        });
    }

    @Override
    public void delete(long id) {
        bicycleDao.deleteById(id);
    }
}

```

```

-----

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.bicycles.entity.models.Bicycle;
import com.example.bicycles.entity.services.IBicycleService;

@RestController
public class BicycleController{

    @Autowired
    IBicycleService bicycleService;

    @GetMapping("/bicycle")
    public List<Bicycle> getAllBicycles() {
        return bicycleService.getAll();
    }

    @GetMapping("/bicycle/{id}")
    public Bicycle getOne(@PathVariable(value = "id") long id) {
        return bicycleService.get(id);
    }

    @PostMapping("/bicycle")
    public void post(Bicycle bicycle) {
        bicycleService.post(bicycle);
    }

    @PutMapping("/bicycle/{id}")
    public void put(Bicycle bicycle, @PathVariable(value = "id") long id) {
        bicycleService.put(bicycle, id);
    }

    @DeleteMapping("/bicycle/{id}")

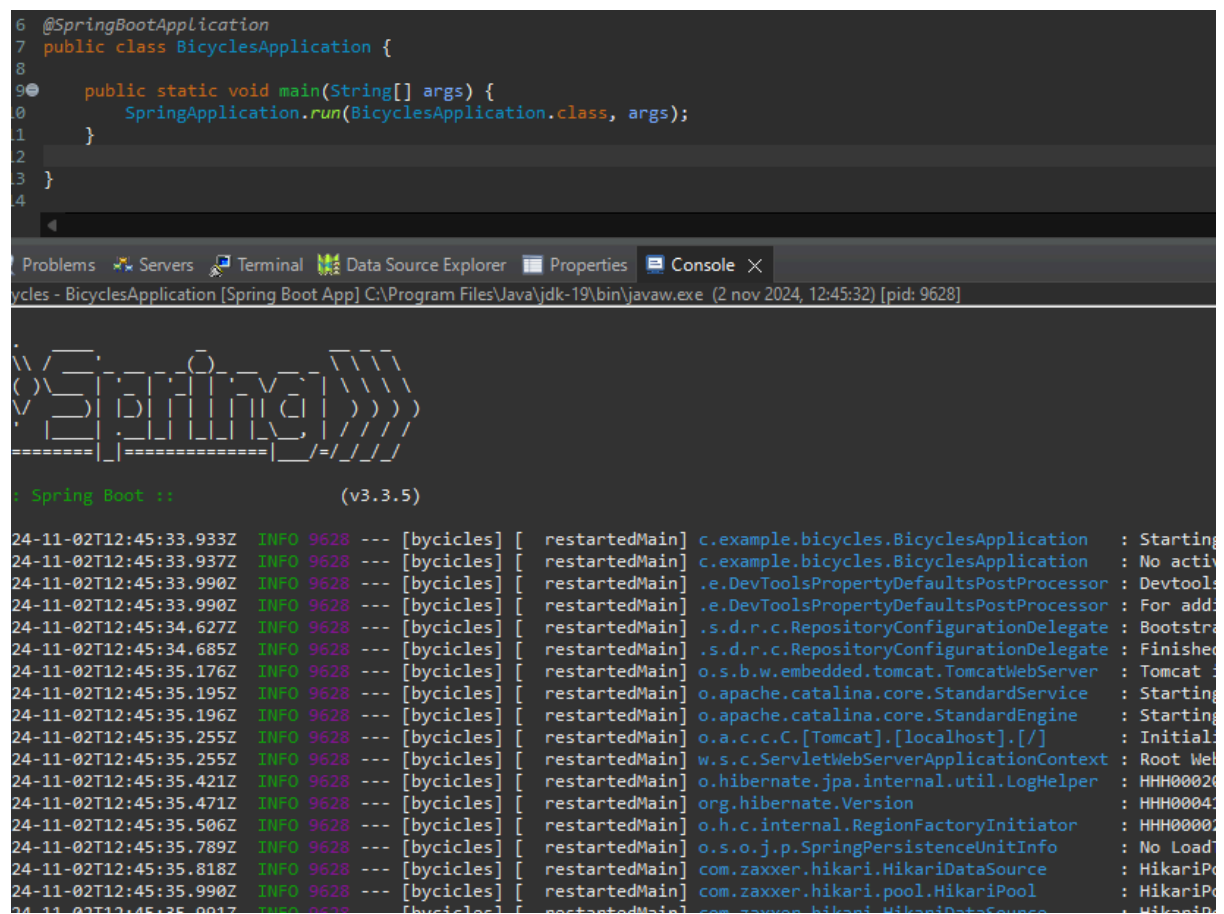
```

```

    public void delete(@PathVariable(value = "id") long id) {
        bicycleService.delete(id);
    }
}

```

Lo ejecutamos como una aplicacion de Spring Boot App y vemos que funciona sin problemas:



The screenshot shows an IDE with the following code in `BicyclesApplication.java`:

```

6 @SpringBootApplication
7 public class BicyclesApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(BicyclesApplication.class, args);
11     }
12
13 }
14

```

The console output shows the application starting successfully. The output is as follows:

```

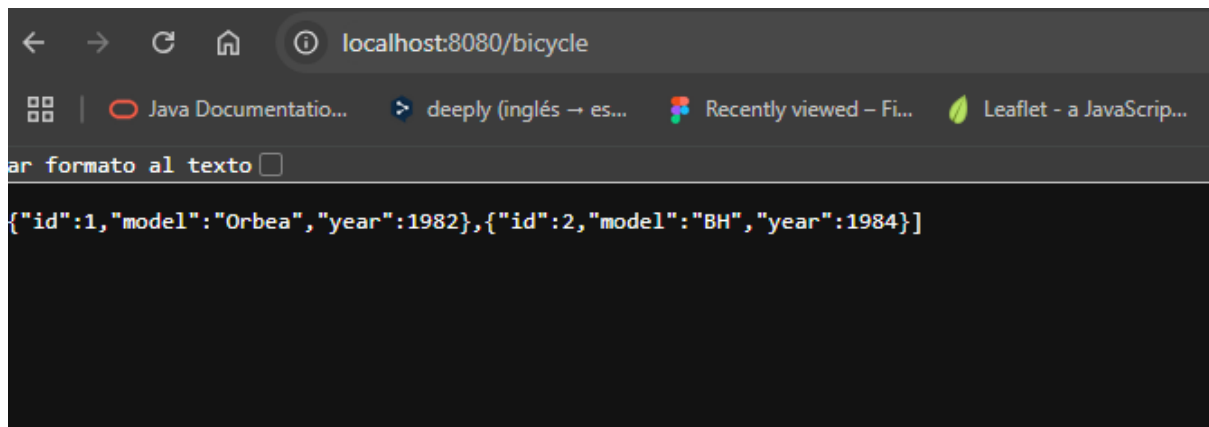
Spring Boot :: (v3.3.5)

24-11-02T12:45:33.933Z INFO 9628 --- [bycycles] [ restartedMain] c.example.bicycles.BicyclesApplication : Starting
24-11-02T12:45:33.937Z INFO 9628 --- [bycycles] [ restartedMain] c.example.bicycles.BicyclesApplication : No activ
24-11-02T12:45:33.990Z INFO 9628 --- [bycycles] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtool
24-11-02T12:45:33.990Z INFO 9628 --- [bycycles] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For add
24-11-02T12:45:34.627Z INFO 9628 --- [bycycles] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstr
24-11-02T12:45:34.685Z INFO 9628 --- [bycycles] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finishe
24-11-02T12:45:35.176Z INFO 9628 --- [bycycles] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat :
24-11-02T12:45:35.195Z INFO 9628 --- [bycycles] [ restartedMain] o.apache.catalina.core.StandardService : Starting
24-11-02T12:45:35.196Z INFO 9628 --- [bycycles] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting
24-11-02T12:45:35.255Z INFO 9628 --- [bycycles] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initiali
24-11-02T12:45:35.255Z INFO 9628 --- [bycycles] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root Web
24-11-02T12:45:35.421Z INFO 9628 --- [bycycles] [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH00020
24-11-02T12:45:35.471Z INFO 9628 --- [bycycles] [ restartedMain] org.hibernate.Version : HHH00041
24-11-02T12:45:35.506Z INFO 9628 --- [bycycles] [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH00002
24-11-02T12:45:35.789Z INFO 9628 --- [bycycles] [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No Load
24-11-02T12:45:35.818Z INFO 9628 --- [bycycles] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPo
24-11-02T12:45:35.990Z INFO 9628 --- [bycycles] [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPo
24-11-02T12:45:35.991Z INFO 9628 --- [bycycles] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPo

```

Y lo buscamos en el puerto 8080 y vemos que nos muestra nuestra base de datos:



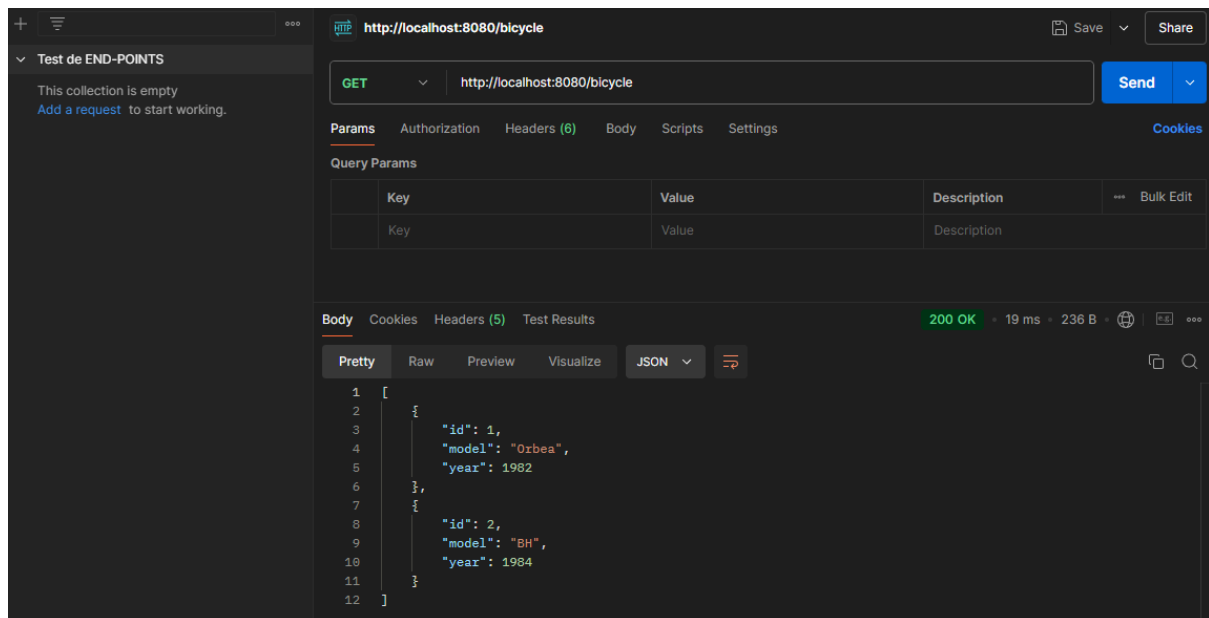


Ahora probaremos todos los END-POINTS usando Postman:

Primero Descargamos Postman

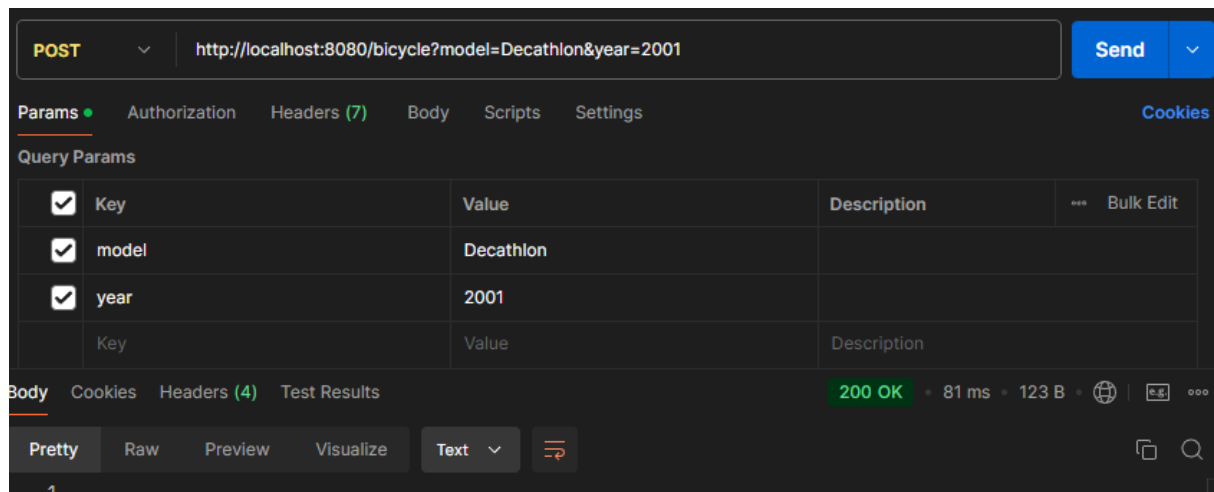
GET

Probamos la petición GET y como vemos que nos devuelve la información almacenada en la base de datos, ha funcionado.

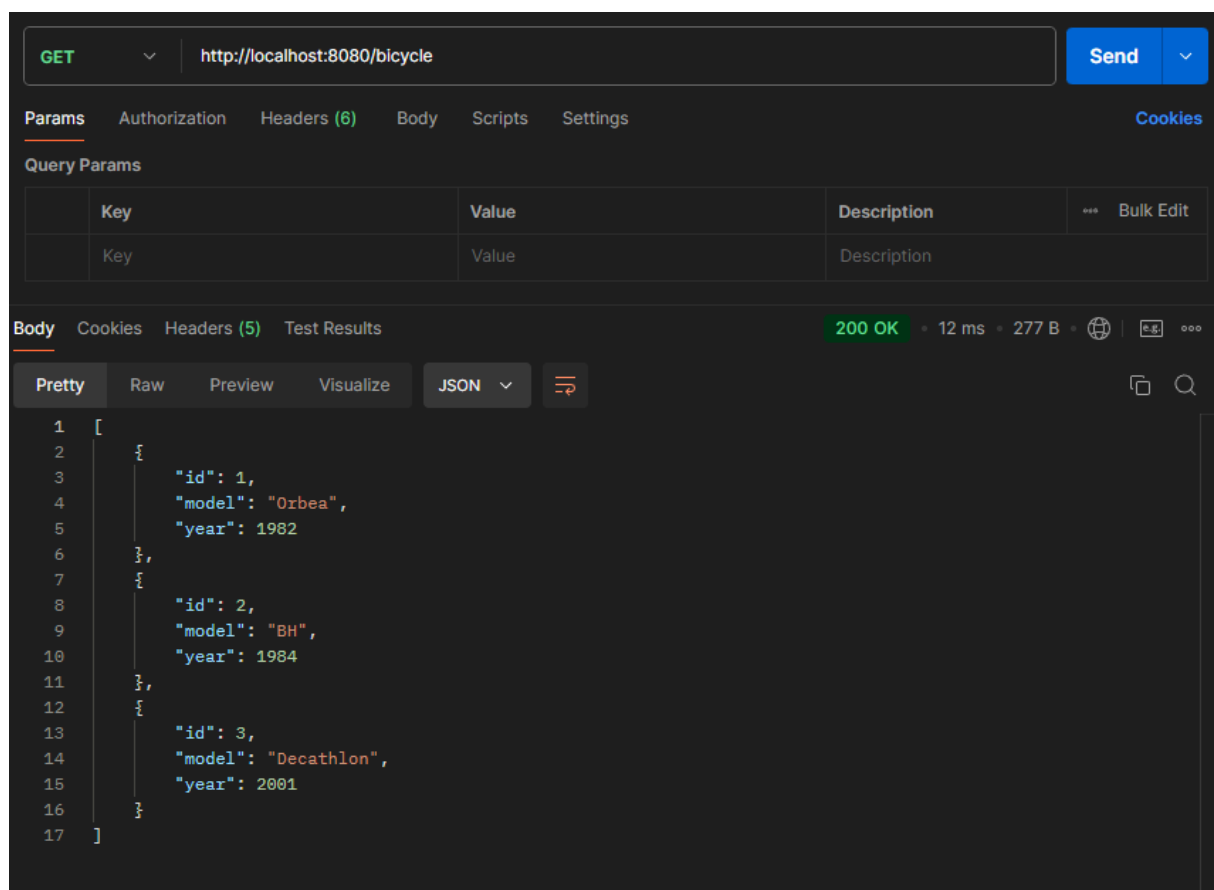


POST

Ahora hacemos una petición POST para añadir una nueva bicicleta.

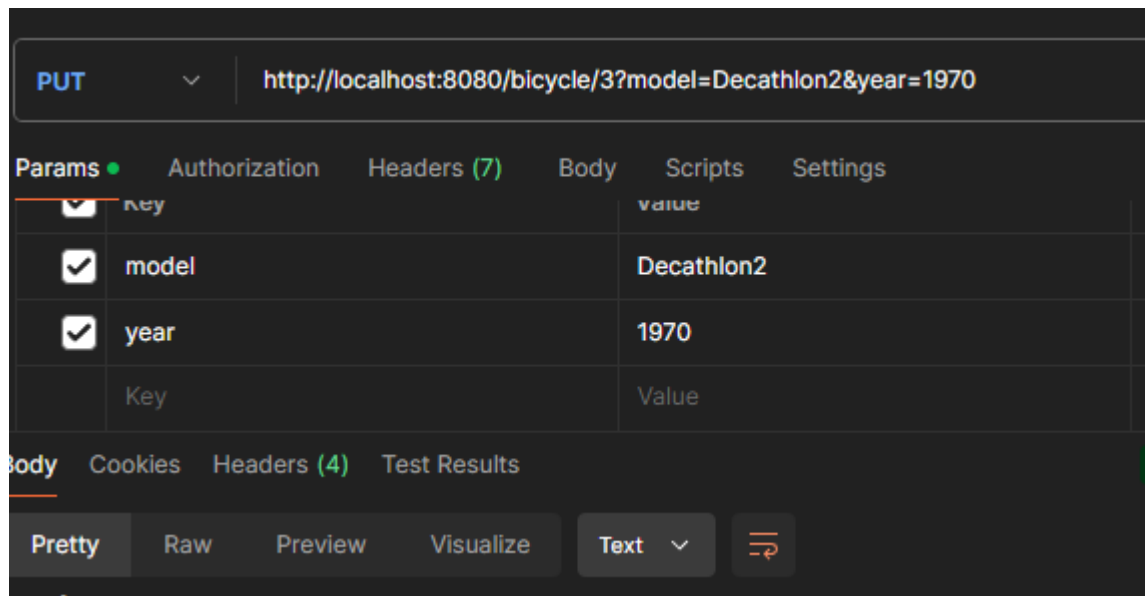


Ahora hacemos una petición GET para comprobar que la nueva bicicleta ha sido insertada y por tanto nuestra petición POST anterior funcionó.

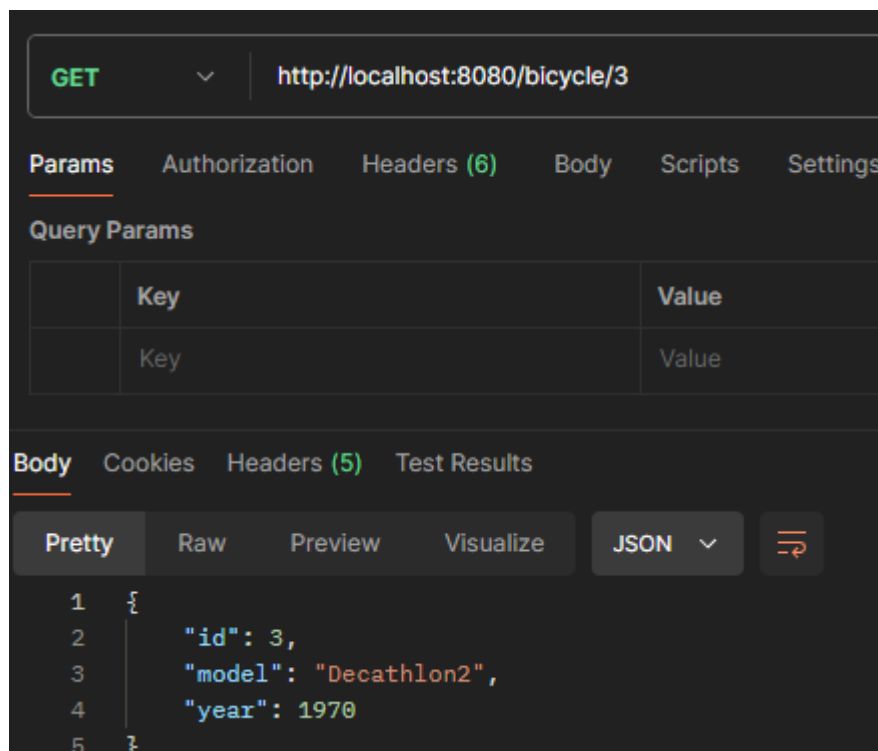


PUT

Ahora hacemos una petición PUT para modificar la bicicleta introducida anteriormente.

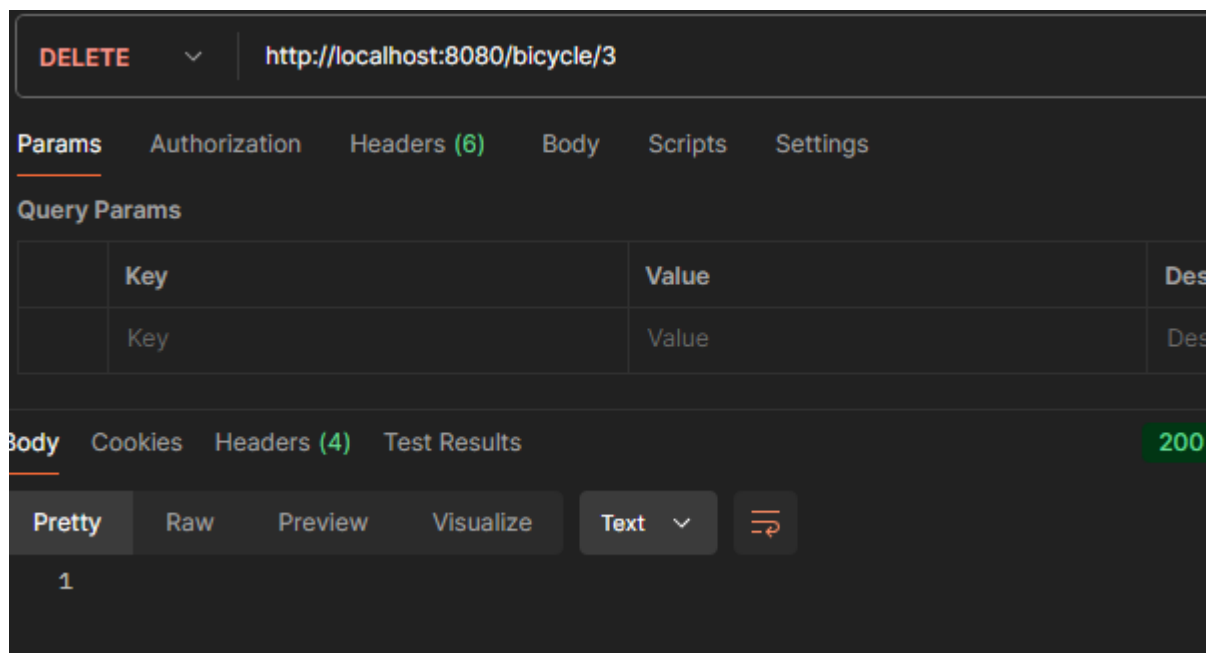


Ahora hacemos una petición GET y se comprueba que la bicicleta insertada anteriormente ha sido actualizada y por tanto nuestra petición PUT anterior funcionó.



## DELETE

Ahora hacemos una petición DELETE para borrar la bicicleta introducida anteriormente.



Ahora hacemos una petición GET y se comprueba que la bicicleta insertada anteriormente ha sido borrada y por tanto nuestra petición DELETE anterior funcionó.

GET http://localhost:8080/bicycle

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1  [
2    {
3      "id": 1,
4      "model": "Orbea",
5      "year": 1982
6    },
7    {
8      "id": 2,
9      "model": "BH",
10     "year": 1984
11   }
12 ]
```

Test de END-POINTS

- GET http://localhost:8080/bicycle
- GET http://localhost:8080/bicycle/3
- POST http://localhost:8080/bicycle
- PUT http://localhost:8080/bicycle/3
- DEL http://localhost:8080/bicycle/3

## Gestión de errores

```
public class ResourceNotFoundException extends RuntimeException {
    public ResourceNotFoundException(String message) {
        super(message);
    }
}
```

```
-----

import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.http.HttpStatus;

@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ResourceNotFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public String handleResourceNotFoundException(ResourceNotFoundException
ex) {
        return ex.getMessage();
    }
}
```

```
-----

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.bicycles.entity.dao.IBicycleDao;
import com.example.bicycles.entity.models.Bicycle;
import com.example.bicycles.exceptions.ResourceNotFoundException;

@Service
public class BicycleServiceImpl implements IBicycleService {

    @Autowired
    private IBicycleDao bicycleDao;

    @Override
```

```

    public Bicycle get(long id) {
        return bicycleDao.findById(id)
            .orElseThrow(() -> new ResourceNotFoundException("Bicycle not
found with ID " + id));
    }

    @Override
    public List<Bicycle> getAll() {
        return (List<Bicycle>) bicycleDao.findAll();
    }

    @Override
    public void post(Bicycle bicycle) {
        bicycleDao.save(bicycle);
    }

    @Override
    public void put(Bicycle bicycle, long id) {
        if (!bicycleDao.existsById(id)) {
            throw new ResourceNotFoundException("Bicycle not found with ID "
+ id);
        }
        bicycle.setId(id);
        bicycleDao.save(bicycle);
    }

    @Override
    public void delete(long id) {
        if (!bicycleDao.existsById(id)) {
            throw new ResourceNotFoundException("Bicycle not found with ID "
+ id);
        }
        bicycleDao.deleteById(id);
    }
}
-----

```