



# SPRING BOOT + MONGODB

ESTUDIANTE: NICOLAS VILLÓN



# Ambiente de trabajo

- Hardware: laptop Gateway 15' p (Intel Pentium cpu p6200 @ 2,13Ghz)
- Sistema operativo: Windows 7 64 bits.
- Entorno de desarrollo: Eclipse Mars
- Spring Boot 2.0 (Apache Tomcat embebido)

# Construcción de un servicio Rest

- Permite inicializar archivo de configuración.

The screenshot shows the Spring Initializr web application in a browser. The address bar shows 'Seguro | https://start.spring.io'. The main header is 'SPRING INITIALIZR bootstrap your application now'. Below this, there are three dropdown menus: 'Generate a' with 'Gradle Project' selected, 'with' with 'Java' selected, and 'and Spring Boot' with '2.0.0' selected. The interface is divided into two main sections: 'Project Metadata' and 'Dependencies'. In 'Project Metadata', there are input fields for 'Group' (containing 'com.example') and 'Artifact' (containing 'demo'). In 'Dependencies', there is a search bar with the text 'Web, Security, JPA, Actuator, Devtools...' and a list of 'Selected Dependencies' showing 'Web' and 'DevTools' with close buttons. A large green button labeled 'Generate Project alt + ⌘' is centered below these sections. At the bottom, there is a footer with the text 'start.spring.io is powered by Spring Initializr and Pivotal Web Services'. A download bar at the very bottom shows a file named 'demo.zip' and a button labeled 'Mostrar todo'.

← → ↻ Seguro | https://start.spring.io ☆ ⋮

## SPRING INITIALIZR bootstrap your application now

Generate a Gradle Project ▾ with Java ▾ and Spring Boot 2.0.0 ▾

### Project Metadata

Artifact coordinates

Group

Artifact

### Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

Web × DevTools ×

[Generate Project alt + ⌘](#)

Don't know what to look for? Want more options? [Switch to the full version.](#)

start.spring.io is powered by [Spring Initializr](#) and [Pivotal Web Services](#)

demo.zip ^

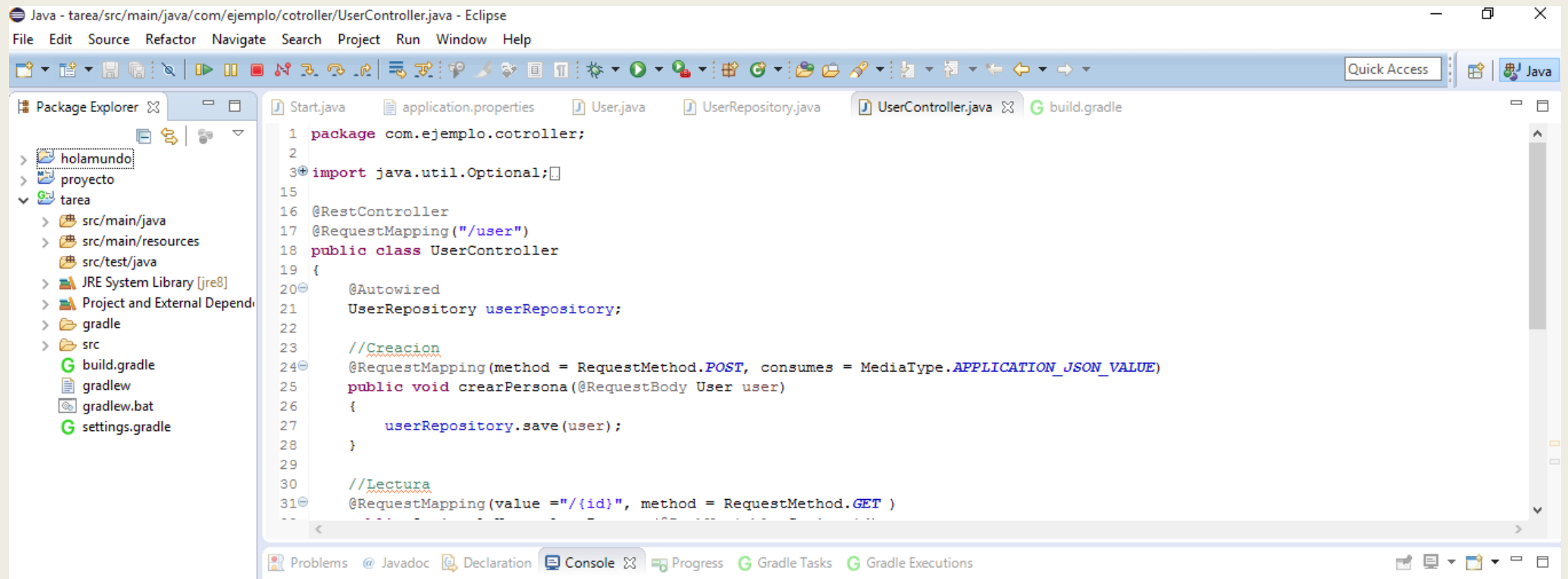
Mostrar todo ×

# Uso del proyecto generado desde la página <https://start.spring.io/>

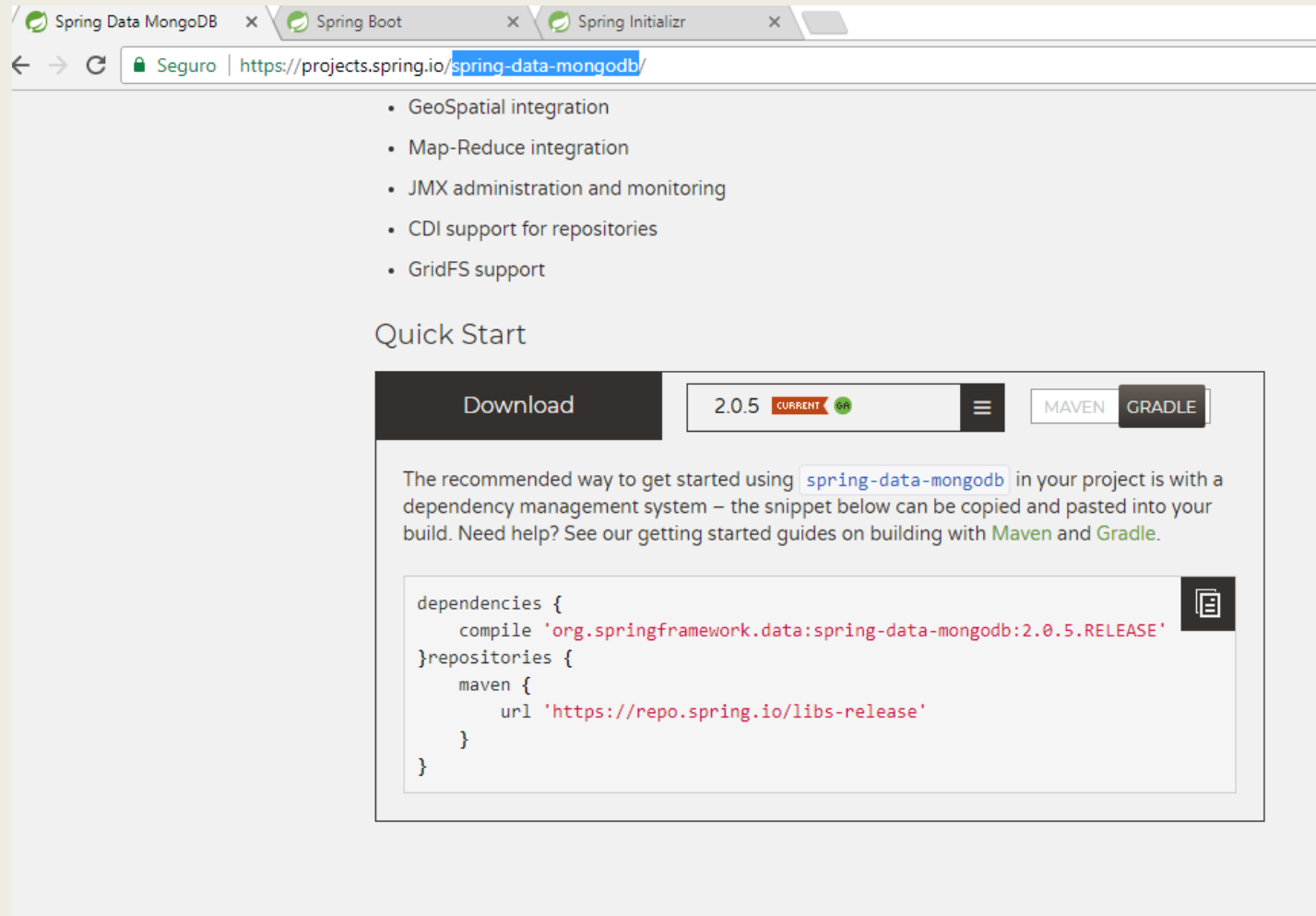
Tomar en cuenta el tipo de proyecto generado.

Importar el proyecto.

# Ide de desarrollo (Eclipse)



# SPRING-DATA-MONGODB



The screenshot shows a web browser with three tabs: 'Spring Data MongoDB', 'Spring Boot', and 'Spring Initializr'. The address bar shows the URL 'https://projects.spring.io/spring-data-mongodb/'. The page content includes a list of features, a 'Quick Start' section, and a 'Download' section with a version selector set to '2.0.5' (marked as 'CURRENT'). Below the version selector are buttons for 'MAVEN' and 'GRADLE'. The 'Quick Start' section contains a paragraph of text and a code snippet for Maven dependencies.

- GeoSpatial integration
- Map-Reduce integration
- JMX administration and monitoring
- CDI support for repositories
- GridFS support

### Quick Start

Download

2.0.5 CURRENT

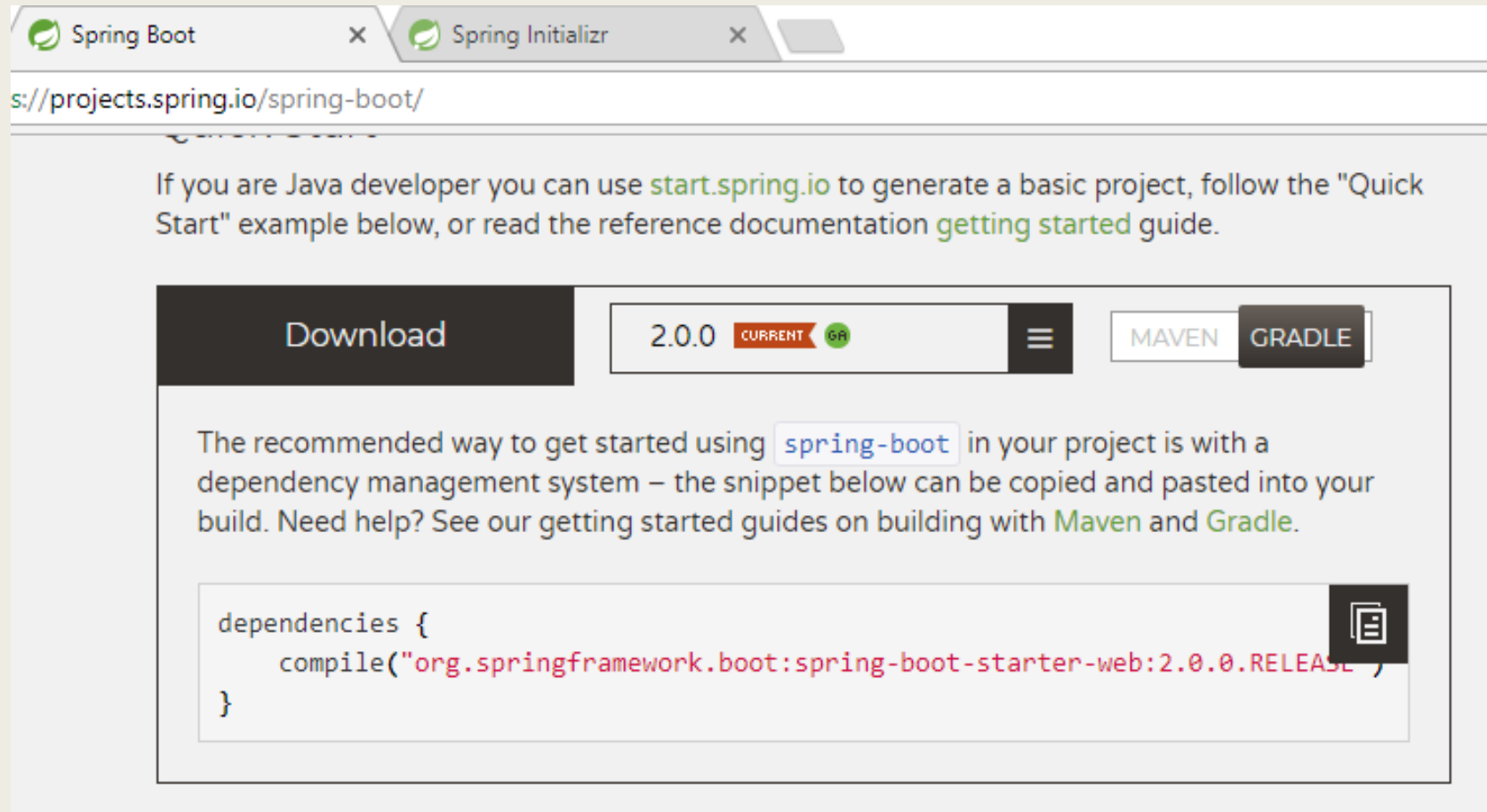
MAVEN

GRADLE

The recommended way to get started using [spring-data-mongodb](#) in your project is with a dependency management system – the snippet below can be copied and pasted into your build. Need help? See our getting started guides on building with [Maven](#) and [Gradle](#).

```
dependencies {
    compile 'org.springframework.data:spring-data-mongodb:2.0.5.RELEASE'
}repositories {
    maven {
        url 'https://repo.spring.io/libs-release'
    }
}
```

# SPRING-BOOT



The screenshot shows a web browser with two tabs: 'Spring Boot' and 'Spring Initializr'. The address bar shows the URL 's://projects.spring.io/spring-boot/'. The main content area has a heading 'Getting started' and a paragraph: 'If you are Java developer you can use [start.spring.io](#) to generate a basic project, follow the "Quick Start" example below, or read the reference documentation [getting started](#) guide.'

Below the text is a 'Download' button and a version selector showing '2.0.0' with 'CURRENT' and 'GA' badges. To the right are 'MAVEN' and 'GRADLE' buttons. A text block explains the recommended way to get started using 'spring-boot' in a project, mentioning dependency management and providing a code snippet for Maven. The code snippet is: 

```
dependencies {
    compile("org.springframework.boot:spring-boot-starter-web:2.0.0.RELEASE")
}
```

 A copy icon is visible next to the code.

Download

2.0.0 CURRENT GA

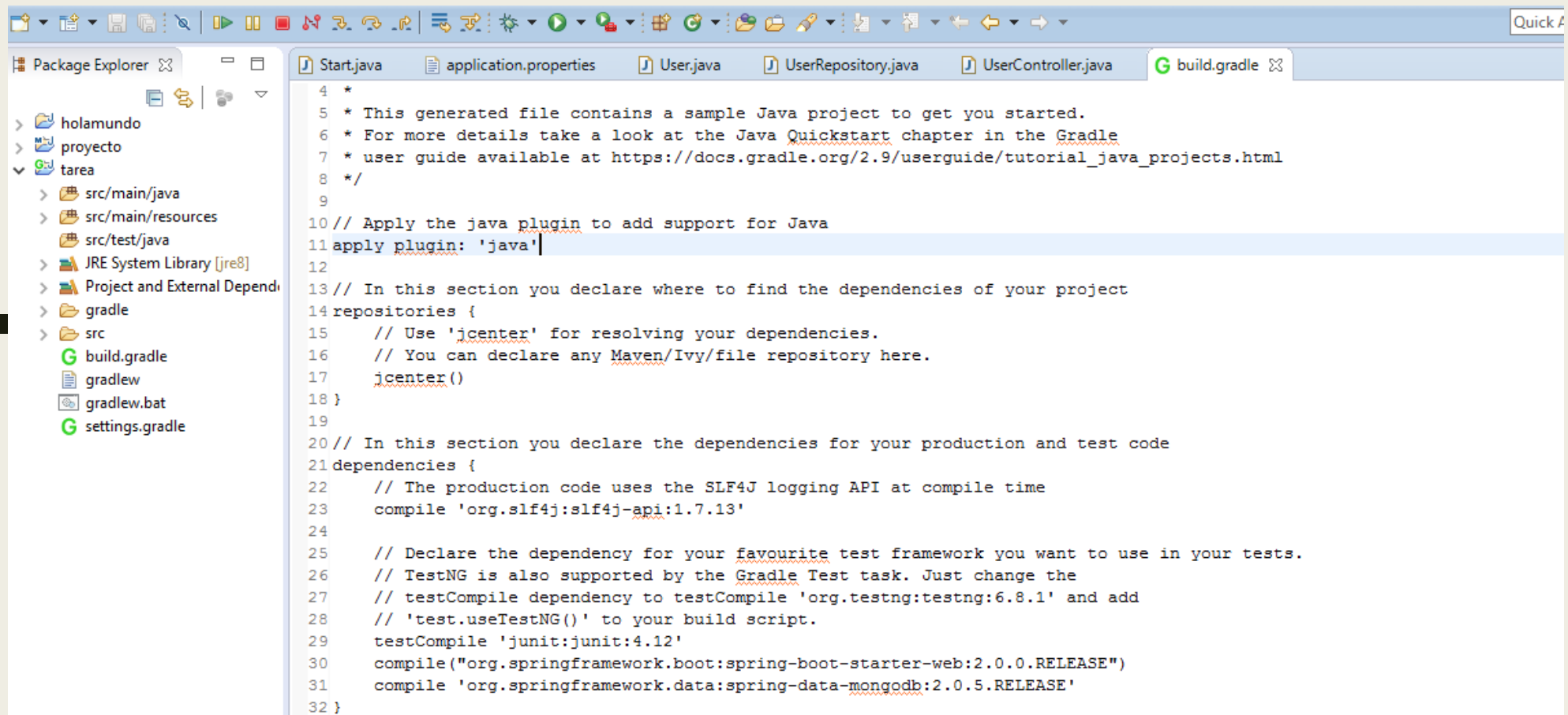
MAVEN GRADLE

The recommended way to get started using `spring-boot` in your project is with a dependency management system – the snippet below can be copied and pasted into your build. Need help? See our getting started guides on building with [Maven](#) and [Gradle](#).

```
dependencies {
    compile("org.springframework.boot:spring-boot-starter-web:2.0.0.RELEASE")
}
```

# Desarrollo del proyecto

- Modificando el archivo build.gradle para agregar las dependencias Spring Boot y Spring Data MongoDB.



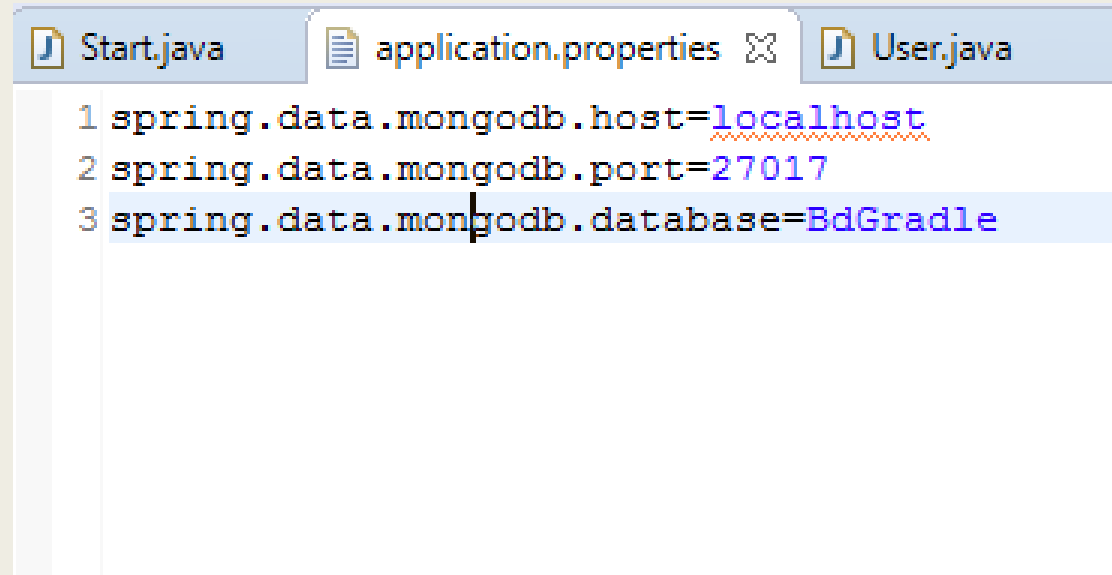
The screenshot shows an IDE window with the 'build.gradle' file open. The left sidebar displays a project structure with folders like 'src/main/java', 'src/main/resources', 'src/test/java', and 'JRE System Library [jre8]'. The main editor area shows the following code:

```
4 *
5 * This generated file contains a sample Java project to get you started.
6 * For more details take a look at the Java Quickstart chapter in the Gradle
7 * user guide available at https://docs.gradle.org/2.9/userguide/tutorial_java_projects.html
8 */
9
10 // Apply the java plugin to add support for Java
11 apply plugin: 'java'
12
13 // In this section you declare where to find the dependencies of your project
14 repositories {
15     // Use 'jcenter' for resolving your dependencies.
16     // You can declare any Maven/Ivy/file repository here.
17     jcenter()
18 }
19
20 // In this section you declare the dependencies for your production and test code
21 dependencies {
22     // The production code uses the SLF4J logging API at compile time
23     compile 'org.slf4j:slf4j-api:1.7.13'
24
25     // Declare the dependency for your favourite test framework you want to use in your tests.
26     // TestNG is also supported by the Gradle Test task. Just change the
27     // testCompile dependency to testCompile 'org.testng:testng:6.8.1' and add
28     // 'test.useTestNG()' to your build script.
29     testCompile 'junit:junit:4.12'
30     compile("org.springframework.boot:spring-boot-starter-web:2.0.0.RELEASE")
31     compile 'org.springframework.data:spring-data-mongodb:2.0.5.RELEASE'
32 }
```



# Configurando MongoDB

- Se crea un nuevo Source folder con la siguiente estructura src/main/resource y luego se crea un nuevo File con el nombre application.properties.



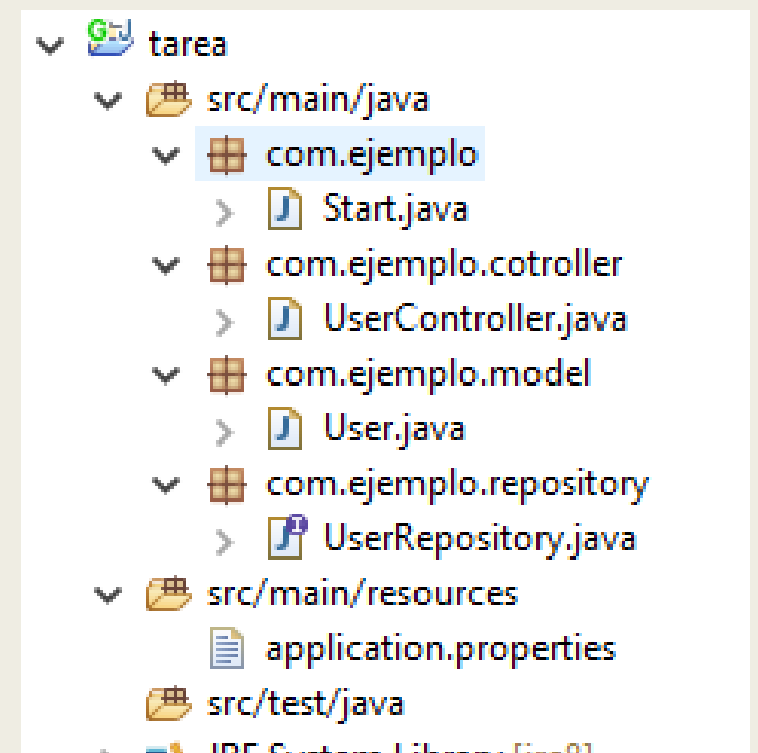
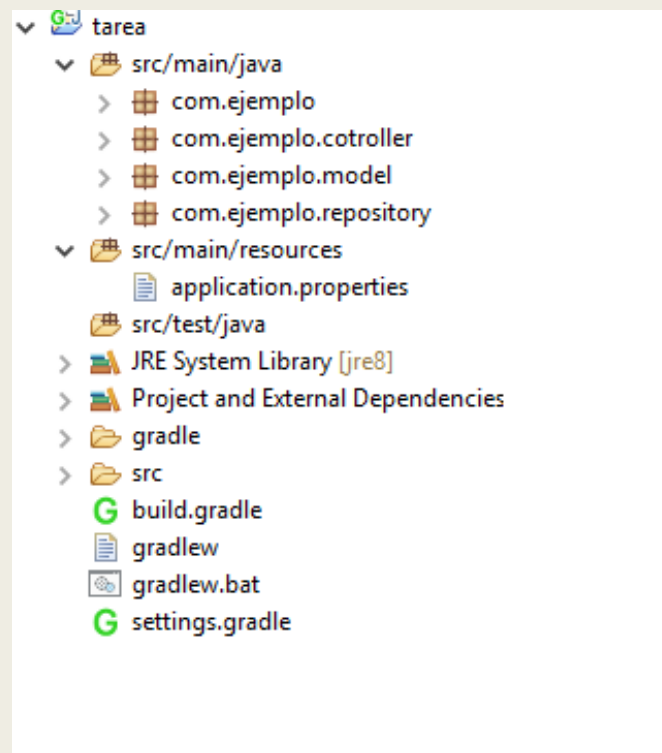
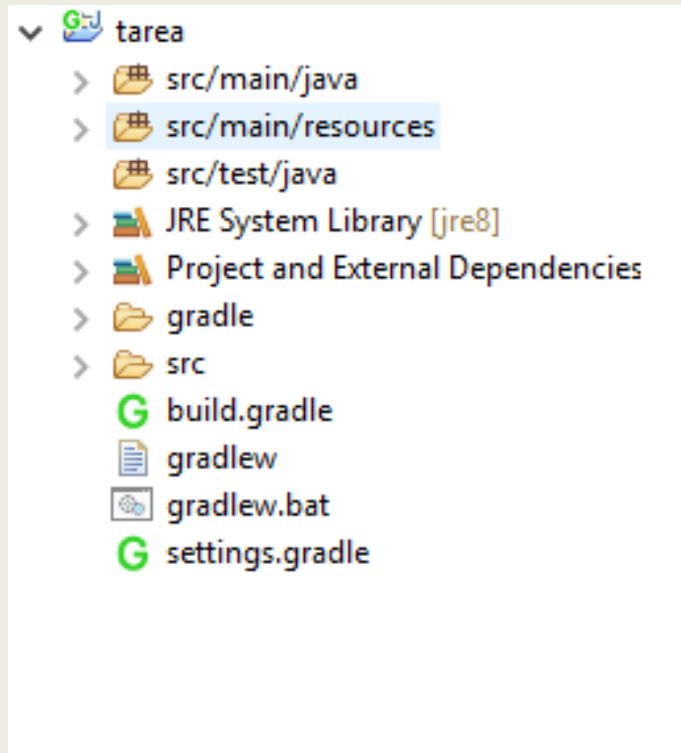
The screenshot shows an IDE window with three tabs: 'Start.java', 'application.properties', and 'User.java'. The 'application.properties' tab is active, displaying the following configuration:

```
1 spring.data.mongodb.host=localhost
2 spring.data.mongodb.port=27017
3 spring.data.mongodb.database=BdGradle
```

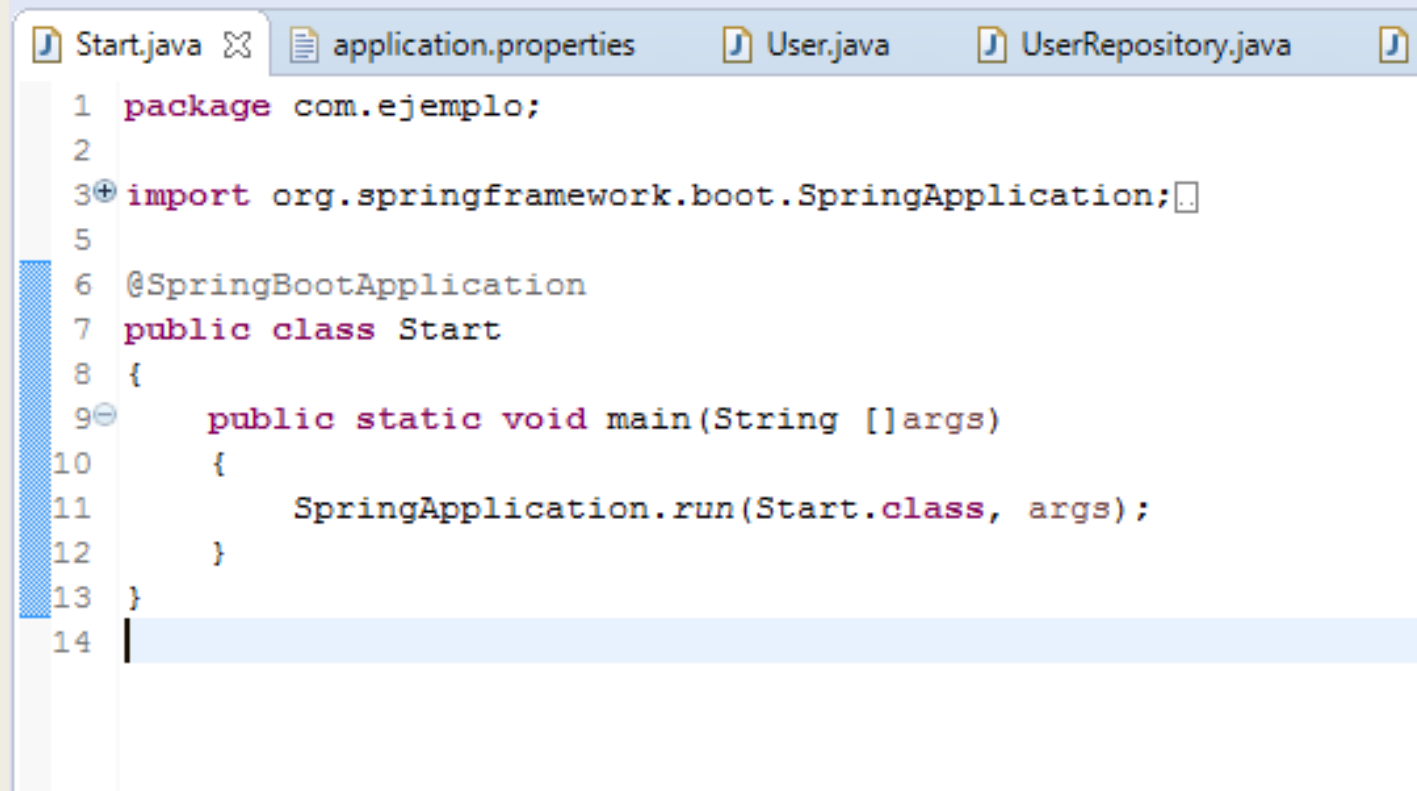


# Creación de paquetes.

- Tomar en cuenta que el proyecto deberá tener un paquete principal y luego de este se desprenden los demás.



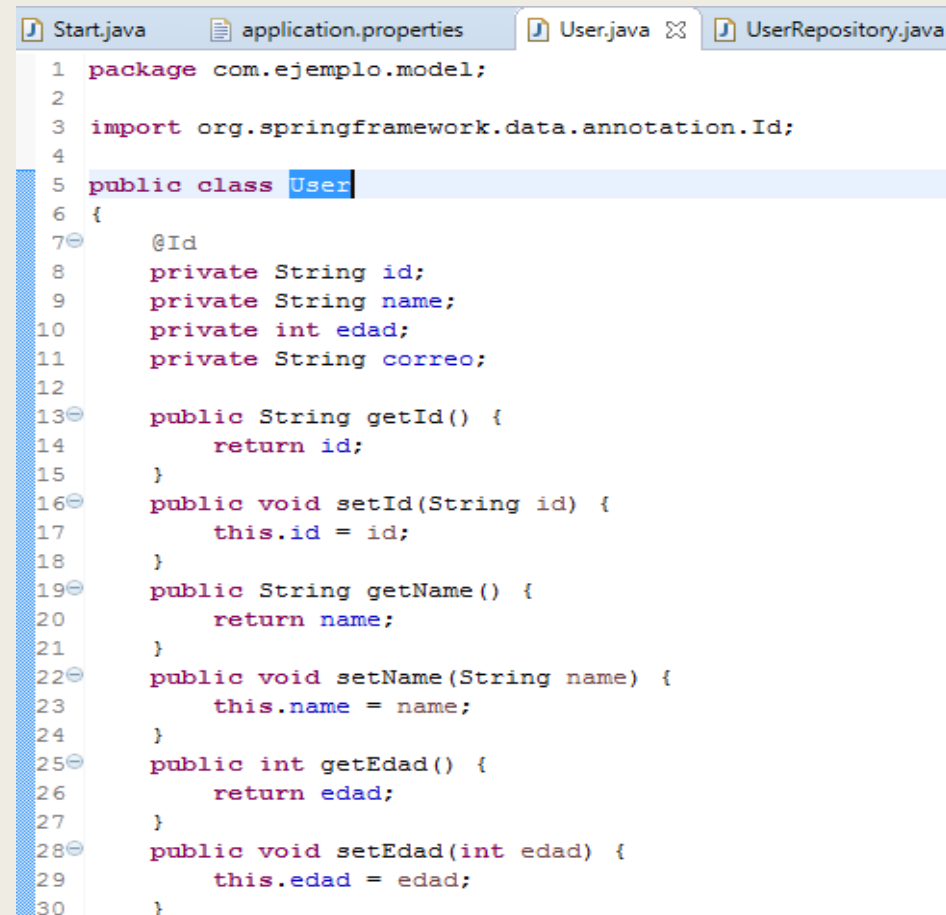
# Creación de un programa principal.



The screenshot shows an IDE window with several tabs: Start.java, application.properties, User.java, and UserRepository.java. The Start.java tab is active, displaying the following Java code:

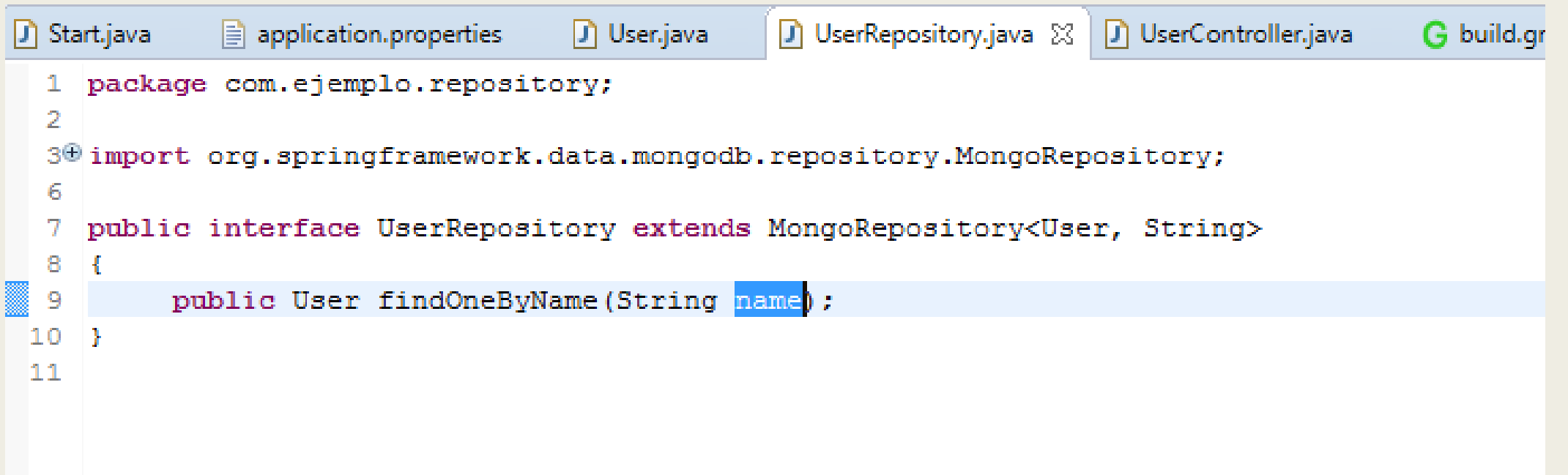
```
1 package com.ejemplo;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 @SpringBootApplication
7 public class Start
8 {
9     public static void main(String []args)
10     {
11         SpringApplication.run(Start.class, args);
12     }
13 }
14
```

# Creación de una clase de modelo de usuario con los campos necesarios para las operaciones de usuario con setters y getters.



```
Start.java  application.properties  User.java  UserRepository.java
1 package com.ejemplo.model;
2
3 import org.springframework.data.annotation.Id;
4
5 public class User
6 {
7     @Id
8     private String id;
9     private String name;
10    private int edad;
11    private String correo;
12
13    public String getId() {
14        return id;
15    }
16    public void setId(String id) {
17        this.id = id;
18    }
19    public String getName() {
20        return name;
21    }
22    public void setName(String name) {
23        this.name = name;
24    }
25    public int getEdad() {
26        return edad;
27    }
28    public void setEdad(int edad) {
29        this.edad = edad;
30    }
}
```

# Creación de un UserRepository Interface



The screenshot shows an IDE with several tabs: Start.java, application.properties, User.java, UserRepository.java (active), UserController.java, and build.gr. The active tab displays the following Java code:

```
1 package com.ejemplo.repository;
2
3+ import org.springframework.data.mongodb.repository.MongoRepository;
4
5
6
7 public interface UserRepository extends MongoRepository<User, String>
8 {
9     public User findOneByName(String name);
10 }
11
```

# Creación del controlador con métodos para comunicarse con el MongoDB

```
Start.java  application.properties  User.java  UserRepository.java  UserController.java  build.gradle

1 package com.ejemplo.cotroller;
2
3 import java.util.Optional;
4
15
16 @RestController
17 @RequestMapping("/user")
18 public class UserController
19 {
20     @Autowired
21     UserRepository userRepository;
22
23     //Creacion
24     @RequestMapping(method = RequestMethod.POST, consumes = MediaType.APPLICATION_JSON_VALUE)
25     public void crearPersona(@RequestBody User user)
26     {
27         userRepository.save(user);
28     }
29
30     //Lectura
31     @RequestMapping(value =("/{id}", method = RequestMethod.GET )
32     public Optional<User> leerPersona(@PathVariable String id)
33     {
34         return userRepository.findById(id);
35     }
36
37     //Actualizacion
38     @RequestMapping(method = RequestMethod.PUT, consumes = MediaType.APPLICATION_JSON_VALUE)
39     public void actualizarPersona(@RequestBody User user)
40     {
41         userRepository.save(user);
42     }
43 }
```

# Ejecutar como un Java application la clase Start.java

```
.  
/\ /_____, _ _ _ _ _ _ _ _ _ \\\ \\ \\ \\  
( )\___|'_||'_|_|\__/_\ |\\\ \\ \\ \\  
\_\ ___)|_)|||_|_|_|(|_|) ))))  
' |_| ._|_|_|_|_|_\__, ||/// ///  
=====|_|=====|_/=/_/_/_/  
  
:: Spring Boot ::                (v2.0.0.RELEASE)  
  
2018-03-16 17:10:11.502 INFO 3132 --- [main] com.ejemplo.Start  
2018-03-16 17:10:11.510 INFO 3132 --- [main] com.ejemplo.Start
```



```
main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext
main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user],methods=[PUT],consumes=[application/json]}" onto public void com.ejemplo.modelo.UsuarioController.put(java.lang.Long,com.ejemplo.modelo.Usuario)
main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user/{id}],methods=[GET]}" onto public java.util.Optional<com.ejemplo.modelo.Usuario> com.ejemplo.modelo.UsuarioController.findById(java.lang.Long)
main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user],methods=[POST],consumes=[application/json]}" onto public void com.ejemplo.modelo.UsuarioController.create(com.ejemplo.modelo.Usuario)
main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user/{id}],methods=[DELETE]}" onto public void com.ejemplo.modelo.UsuarioController.delete(java.lang.Long)
main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<String,Object>> com.ejemplo.modelo.UsuarioController.handleException(org.springframework.http.ResponseEntity<?>)
main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView com.ejemplo.modelo.UsuarioController.handleException(org.springframework.http.ResponseEntity<?>)
main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHandlerMethod]
main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHandlerMethod]
main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHandlerMethod]
main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
main] com.ejemplo.Start : Started Start in 15.545 seconds (JVM running for 16.567)
```



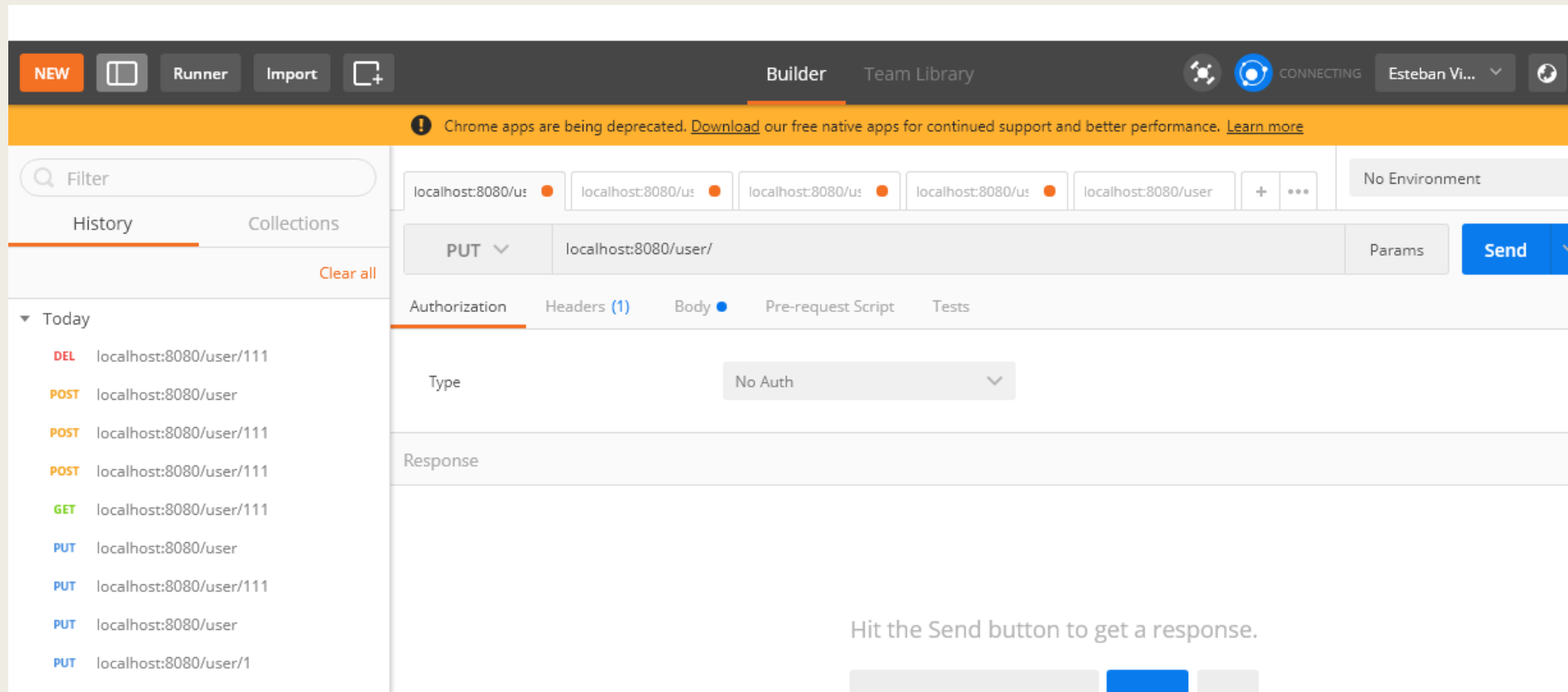
# Comandos básicos del MongoDB

- Creación o ingreso a una base: **use abc**
- Listar la bd: **show dbs**
- Listar las colecciones: **show collections**
- Consultas: **db.user.find()** o **db.user.find().pretty()** para más detalles.

Tener a consideración que para abrir el MongoDB (consola), se debe arrancar el servidor y luego el cliente.

 mongod	22/2/2018 20:00	Aplicación	30.247 KB
 mongo	22/2/2018 19:55	Aplicación	13.789 KB

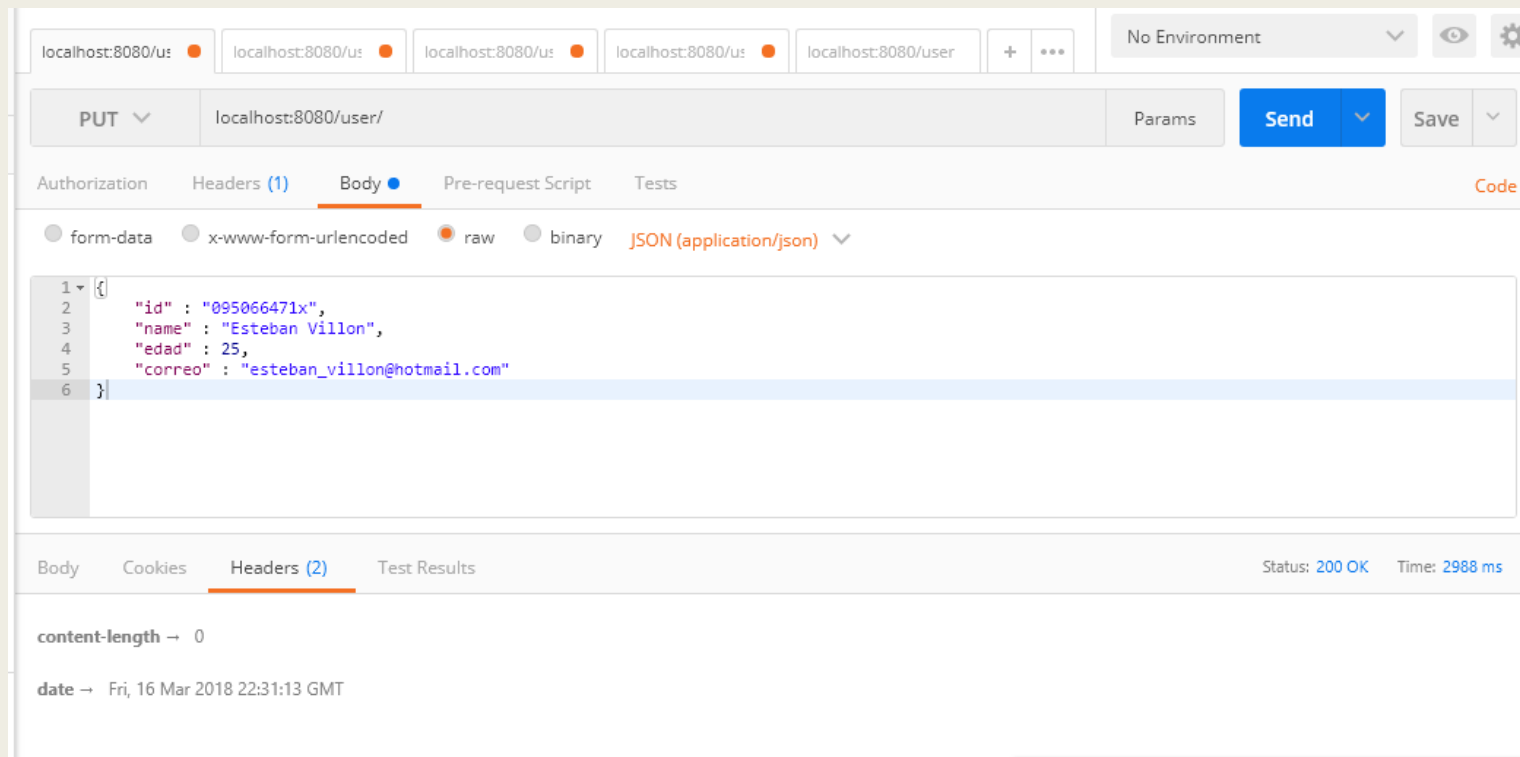
# Probando el proyecto previamente levantado (Postman).



# Comprobación que no exista la base.

```
> show dbs
admin      0.000GB
bigData    0.000GB
config     0.000GB
local      0.000GB
test       0.000GB
>
```

# Inserción (Postman)



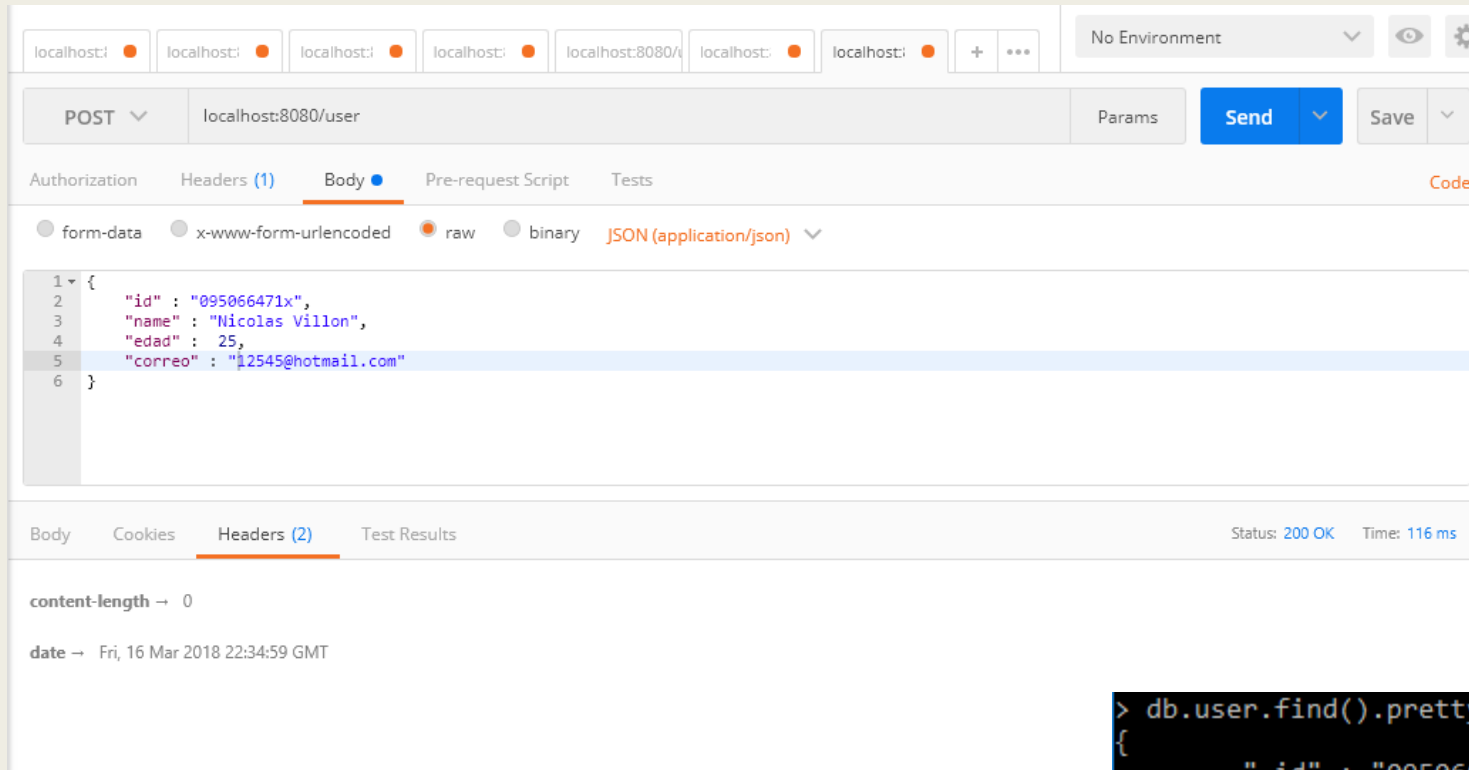
```
> show dbs
BdGradle 0.000GB
admin     0.000GB
bigData   0.000GB
config    0.000GB
local     0.000GB
test      0.000GB
> use BdGradle
switched to db BdGradle
> db.user.find()
{ "_id" : "095066471x", "name" : "Esteban Villon", "edad" : 25, "correo" : "esteban_villon@hotmail.com", "_class" : "com.ejemplo.model.User" }
>
```

# Consulta (Postman)

The screenshot displays the Postman application interface. At the top, there is a row of environment variables, each labeled 'localhost:808' with a red status indicator, followed by a '+ ...' button and a 'No Environment' dropdown menu with an eye icon and a settings gear icon. Below this, the request method is set to 'GET' and the URL is 'localhost:8080/user/095066471x'. To the right of the URL are buttons for 'Params', 'Send' (in blue), and 'Save'. A tab bar below the URL shows 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests', with 'Authorization' currently selected. Under the 'Authorization' tab, the 'Type' is set to 'No Auth'. Below the tab bar, another set of tabs shows 'Body', 'Cookies', 'Headers (3)', and 'Test Results', with 'Body' selected. To the right of these tabs, the status is '200 OK' and the time is '638 ms'. The 'Body' tab displays the response in 'JSON' format, with options for 'Pretty', 'Raw', and 'Preview'. The JSON response is as follows:

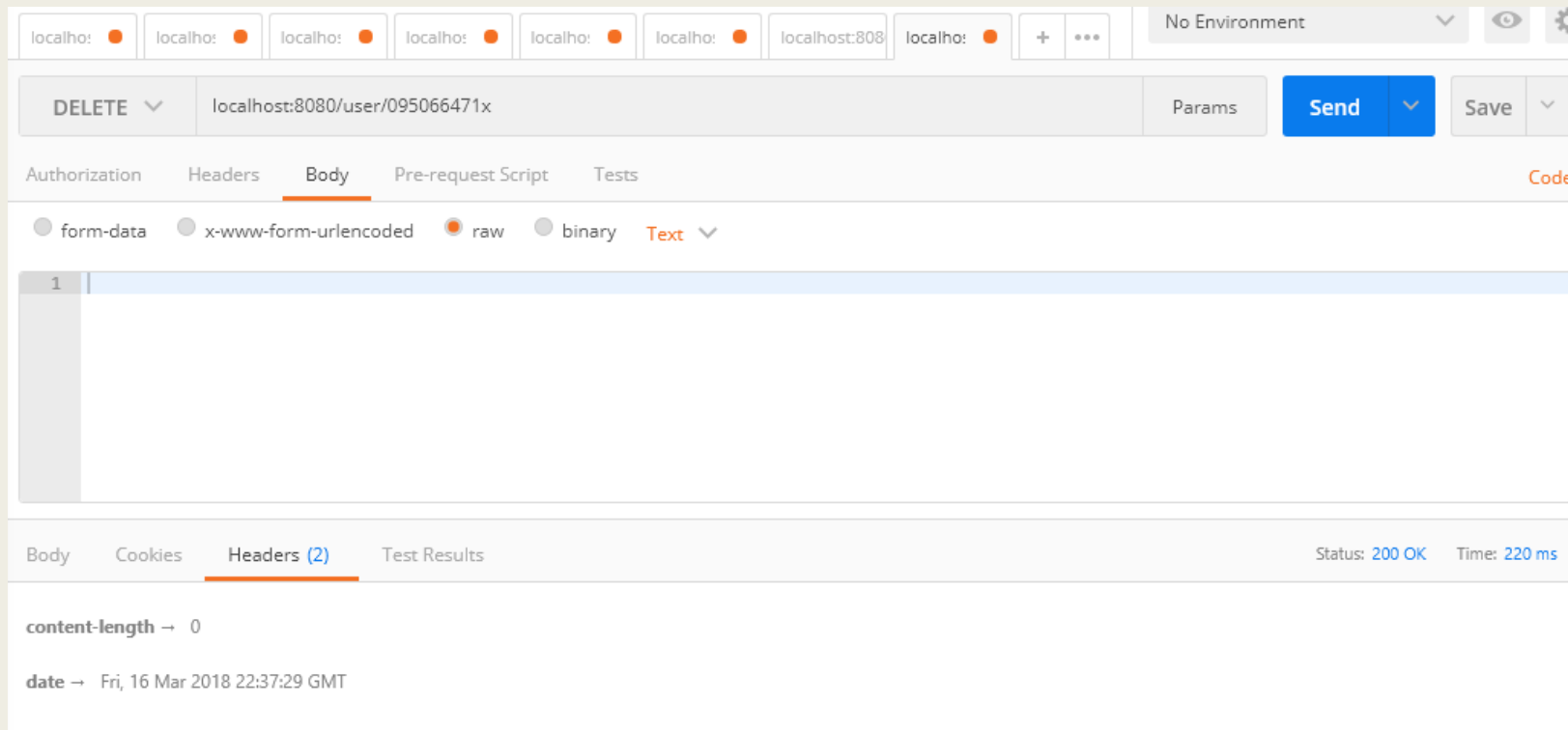
```
1 {
2   "id": "095066471x",
3   "name": "Esteban Villon",
4   "edad": 25,
5   "correo": "esteban_villon@hotmail.com"
6 }
```

# Actualización (Postman)



```
> db.user.find().pretty()  
{  
  "_id" : "095066471x",  
  "name" : "Nicolas Villon",  
  "edad" : 25,  
  "correo" : "12545@hotmail.com",  
  "_class" : "com.ejemplo.model.User"  
}  
>
```

# Borrar (Postman)



```
> db.user.find().pretty()  
> db.user.find({"_id": "095066471x"})  
>
```