Tutorial 3 - Parte 3

Spring

FCHAVEZ

Inyección de dependencias

- Las clases son un mecanismo de modularización que nos permite hacer administrable la complejidad de los proyectos.
- Las clases, que modelan funciones sencillas, deben trabajar unas con otras para crear comportamiento complejo, lo que crea dependencias.
- El concepto de inyección de dependencias señala que una clase B que necesita de una clase A NO debe instanciar los objetos de tipo A que necesita, sino recibirlos (mediante un método set) para evitar dependencias entre la implementación de A y la de B.

Inversión de control

- La inyección de dependencias potencia la inversión de control.
  - En la inyección de control un “agente” es el encargado de inyectar las dependencias de forma adecuada. Para el caso de Spring este agente es el Spring Framework.
  - El programador cuenta con la anotación @Autowired para indicar a Spring Framework donde inyectar dependencias.

IoC en el proyecto

```
@Repository
public class UsuarioRepository implements UserRepository{
    private UsuarioCrudRepository repository;
    private UserMapper mapper;
…
```

Dependencias

Nota: en este momento ninguna de estas dos variables es instancia, por lo que si las ocuparemos obtendríamos un `NullPointerException`.

IoC en el proyecto

```
@Repository
public class UsuarioRepository implements UserRepository{
    @Autowired
    private UsuarioCrudRepository repository;
    @Autowired
    private UserMapper mapper;
```

Solicitamos a Spring Framework que instancie estos objetos y que ademas los inyecte a la instancia de `UsuarioRepository` que también creará.

Wow

¡Qué mágico!

Buenoooo … no tanto.

Sólo puede instanciar e inyectar objetos que sean componentes de Spring

¿Dudas?

- ¿Y dónde veo que las dependencias que estamos inyectando son componentes de Spring?

```
public interface UsuarioCrudRepository extends CrudRepository<Usuario, Integer> {
    Usuario findByLogin(String login);
}
```

Al heredar de `CrudRepository` (como esta clase es un componente de Spring) `UsuarioCrudRepository` se hace un componente

¿Dudas?

- ¿Y dónde veo que las dependencias que estamos inyectando son componentes de Spring?

```
@Mapper(componentModel = "spring")
public interface UserMapper {
    @Mappings({
            @Mapping(source = "id", target = "id"),
            @Mapping(source = "login", target = "login"),
            @Mapping(source="contrasena", target="password"),
            @Mapping(source="correo", target="email")
    })
```

Le decimos que use el modelo de componentes de Spring

Terminamos ...

- Por fin terminamos el repositorio de nuestra aplicación

Terminamos ...

- Descansa un poco

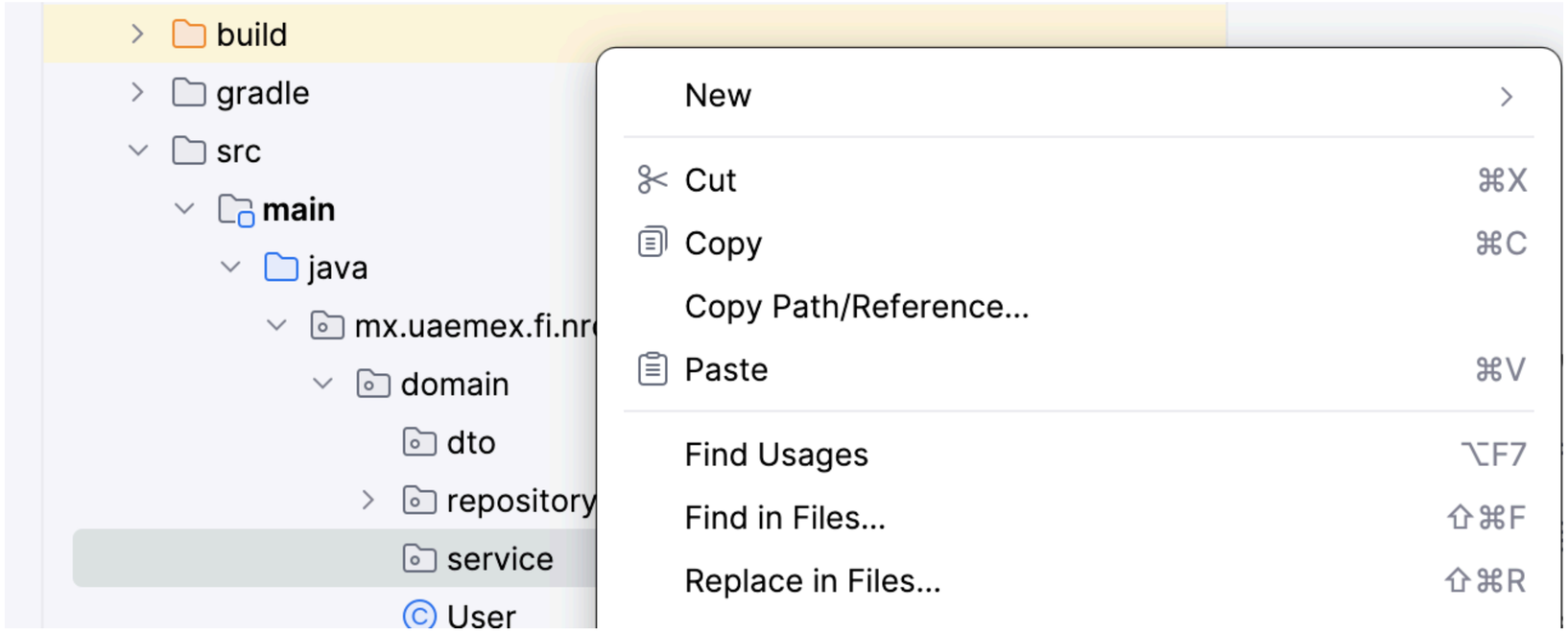
porque vamos a seguir

Servicios de dominio

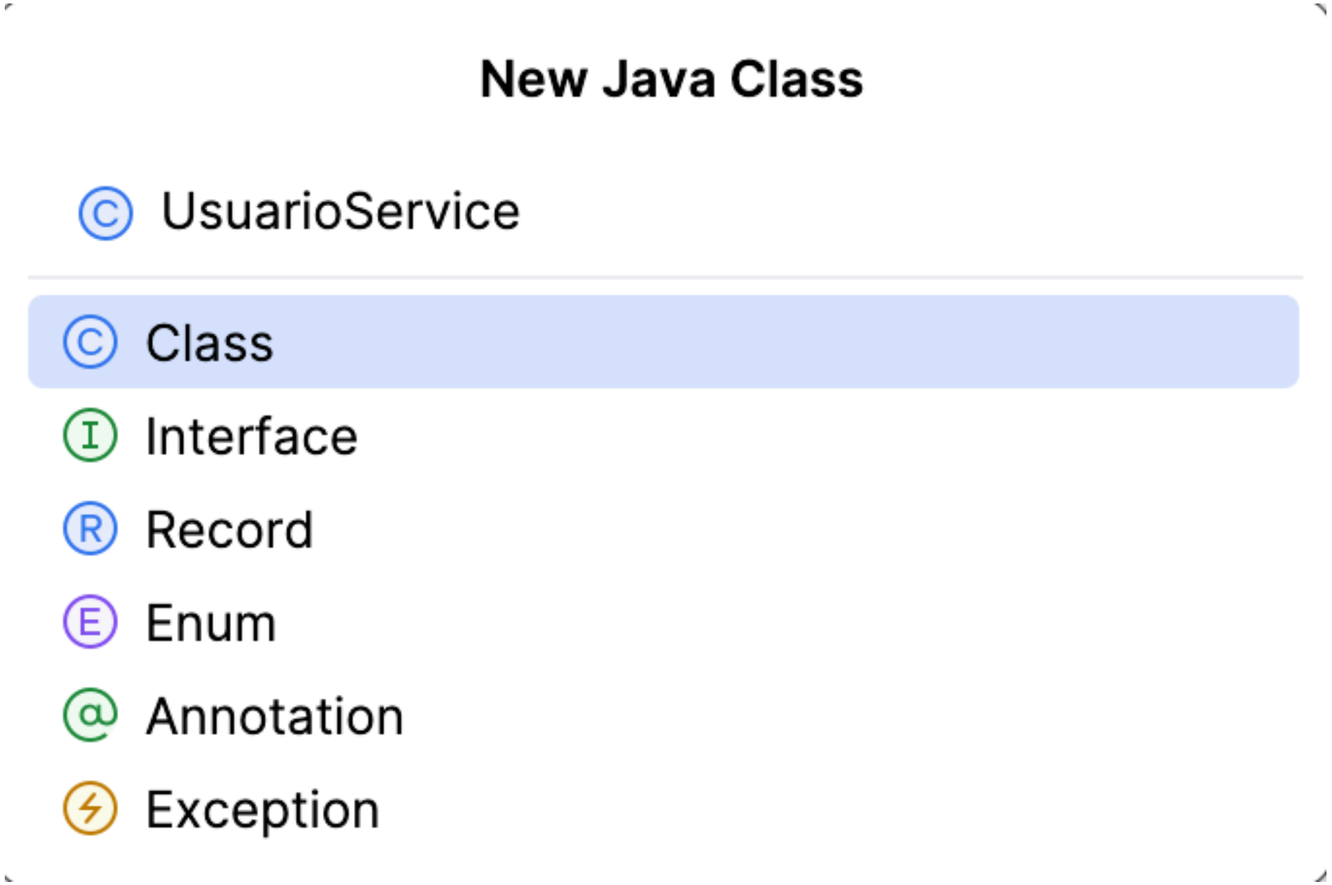
- Interfaz entre el controlador y el repositorio

Servicio de Usuarios

- Click derecho sobre el paquete: service

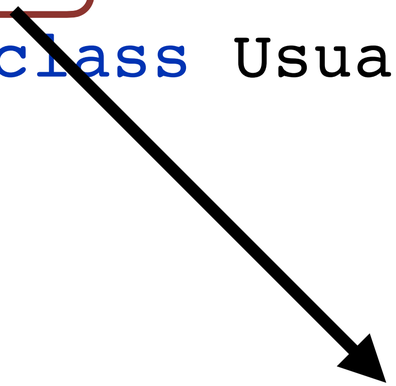


Servicio de Usuarios



Servicio de Usuarios

```
@Service
public class UsuarioService {
}
```



Hacemos que la clase sea un servicio

Servicio de Usuarios

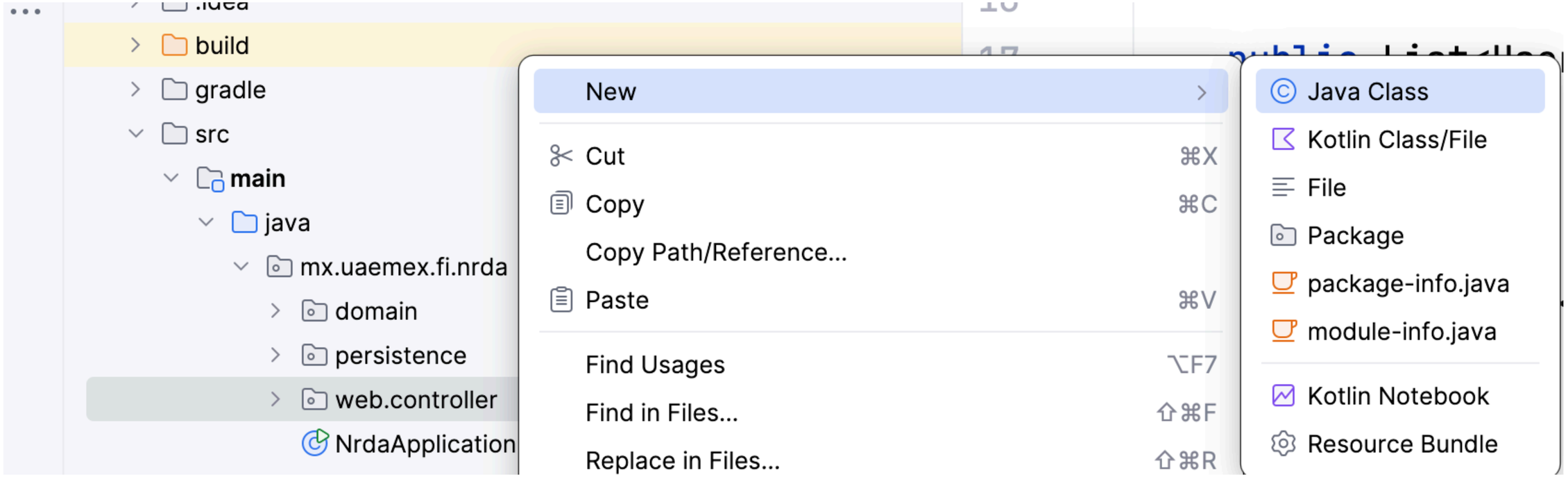
```java
@Service
public class UsuarioService {

    @Autowired
    private UsuarioRepository usuarioRepository;

    public List<User> findAll() {
        return this.usuarioRepository.getAll();
    }

    public Optional<User> getUser(int id){
        return this.usuarioRepository.getUser(id);
    }

    public User save(User user) {
        return this.usuarioRepository.save(user);
    }

    public boolean delete(int id){
        return getUser(id).map(user -> {usuarioRepository.deleteById(id); return true;}).orElse(false);
    }
}
```

Servicio

Controll time

- Ya estamos listos para hacer el controlador

Controller

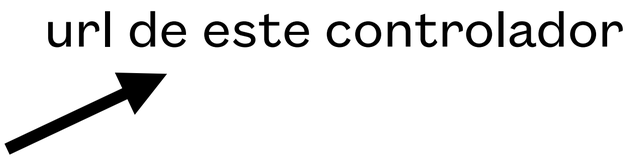

build
gradle
src
main
java
mx.uaemex.fi.nrda
domain
persistence
web.controller
NrdaApplication
New
Cut ⌘X
Copy ⌘C
Copy Path/Reference...
Paste ⌘V
Find Usages ⌥F7
Find in Files... ⇧⌘F
Replace in Files... ⇧⌘R
Java Class
Kotlin Class/File
File
Package
package-info.java
module-info.java
Kotlin Notebook
Resource Bundle

UsuarioController

Lo hacemos un controlador

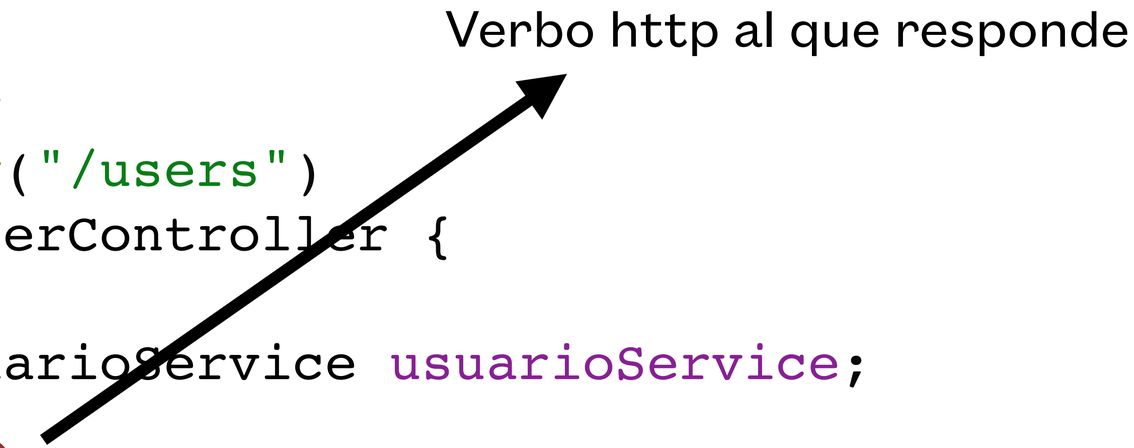
```
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    private UsuarioService usuarioService;

    @GetMapping("all")
    public List<User> getAll(){
        return usuarioService.findAll();
    }
```

UsuarioController



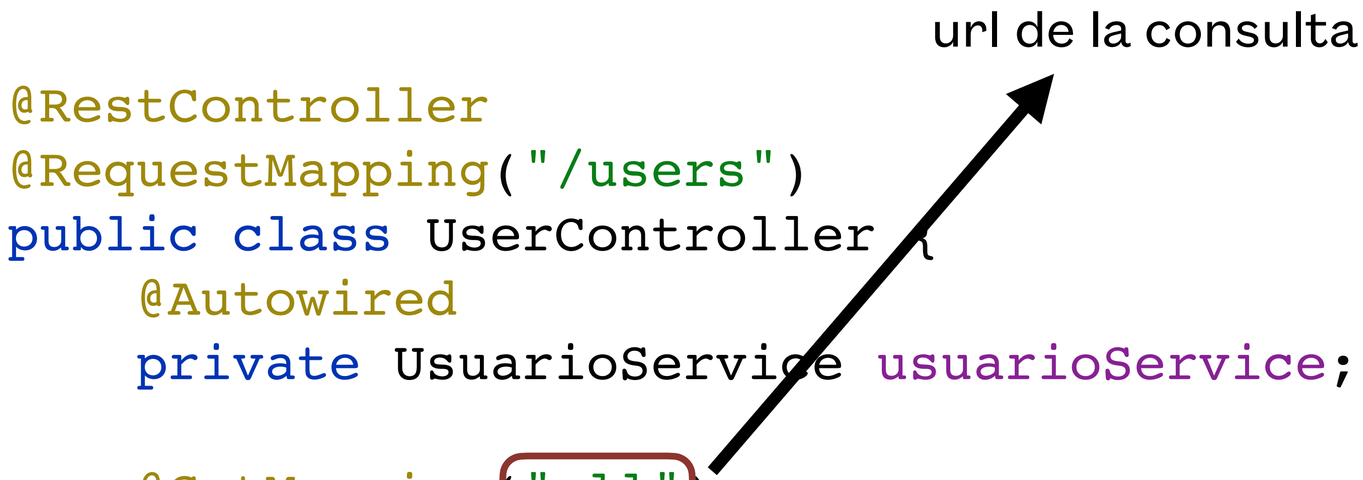

```
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    private UsuarioService usuarioService;

    @GetMapping("all")
    public List<User> getAll(){
        return usuarioService.findAll();
    }
```

UsuarioController




```
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    private UsuarioService usuarioService;

    @GetMapping("all")
    public List<User> getAll(){
        return usuarioService.findAll();
    }
```

UsuarioController




```
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    private UsuarioService usuarioService;

    @GetMapping("all")
    public List<User> getAll(){
        return usuarioService.findAll();
    }
```

UsuarioController


```
    @GetMapping("/{id}") no usages
    public Optional<User> getUser(int id){
        return usuarioService.getUser(id);
    }

    @PostMapping("/save")  no usages
    public User save(User user){
        return usuarioService.save(user);
    }

    @DeleteMapping("/{id}")  no usages
    public boolean delete(int userId){
        return usuarioService.delete(userId);
    }
}
```

UsuarioController

Verbo http y url

```
    @GetMapping("/{id}")  no usages
    public Optional<User> getUser(int id){
        return usuarioService.getUser(id);
    }

    @PostMapping("/save")  no usages
    public User save(User user){
        return usuarioService.save(user);
    }

    @DeleteMapping("/{id}")  no usages
    public boolean delete(int userId){
        return usuarioService.delete(userId);
    }
}
```

UsuarioController

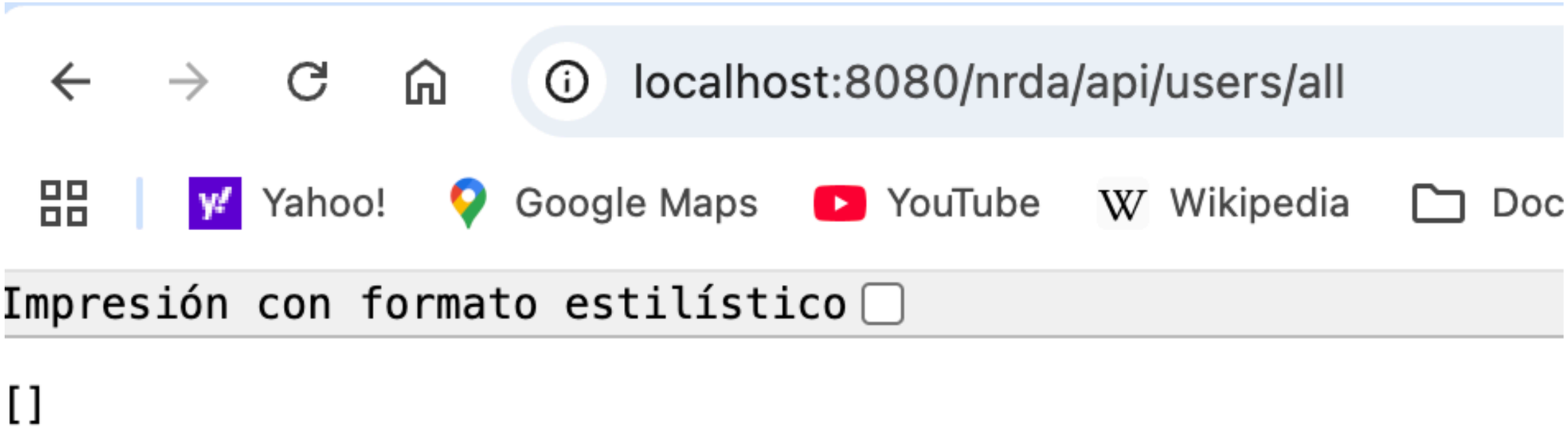
```
    @GetMapping("/{id}")  no usages
    public Optional<User> getUser(int id){
        return usuarioService.getUser(id);
    }

    @PostMapping("/save")  no usages
    public User save(User user){
        return usuarioService.save(user);
    }

    @DeleteMapping("/{id}") no usages
    public boolean delete(int userId){
        return usuarioService.delete(userId);
    }
}
```

Verbo http y url

¡Listo!

- Hora de probar

Prueba

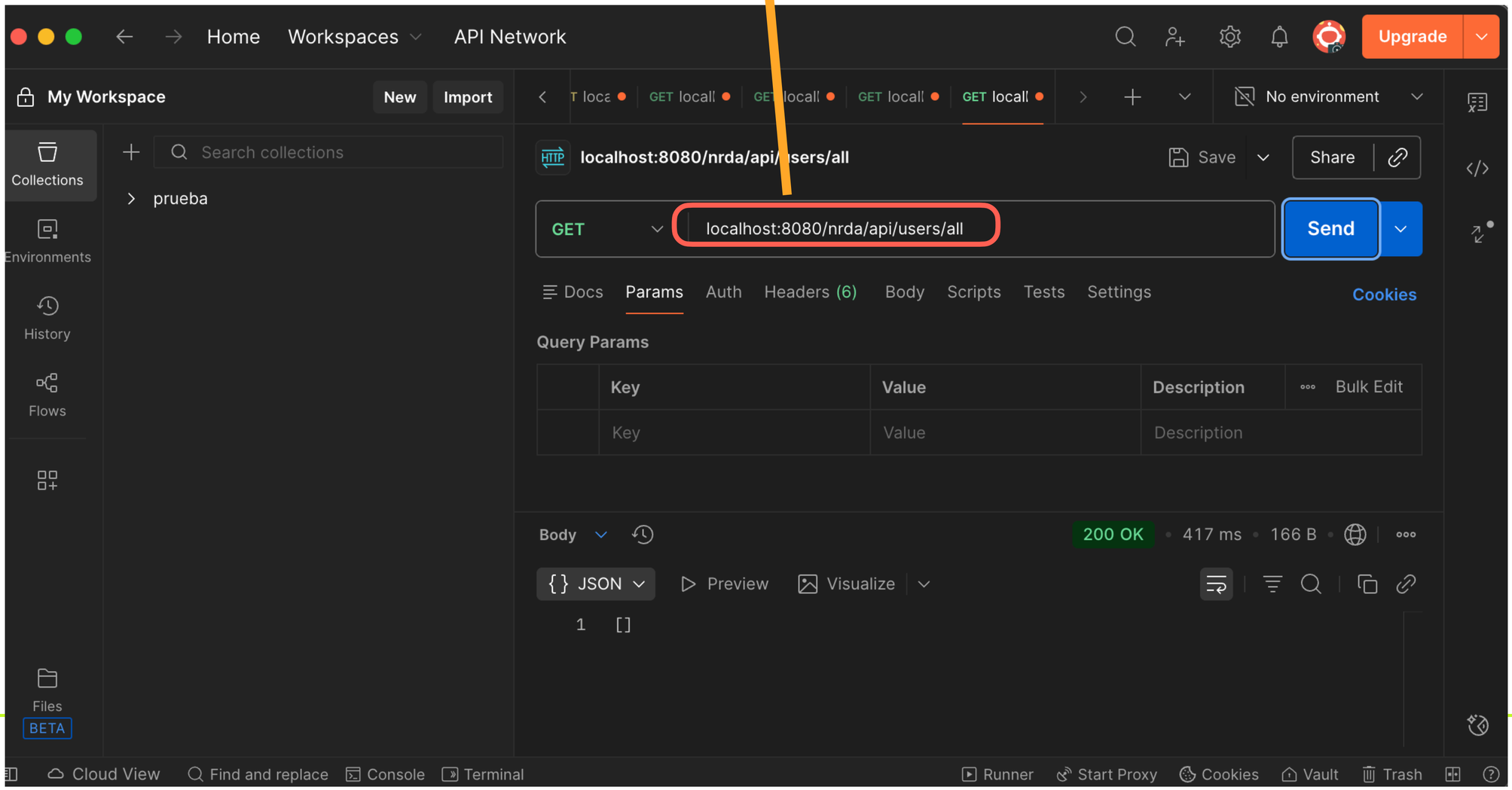



```
nrda=# select * from usuarios;
 id | login | password | email | apellido_paterno | apellido_materno | correo | telefono
----+-------+----------+-------+------------------+------------------+--------+----------
(0 rows)
```

Prueba

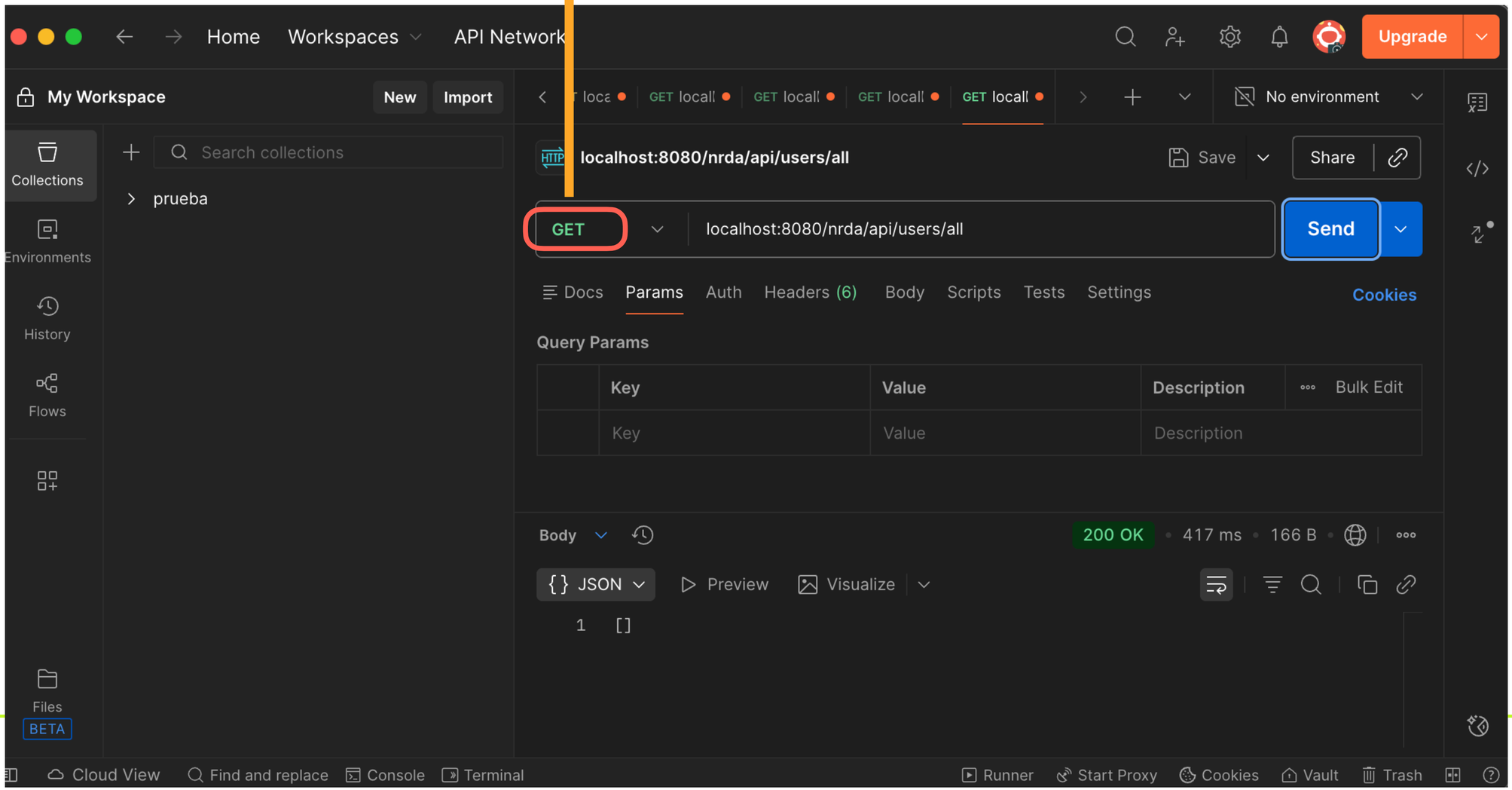
Prueba

Postman



Prueba

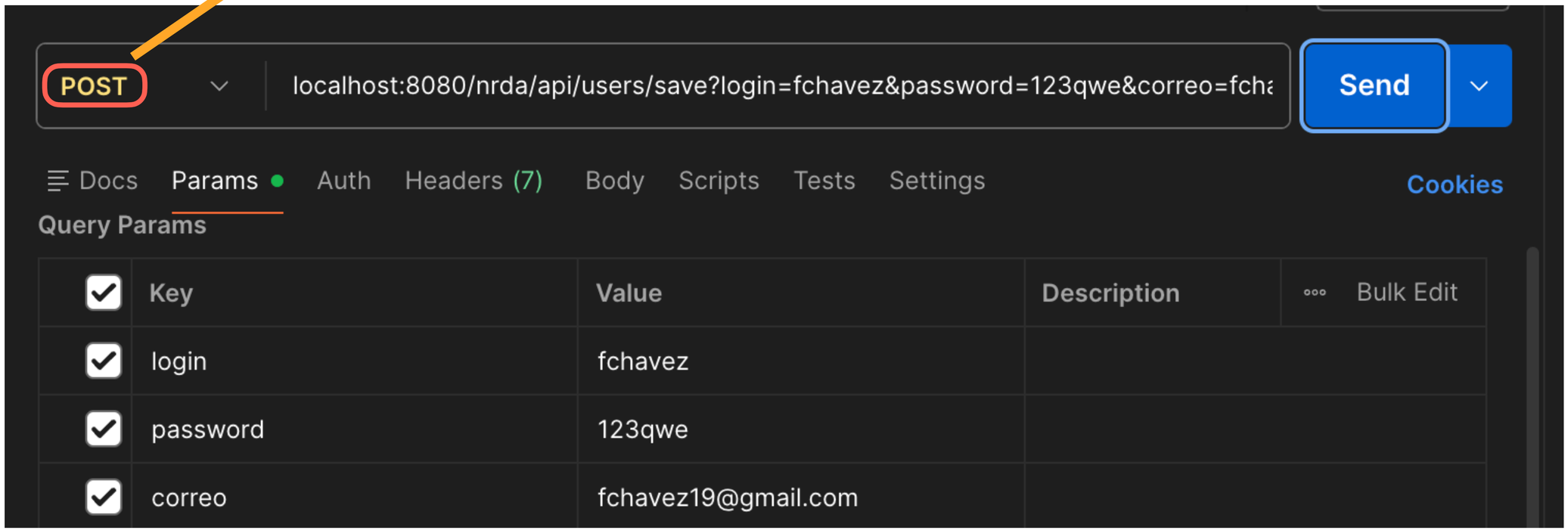
Postman



Prueba

Postman

Verbo http



Prueba

Postman

Parámetros