SPRING BOOT + MONGODB

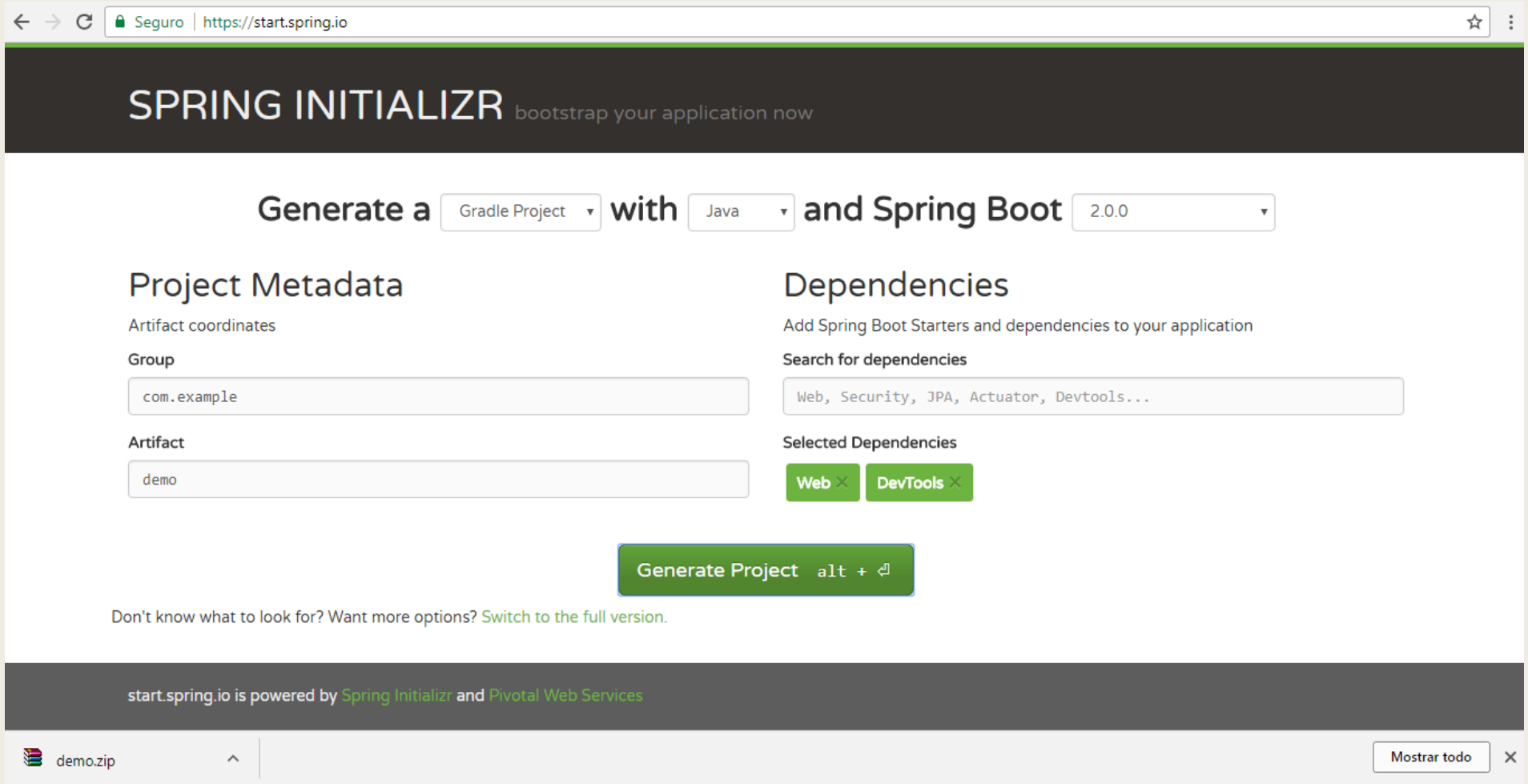
ESTUDIANTE: NICOLAS VILLÓN

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.



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)

Java - tarea/src/main/java/com/ejemplo/cotroller/UserController.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java

Package Explorer

- holamundo
- proyecto
- tarea
  - src/main/java
  - src/main/resources
  - src/test/java
  - JRE System Library [jre8]
  - Project and External Depend
  - gradle
  - src
  - build.gradle
  - gradlew
  - gradlew.bat
  - settings.gradle

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

```
1  package com.ejemplo.cotroller;
2
3  import java.util.Optional;
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 )
```

Problems | Javadoc | Declaration | Console | Progress | Gradle Tasks | Gradle Executions

SPRING-DATA-MONGODB

Spring Data MongoDB | Spring Boot | Spring Initializr

Seguro | https://projects.spring.io/spring-data-mongodb/

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

Quick Start

Download | 2.0.5 CURRENT GA | 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

Spring Boot | Spring Initializr

s://projects.spring.io/spring-boot/

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.

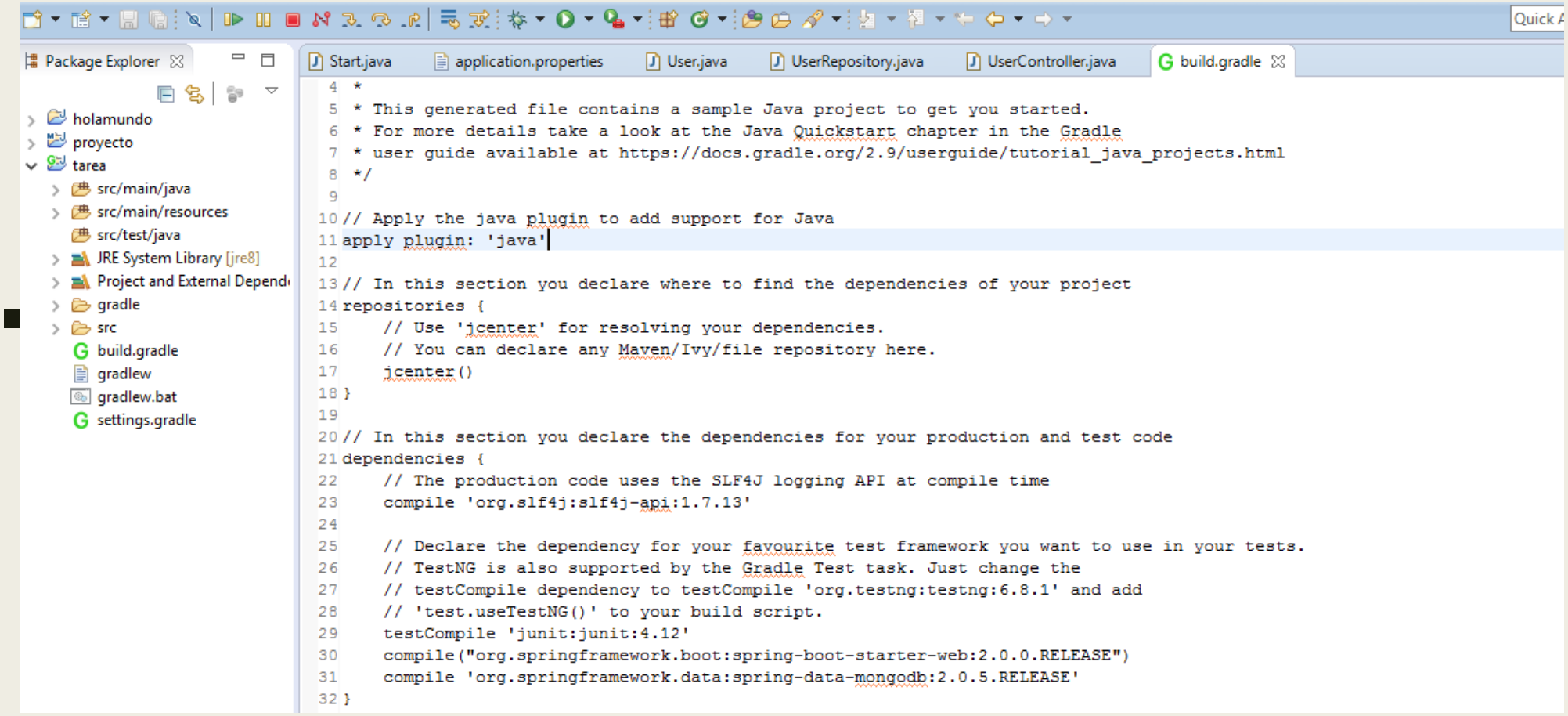
Download | 2.0.0 CURRENT GA | MAVEN | GRADLE

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.



Configurando MongoDB

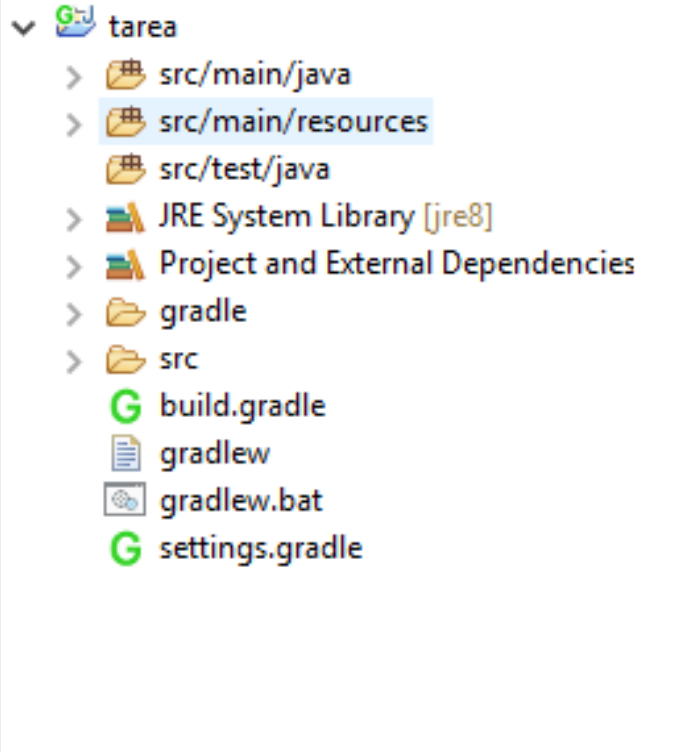
- Se crea un nuevo Source folder con la siguiente escructura src/main/resource y luego se crea un nuevo File con el nombre application.properties.

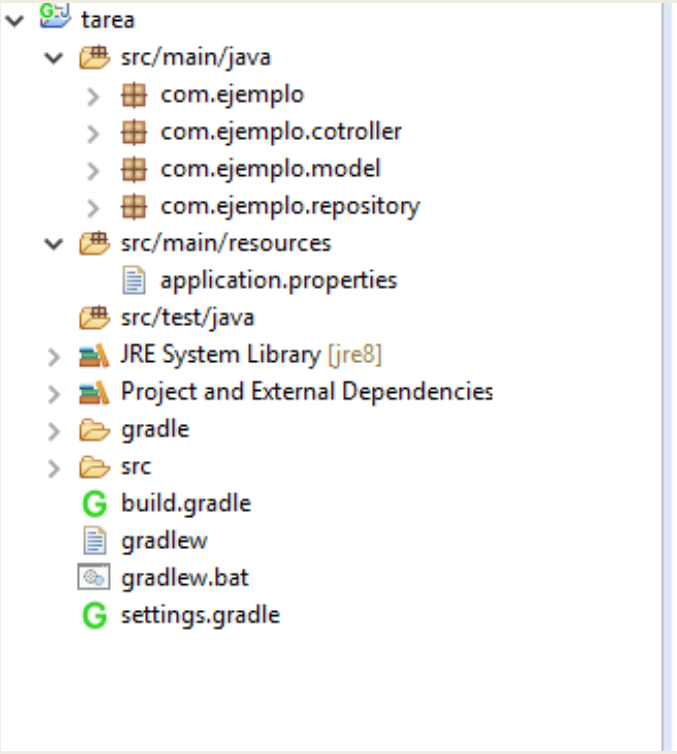
Start.java | application.properties | User.java

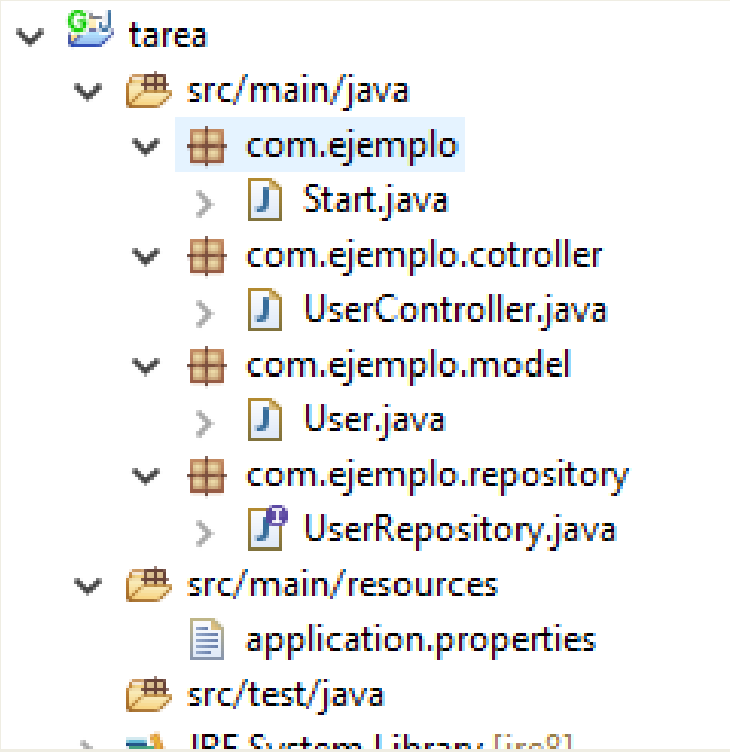
```
1 spring.data.mongodb.host=localhost
2 spring.data.mongodb.port=27017
3 spring.data.mongodb.database=BdGradle
```

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.

Start.java | application.properties | User.java | UserRepository.java

```
1  package com.ejemplo;
2
3  import org.springframework.boot.SpringApplication;
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

```java
package com.ejemplo.model;

import org.springframework.data.annotation.Id;

public class User
{
    @Id
    private String id;
    private String name;
    private int edad;
    private String correo;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
```

Creación de un UserRepository Interface

Start.java | application.properties | User.java | UserRepository.java | UserController.java | build.gr

```
1  package com.ejemplo.repository;
2
3⊕ import org.springframework.data.mongodb.repository.MongoRepository;
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

```java
package com.ejemplo.cotroller;

import java.util.Optional;

@RestController
@RequestMapping("/user")
public class UserController
{
    @Autowired
    UserRepository userRepository;

    //Creacion
    @RequestMapping(method = RequestMethod.POST, consumes = MediaType.APPLICATION_JSON_VALUE)
    public void crearPersona(@RequestBody User user)
    {
        userRepository.save(user);
    }

    //Lectura
    @RequestMapping(value ="/{id}", method = RequestMethod.GET )
    public Optional<User> leerPersona(@PathVariable String id)
    {
        return userRepository.findById(id);
    }
    //Actualizacion
    @RequestMapping(method = RequestMethod.PUT, consumes = MediaType.APPLICATION_JSON_VALUE)
    public void actualizarPersona(@RequestBody User user)
    {
        userRepository.save(user);
    }
```

Ejecutar como un Java application la clase Start.java

Problems | Javadoc | Declaration | Console | Progress | Gradle Tasks | Gradle Executions

Start [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (16 de mar. de 2018 5:10:08 p. m.)

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: 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.Annotation
      main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user],methods=[PUT],consumes=[application/json]}" onto public void com.ejem
      main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user/{id}],methods=[GET]}" onto public java.util.Optional<com.ejemplo.model
      main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user],methods=[POST],consumes=[application/json]}" onto public void com.ejer
      main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user/{id}],methods=[DELETE]}" onto public void com.ejemplo.cotroller.UserCo
      main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<
      main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.M
      main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.serv
      main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.reso
      main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.
      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)
```

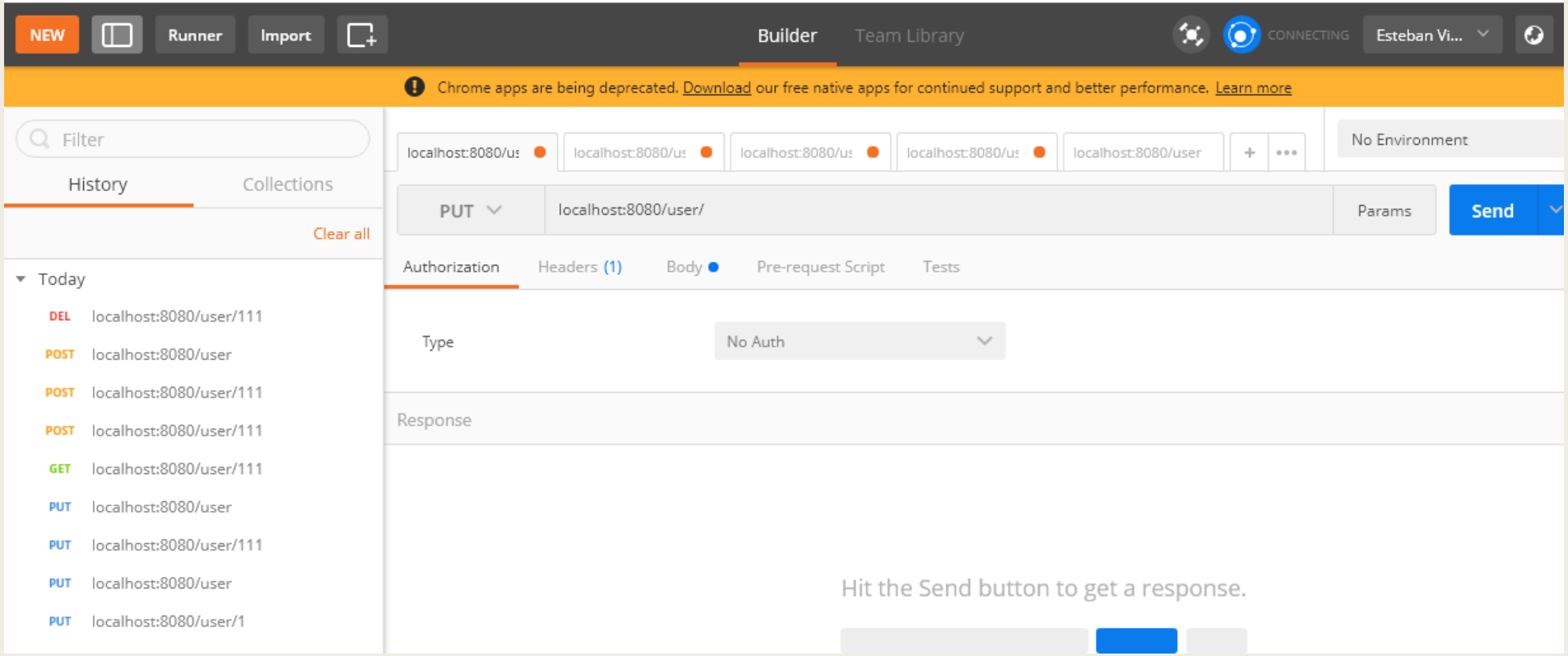
Comandos básicos del MongoDB

- 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

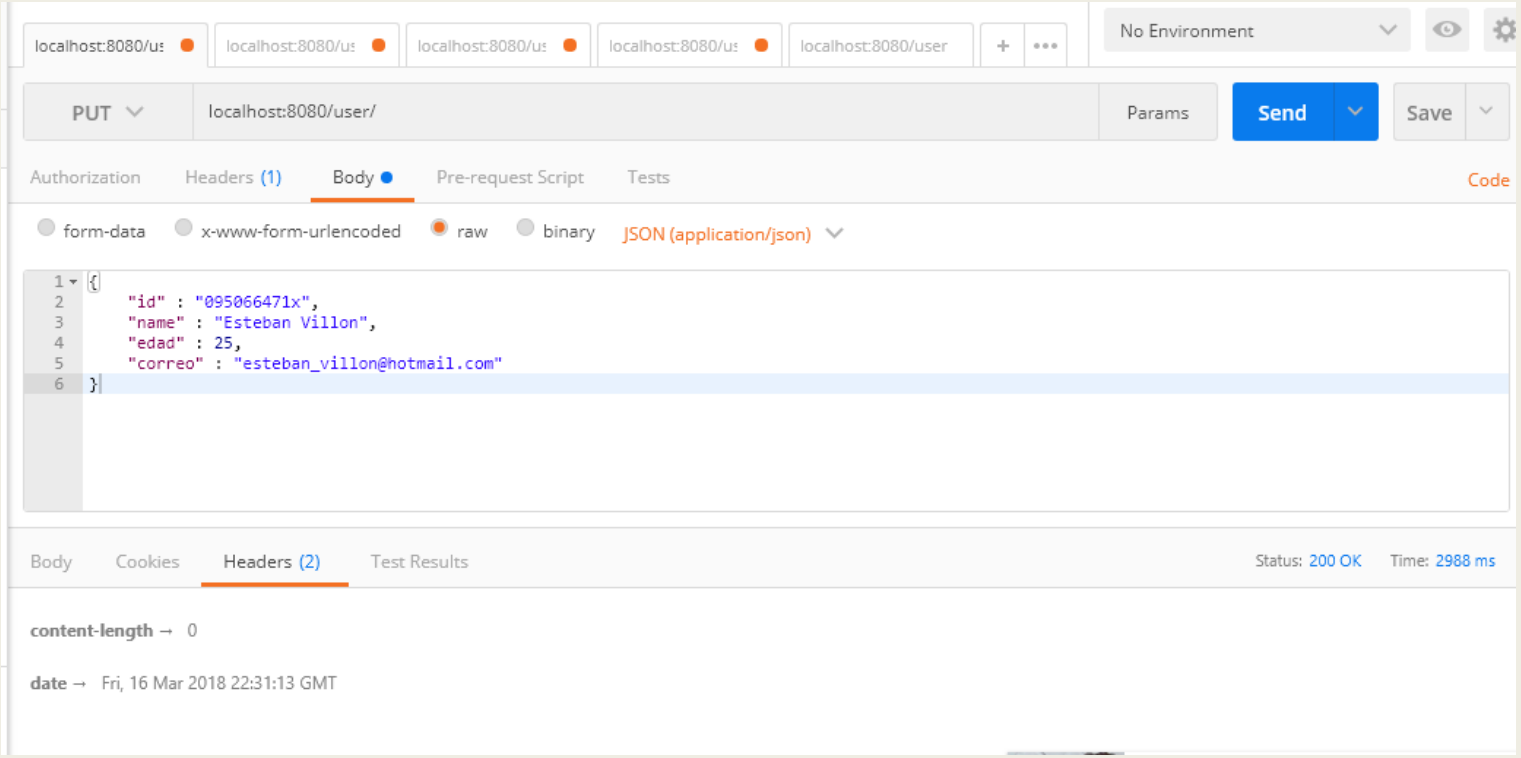
Probando el proyecto previamente levantado (Postman).



Comprobación que no exista la base.

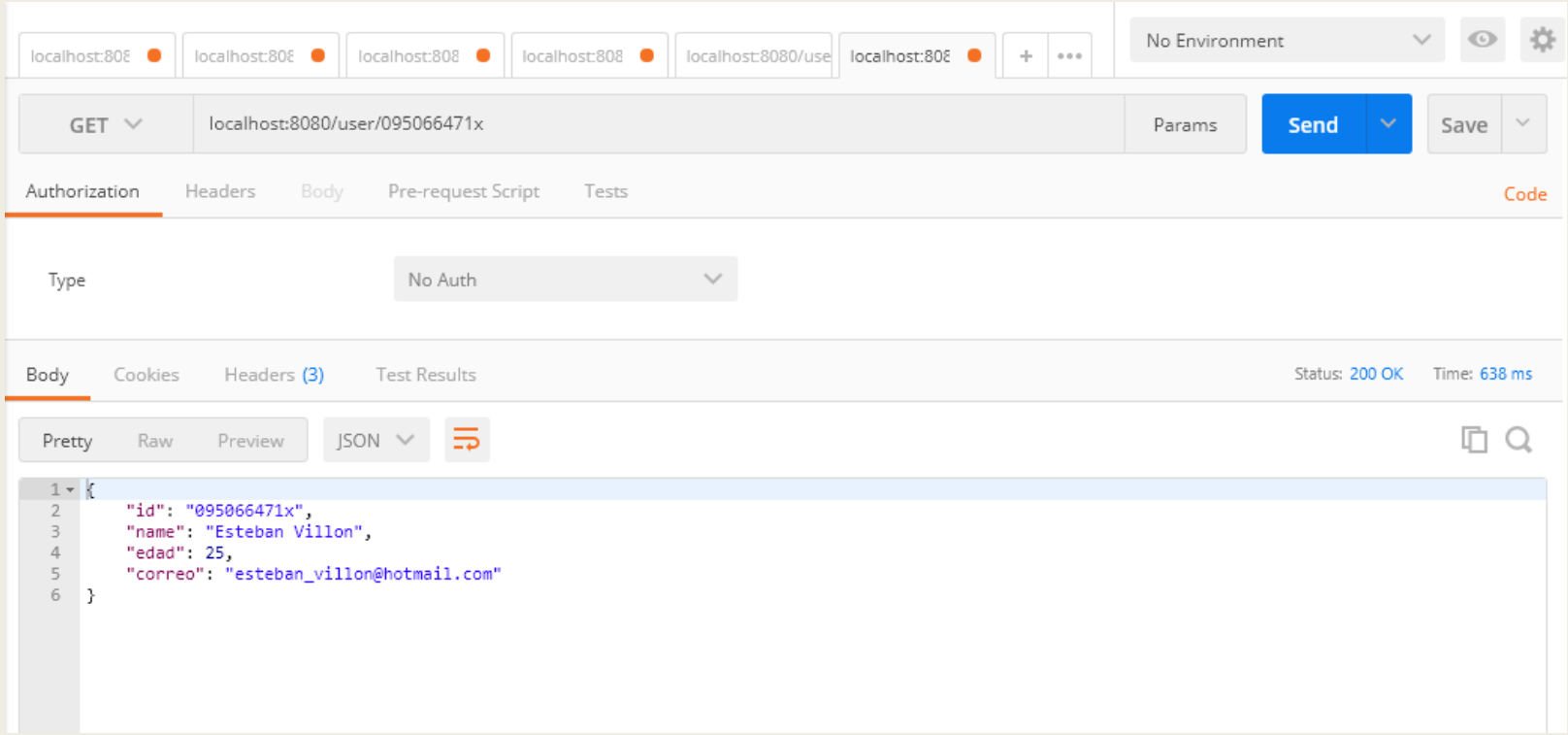
```
> show dbs
admin     0.000GB
bigData   0.000GB
config    0.000GB
local     0.000GB
test      0.000GB
>
```

Inserción (Postman)

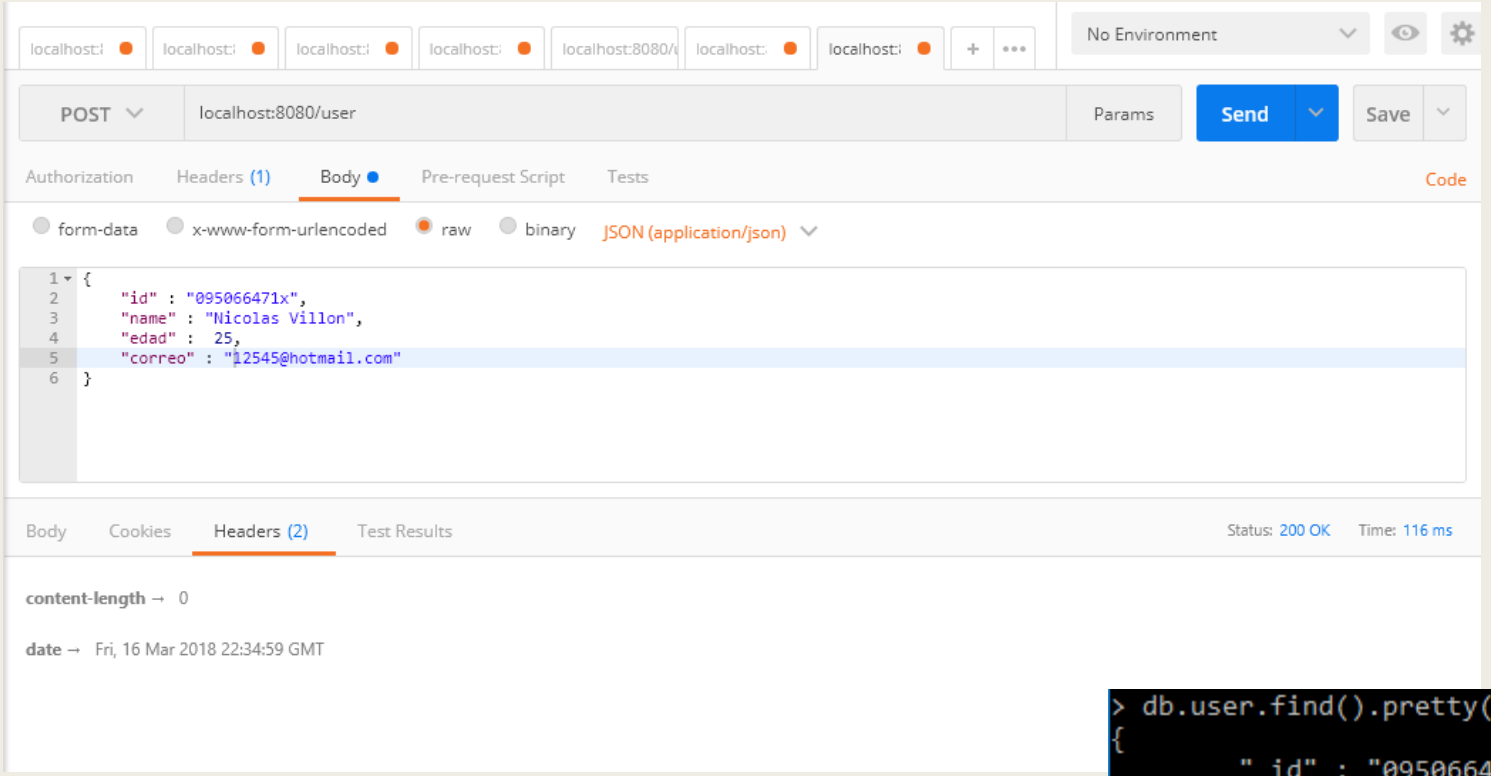



```
> show dbs
BdGradle  0.000GB
admin     0.000GB
bigData   0.000GB
config    0.000GB
local     0.000GB
test      0.000GB
> use BdGradle
switched to db BdGradle
> db.user.find()
{ "_id" : "095066471x", "name" : "Esteban Villon", "edad" : 25, "correo" : "esteban_villon@hotmail.com", "_class" : "com.ejemplo.model.User"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

Consulta (Postman)

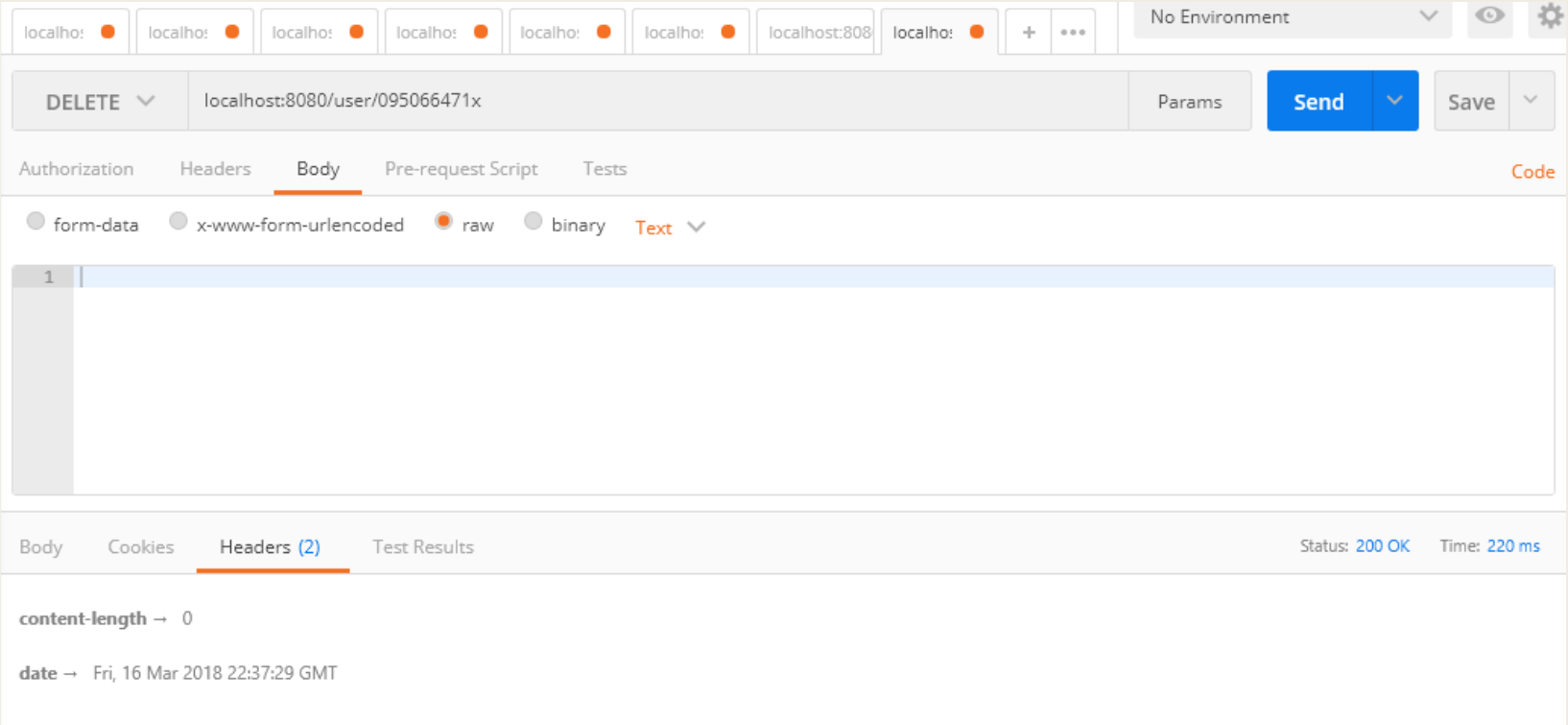


Actualización (Postman)




```
> 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"})
>
```