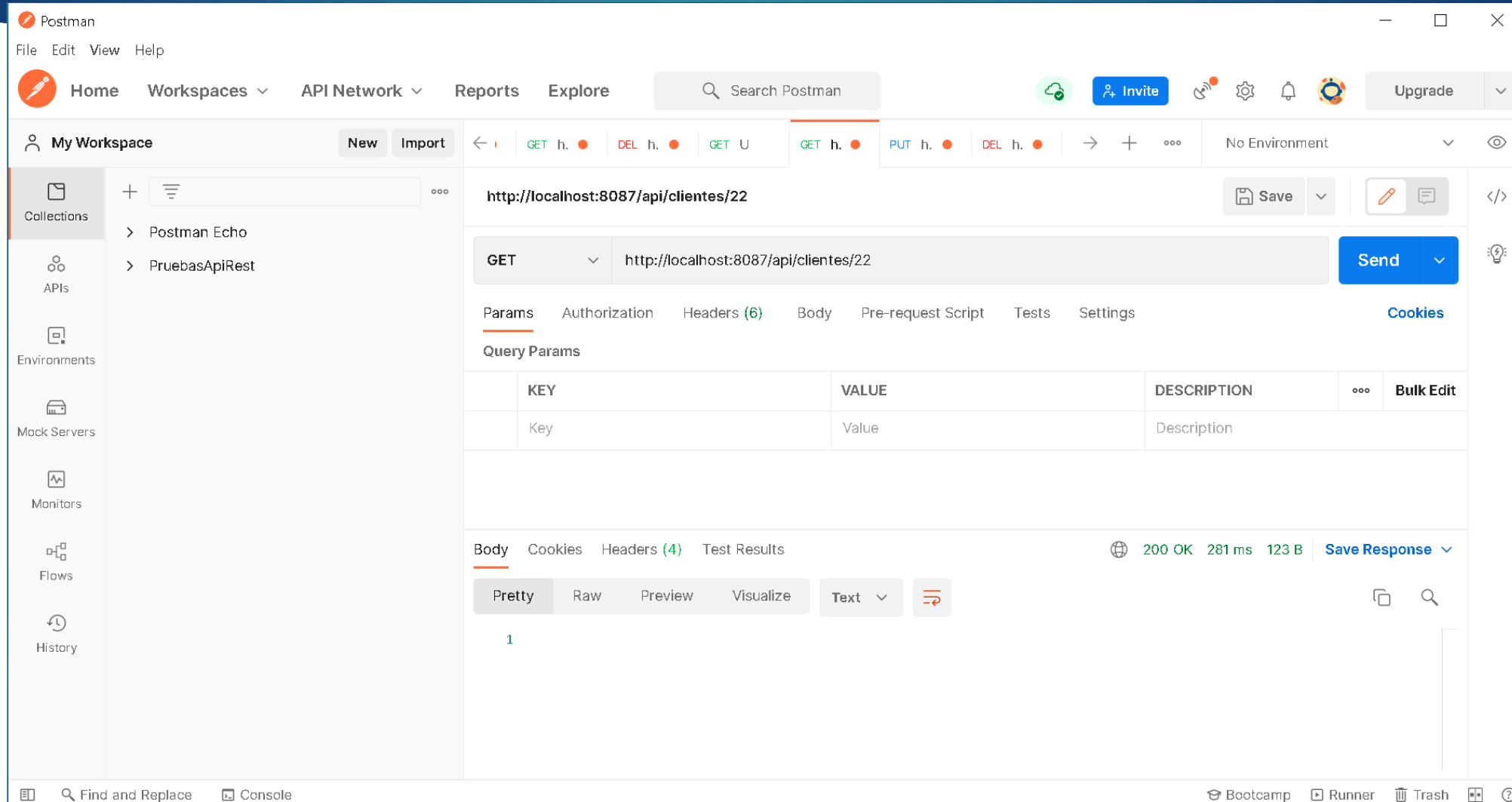


Manejo de Errores



Postman

File Edit View Help

Home Workspaces API Network Reports Explore Search Postman

My Workspace New Import

http://localhost:8087/api/clientes/22

GET http://localhost:8087/api/clientes/22

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (4) Test Results 200 OK 281 ms 123 B Save Response

Pretty Raw Preview Visualize Text

1

*ClienteRestController.java ×

```
36
37=  /*@GetMapping("/clientes/{id}")
38  public Cliente show(@PathVariable Long id) {
39      return clienteService.findById(id);
40  }*/|
41=  @GetMapping("/clientes/{id}")
42  public ResponseEntity<?> show(@PathVariable Long id){
43      Cliente cliente = null;
44      Map<String,Object> response = new HashMap<>();
45
46      try {
47          cliente = clienteService.findById(id);
48      } catch (DataAccessException e) {
49          response.put("mensaje", "Error al realizar consulta en base de datos");
50          response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
51          return new ResponseEntity<Map<String,Object>>(response,HttpStatus.INTERNAL_SERVER_ERROR);
52      }
53
54      if(cliente == null) {
55          response.put("mensaje", "El cliente ID: ".concat(id.toString()).concat("no existe en la base de datos"));
56          return new ResponseEntity<Map<String,Object>>(response,HttpStatus.NOT_FOUND);
57      }
58
59      return new ResponseEntity<Cliente>(cliente,HttpStatus.OK);
60  }
61
```

Postman

File Edit View Help

Home Workspaces API Network Reports Explore Search Postman

My Workspace New Import

Collections

- Postman Echo
- PruebasApiRest

APIs

Environments

Mock Servers

Monitors

Flows

History

http://localhost:8087/api/clientes/22

GET h. • DEL h. • GET U GET h. • PUT h. • DEL h. • No Environment

Save

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

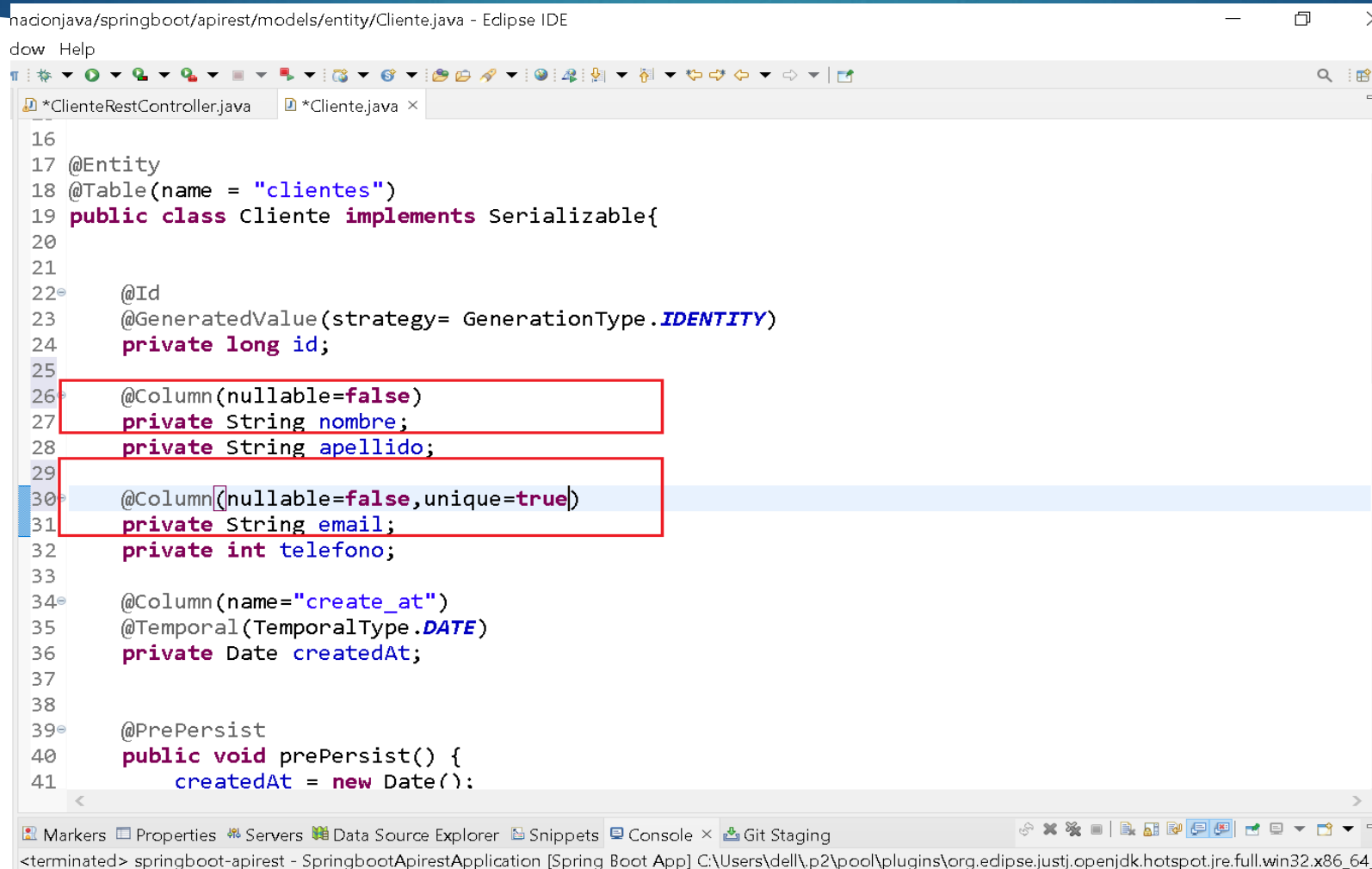
KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 404 Not Found 26 ms 231 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "mensaje": "El cliente ID: 22no existe en la base de datos"  
3 }
```

En nuestra entidad cliente programamos que el campo nombre no sea nulo, y el campo email no sea nulo y sea único



```
nadonjava/springboot/apirest/models/entity/Cliente.java - Eclipse IDE
dow Help

*ClienteRestController.java *Cliente.java x
16
17 @Entity
18 @Table(name = "clientes")
19 public class Cliente implements Serializable{
20
21
22     @Id
23     @GeneratedValue(strategy= GenerationType.IDENTITY)
24     private long id;
25
26     @Column(nullable=false)
27     private String nombre;
28     private String apellido;
29
30     @Column(nullable=false,unique=true)
31     private String email;
32     private int telefono;
33
34     @Column(name="create_at")
35     @Temporal(TemporalType.DATE)
36     private Date createdAt;
37
38
39     @PrePersist
40     public void prePersist() {
41         createdAt = new Date();
42     }
43 }
```

Markers Properties Servers Data Source Explorer Snippets Console x Git Staging

<terminated> springboot-apirest - SpringbootApirestApplication [Spring Boot App] C:\Users\del\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

Ahora agregamos a insertar

est/src/main/java/com/formacionjava/springboot/apirest/controllers/ClienteRestController.java - Eclipse IDE

Search Project Run Window Help

```
*ClienteRestController.java x
61
62
63= /*@PostMapping("/clientes")
64  @ResponseStatus(HttpStatus.CREATED)
65  public Cliente create(@RequestBody Cliente cliente) {
66      return clienteService.save(cliente);
67  }*/
68
69= @PostMapping("/clientes")
70  public ResponseEntity<?> create(@RequestBody Cliente cliente){
71      Cliente clienteNew = null;
72      Map<String,Object> response = new HashMap<>();
73
74      try {
75          clienteNew = clienteService.save(cliente);
76      } catch (DataAccessException e) {
77          response.put("mensaje","Error al realizar insert en base de datos");
78          response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
79          return new ResponseEntity<Map<String,Object>>(response,HttpStatus.INTERNAL_SERVER_ERROR);
80      }
81
82      response.put("mensaje","El cliente ha sido creado con éxito!");
83      response.put("cliente", clienteNew);
84      return new ResponseEntity<Map<String,Object>>(response,HttpStatus.CREATED);
85  }
86
```

Comprobamos las excepciones

The screenshot displays the Postman application interface. At the top, there's a navigation bar with 'Reports' and 'Explore' tabs, a search bar labeled 'Search Postman', and several utility buttons like 'Invite', 'Upgrade', and a settings icon. Below this is a collection of request tabs, with the 'POST' tab for 'http://localhost:8087/api/clientes' being the active one. The main workspace is divided into two sections. The top section shows the request configuration: the method is 'POST', the URL is 'http://localhost:8087/api/clientes', and the body is set to 'JSON' with a sample payload:

```
{  "nombre": "Rolando",  "apellido": "Lopez",}
```

. The bottom section shows the response, with a status of '201 Created', a time of '362 ms', and a size of '370 B'. The response body is formatted as 'Pretty' JSON:

```
{  "cliente": {    "id": 11,    "nombre": "Rolando",    "apellido": "Lopez",    "email": "rlopez@hotmail.com",    "telefono": 6432332,    "createdAt": "2021-11-21T23:11:53.787+00:00"  },  "mensaje": "El cliente ha sido creado con éxito!"}
```

No nos deja registrar un usuario con un mismo email por la propiedad unique

http://localhost:8087/api/clientes

POST http://localhost:8087/api/clientes

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

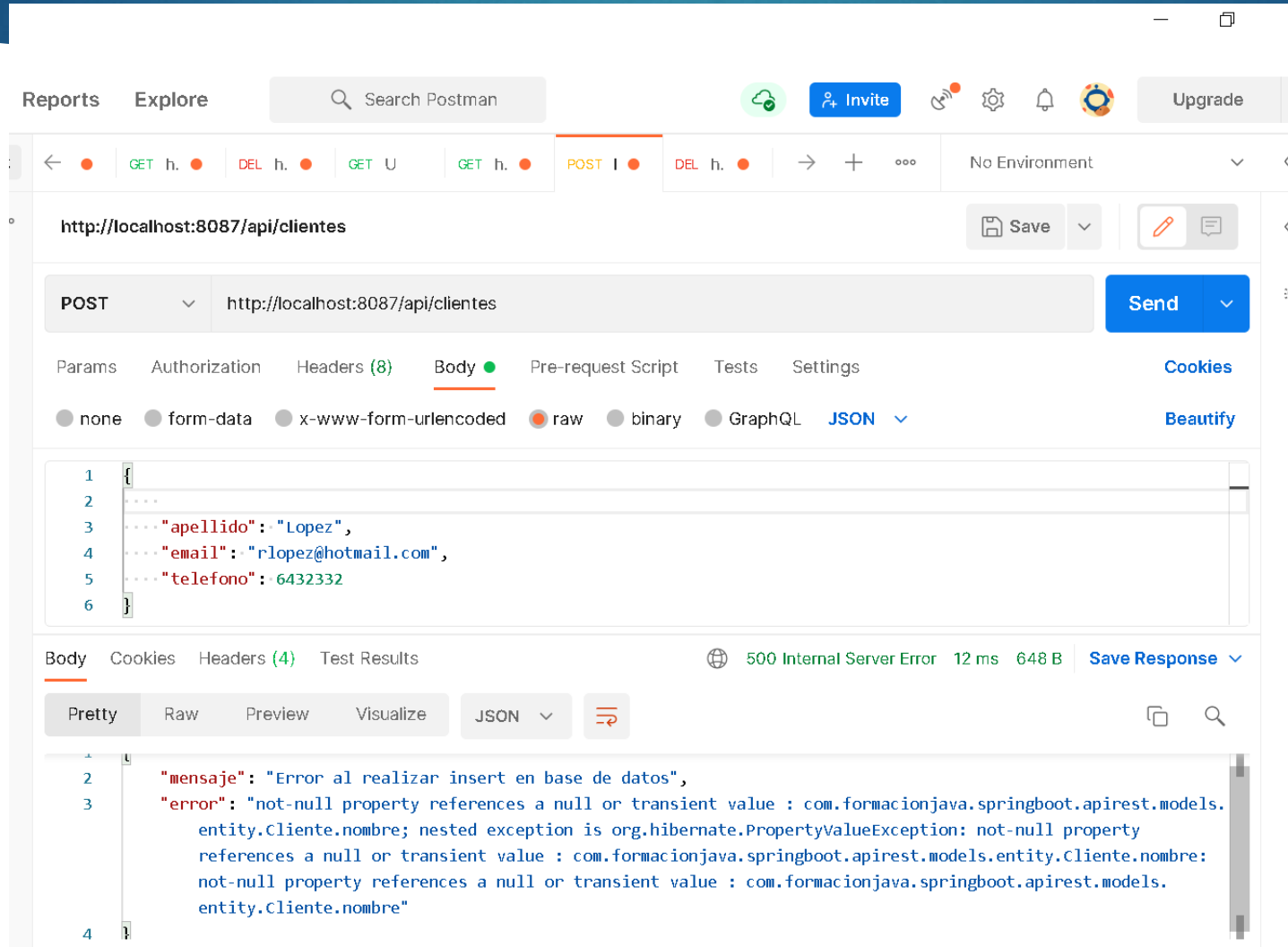
```
1 {
2   ...."nombre": "Rolando",
3   ...."apellido": "Lopez",
4   ...."email": "rlopez@hotmail.com",
5   ...."telefono": 6432332
6 }
```

Body Cookies Headers (4) Test Results 500 Internal Server Error 40 ms 499 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "mensaje": "Error al realizar insert en base de datos",
3   "error": "could not execute statement; SQL [n/a]; constraint [clientes.UK_1c96wv36rk2hwui7qhjks3mvg]; nested
4     exception is org.hibernate.exception.ConstraintViolationException: could not execute statement: Duplicate
      entry 'rlopez@hotmail.com' for key 'clientes.UK_1c96wv36rk2hwui7qhjks3mvg'"
}
```

Si no pasamos el campo nombre también nos informa que es requerido



También el campo email es requerido

The screenshot shows the Postman interface with a POST request to `http://localhost:8087/api/clientes`. The request body is a JSON object:

```
1 {
2   "nombre": "Rolando",
3   "apellido": "Lopez",
4   "telefono": 6432332
5 }
6
```

The response is a 500 Internal Server Error with the following message:

```
1 {
2   "mensaje": "Error al realizar insert en base de datos",
3   "error": "not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.email; nested exception is org.hibernate.PropertyValueException: not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.email:"
}
```

Ahora para el Update

ClienteRestController.java ×

```
97     */
98
99     @PutMapping("/clientes/{id}")
100     public ResponseEntity<?> update(@RequestBody Cliente cliente, @PathVariable Long id) {
101         Cliente clienteActual= clienteService.findById(id);
102
103         Cliente clienteUpdated = null;
104         Map<String,Object> response = new HashMap<>();
105
106         if(clienteActual == null){
107             response.put("mensaje", "Error: no se pudo editar, el cliente ID: ".concat(id.toString()).concat("no existe el id en la base de datos"));
108             return new ResponseEntity<Map<String,Object>>(response,HttpStatus.NOT_FOUND);
109         }
110
111         try {
112             clienteActual.setApellido(cliente.getApellido());
113             clienteActual.setNombre(cliente.getNombre());
114             clienteActual.setEmail(cliente.getEmail());
115             clienteActual.setCreatedAt(cliente.getCreatedAt());
116
117             clienteUpdated = clienteService.save(clienteActual);
118         } catch (DataAccessException e) {
119             response.put("mensaje", "Error al actualizar el cliente en base de datos");
120             response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
121             return new ResponseEntity<Map<String,Object>>(response,HttpStatus.INTERNAL_SERVER_ERROR);
122         }
123
124         response.put("mensaje","El cliente ha sido actualizado con éxito!");
125         response.put("cliente", clienteUpdated);
126
127         return new ResponseEntity<Map<String,Object>>(response, HttpStatus.CREATED);
128     }
```

Probamos con postman

The screenshot displays the Postman application interface. At the top, there's a navigation bar with 'Reports' and 'Explore' tabs, a search bar, and various utility buttons like 'Invite', 'Settings', 'Notifications', and 'Upgrade'. Below this, a request history bar shows several requests with their methods (GET, DEL, PUT) and status (h. ●). The main workspace is configured for a PUT request to 'http://localhost:8087/api/clientes/1'. The 'Body' tab is selected, showing a JSON payload. The response section at the bottom indicates a '201 Created' status with a response time of 70 ms and a body size of 376 B. The response body is displayed in a 'Pretty' JSON format.

Request Details:

- Method: PUT
- URL: http://localhost:8087/api/clientes/1
- Body Type: raw (JSON)
- Body Content:

```
1 {  
2   "cliente": {  
3     "id": 1,  
4     "nombre": "Cristiano",  
5     "apellido": "Ronaldo",  
6     "email": "cr7@hotmail.com",  
7     "telefono": 65433223,  
8     "createdAt": "2021-11-21T00:00:00.000+00:00"  
9   },  
10  "mensaje": "El cliente ha sido actualizado con éxito!"  
11 }
```

Response Details:

- Status: 201 Created
- Time: 70 ms
- Size: 376 B
- Save Response: [button]

No me deja actualizar un email ya registrado

The screenshot shows the Postman interface with a PUT request to `http://localhost:8087/api/clientes/1`. The request body is a JSON object:

```
{
  "nombre": "Cristiano",
  "apellido": "Ronaldo",
  "email": "rlopez@hotmail.com",
  "telefono": 64323322,
  "createdAt": "2021-11-21"
}
```

The response is a 500 Internal Server Error (41 ms, 505 B). The response body is shown in the 'Body' tab, formatted as JSON:

```
{
  "mensaje": "Error al actualizar el cliente en base de datos",
  "error": "could not execute statement; SQL [n/a]; constraint [clientes.UK_1c96wv36rk2hwui7qhjks3mvg]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement: Duplicate entry 'rlopez@hotmail.com' for key 'clientes.UK_1c96wv36rk2hwui7qhjks3mvg'"
}
```

No me deja actualizar sin el campo email

The screenshot shows the Postman interface with a PUT request to `http://localhost:8087/api/clientes/1`. The request body is a JSON object:

```
{
  "nombre": "Cristiano",
  "apellido": "Ronaldo",
  "telefono": 64323322,
  "createdAt": "2021-11-21"
}
```

The response is a 500 Internal Server Error with the following message:

```
{
  "mensaje": "Error al actualizar el cliente en base de datos",
  "error": "not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.email; nested exception is org.hibernate.PropertyValueException: not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.email: not-null property references a null or transient value : com.formacionjava.springboot."
}
```

The error message indicates that the `email` field is required but is null or transient.

No me deja actualizar sin el campo nombre

The screenshot shows the Postman interface with a PUT request to `http://localhost:8087/api/clientes/1`. The request body is a JSON object:

```
{
  "apellido": "Ronaldo",
  "email": "cr7@hotmail.com",
  "telefono": 64323322,
  "createdAt": "2021-11-21"
}
```

The response is a 500 Internal Server Error with a status of 17 ms and 654 B. The response body is a JSON object:






```
{
  "mensaje": "Error al actualizar el cliente en base de datos",
  "error": "not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.nombre; nested exception is org.hibernate.PropertyValueException: not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.nombre: not-null property references a null or transient value : com.formacionjava.springboot."
}
```

Manejo de error con Delete

```
*ClienteRestController.java x
126
127     return new ResponseEntity<Map<String, Object>>(response, HttpStatus.CREATED);
128 }
129 /*@DeleteMapping("clientes/{id}")
130 public void delete(@PathVariable Long id) {
131     clienteService.delete(id);
132 }*/
133
134 @DeleteMapping("clientes/{id}")
135 public ResponseEntity<?> delete(@PathVariable Long id) {
136     Map<String, Object> response = new HashMap<>();
137
138     try {
139
140         clienteService.delete(id);
141     } catch (DataAccessException e) {
142         response.put("mensaje", "Error al eliminar el cliente en la base de datos");
143         response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
144         return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
145     }
146     response.put("mensaje", "El cliente ha sido eliminado con éxito!");
147
148
149     return new ResponseEntity<Map<String, Object>>(response, HttpStatus.OK);
150 }
151 }
152
```

ReportsExplore

Search Postman

 Invite

Upgrade

←

POST

GET h.

DEL h.

GET h.



PUT h.

DEL h.

→ + ...

No Environment

Save


 

</>

DELETE

http://localhost:8087/api/clientes/1

Send



Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK 89 ms 218 B

Save Response



Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "mensaje": "El cliente ha sido eliminado con éxito!"
3 }
```


Si no encuentra un id valido

The screenshot shows a REST client interface with a DELETE request to `http://localhost:8087/api/clientes/20`. The response is a 500 Internal Server Error with a JSON body indicating that no client with id 20 exists.

Request:

- Method: DELETE
- URL: `http://localhost:8087/api/clientes/20`

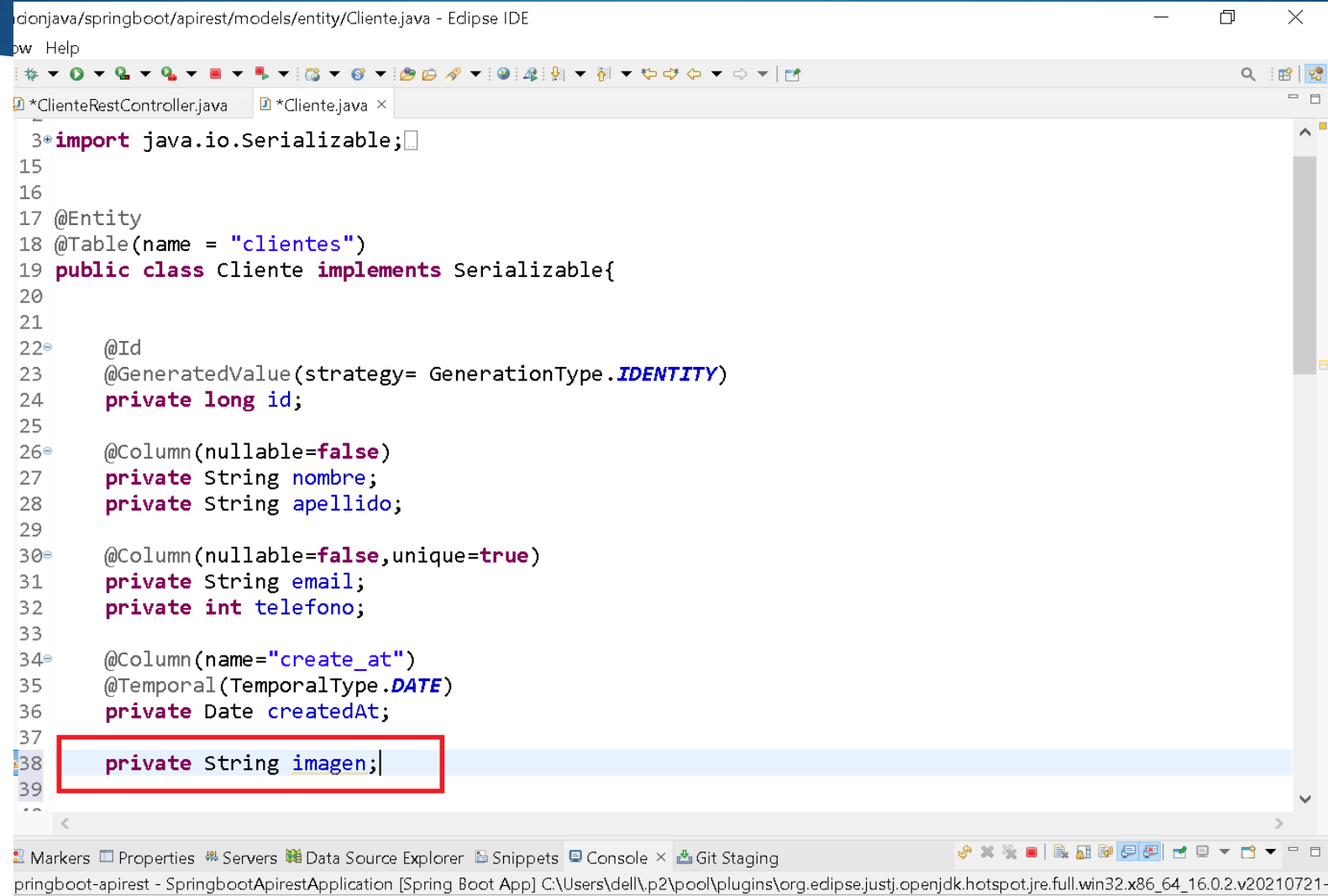
Response:

500 Internal Server Error 30 ms 415 B

Body (JSON):

```
{
  "mensaje": "Error al eliminar el cliente en la base de datos",
  "error": "No class com.formacionjava.springboot.apirest.models.entity.Cliente entity with id 20 exists!: No class com.formacionjava.springboot.apirest.models.entity.Cliente entity with id 20 exists!"
}
```

Crear una Api para subida de archivo, agregamos el siguiente campo y sus respectivos setter y getter



```
14 *import java.io.Serializable;
15
16
17 @Entity
18 @Table(name = "clientes")
19 public class Cliente implements Serializable{
20
21
22     @Id
23     @GeneratedValue(strategy= GenerationType.IDENTITY)
24     private long id;
25
26     @Column(nullable=false)
27     private String nombre;
28     private String apellido;
29
30     @Column(nullable=false,unique=true)
31     private String email;
32     private int telefono;
33
34     @Column(name="create_at")
35     @Temporal(TemporalType.DATE)
36     private Date createdAt;
37
38     private String imagen;
39
40 }
```

```

67 public void setEmail(String email) {
68     this.email = email;
69 }
70 public int getTelefono() {
71     return telefono;
72 }
73 public void setTelefono(int telefono) {
74     this.telefono = telefono;
75 }
76 public Date getCreatedAt() {
77     return createdAt;
78 }
79 public void setCreatedAt(Date createdAt) {
80     this.createdAt = createdAt;
81 }

```

```

82
83
84 public String getImagen() {
85     return imagen;
86 }
87
88 public void setImagen(String imagen) {
89     this.imagen = imagen;
90 }

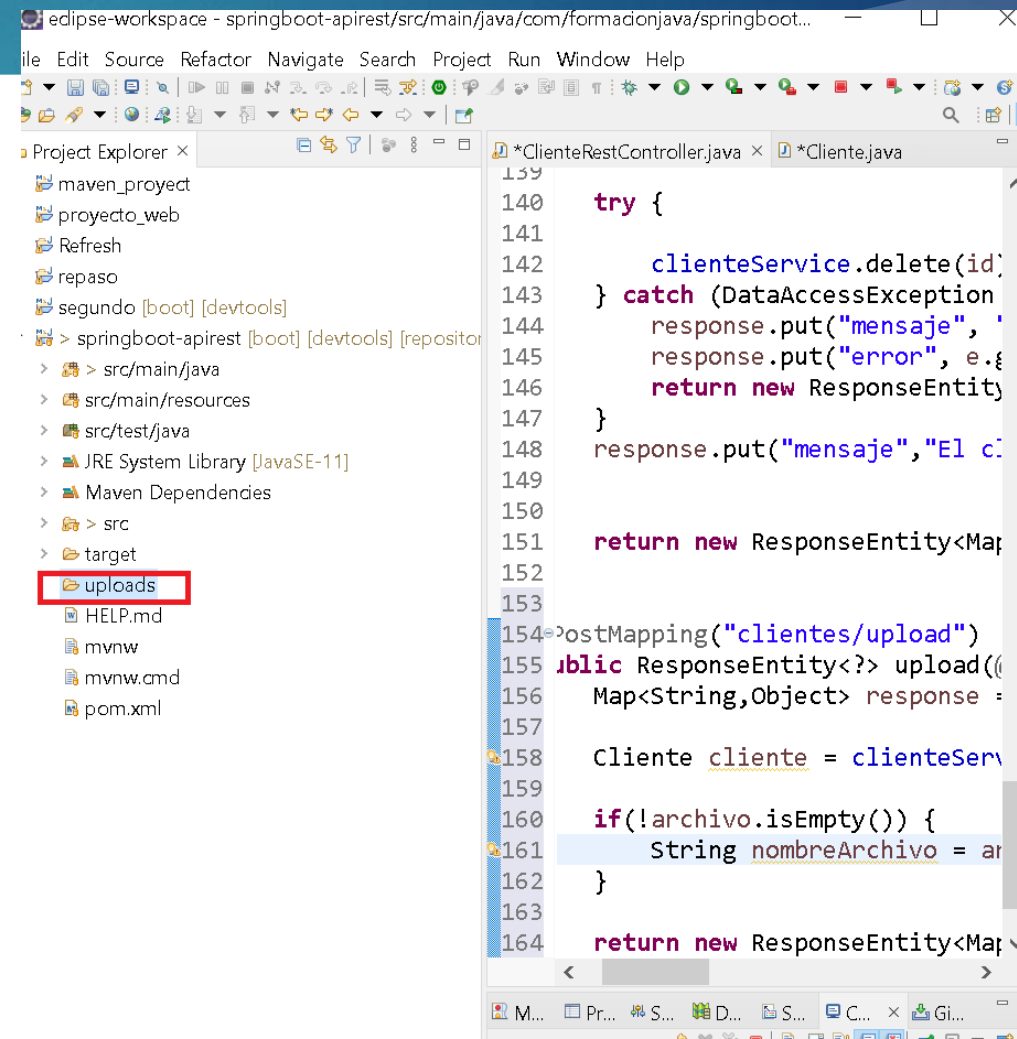
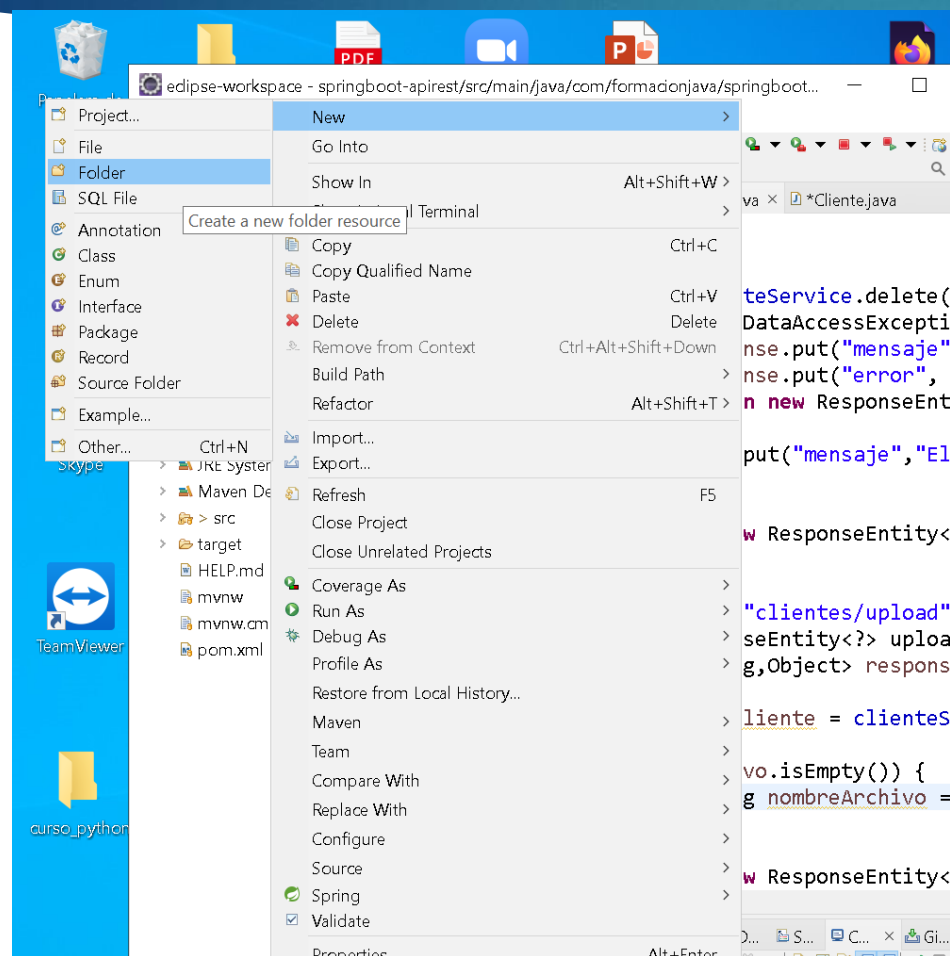
```

```

91
92
93 /**

```

Creamos una carpeta en el directorio principal del proyecto para alojar las imágenes



Agregamos un nuevo método al controlador

ClienteRestController.java x Cliente.java

```
157
158 @PostMapping("clientes/upload")
159 public ResponseEntity<?> upload(@RequestParam("archivo") MultipartFile archivo, @RequestParam("id") Long id){
160     Map<String,Object> response = new HashMap<>();
161
162     Cliente cliente = clienteService.findById(id);
163
164     if(!archivo.isEmpty()) {
165         String nombreArchivo = archivo.getOriginalFilename();
166         Path rutaArchivo = Paths.get("uploads").resolve(nombreArchivo).toAbsolutePath();
167
168         try {
169             Files.copy(archivo.getInputStream(), rutaArchivo);
170         } catch (IOException e) {
171             // TODO Auto-generated catch block
172             response.put("mensaje", "Error al subir la imagen del cliente");
173             response.put("error", e.getMessage().concat(": ").concat(e.getCause().getMessage()));
174             return new ResponseEntity<Map<String,Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
175         }
176
177         cliente.setImagen(nombreArchivo);
178
179         clienteService.save(cliente);
180
181         response.put("cliente", cliente);
182         response.put("mensaje", "Has subido correctamente la imagen: "+ nombreArchivo);
183     }
184
185     return new ResponseEntity<Map<String,Object>>(response, HttpStatus.CREATED);
186
187
188
```

Probamos la nueva api con postman

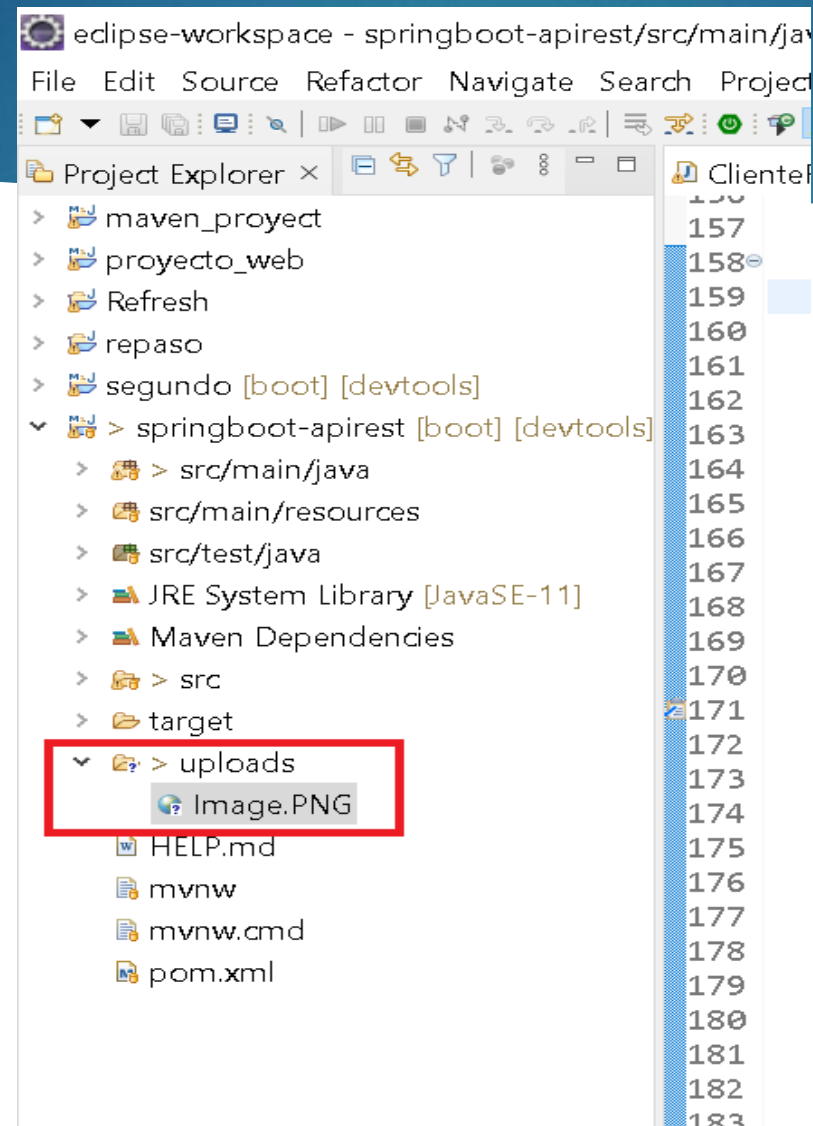
The screenshot shows the Postman interface for a POST request to `http://localhost:8087/api/clientes/upload/`. The request is configured with the following details:

- Method:** POST
- URL:** `http://localhost:8087/api/clientes/upload/`
- Body Type:** form-data
- Body Data:**

KEY	VALUE
archivo	Image.PNG
id	1

The response is displayed in the bottom pane, showing a successful status (201 Created) and the following JSON body:

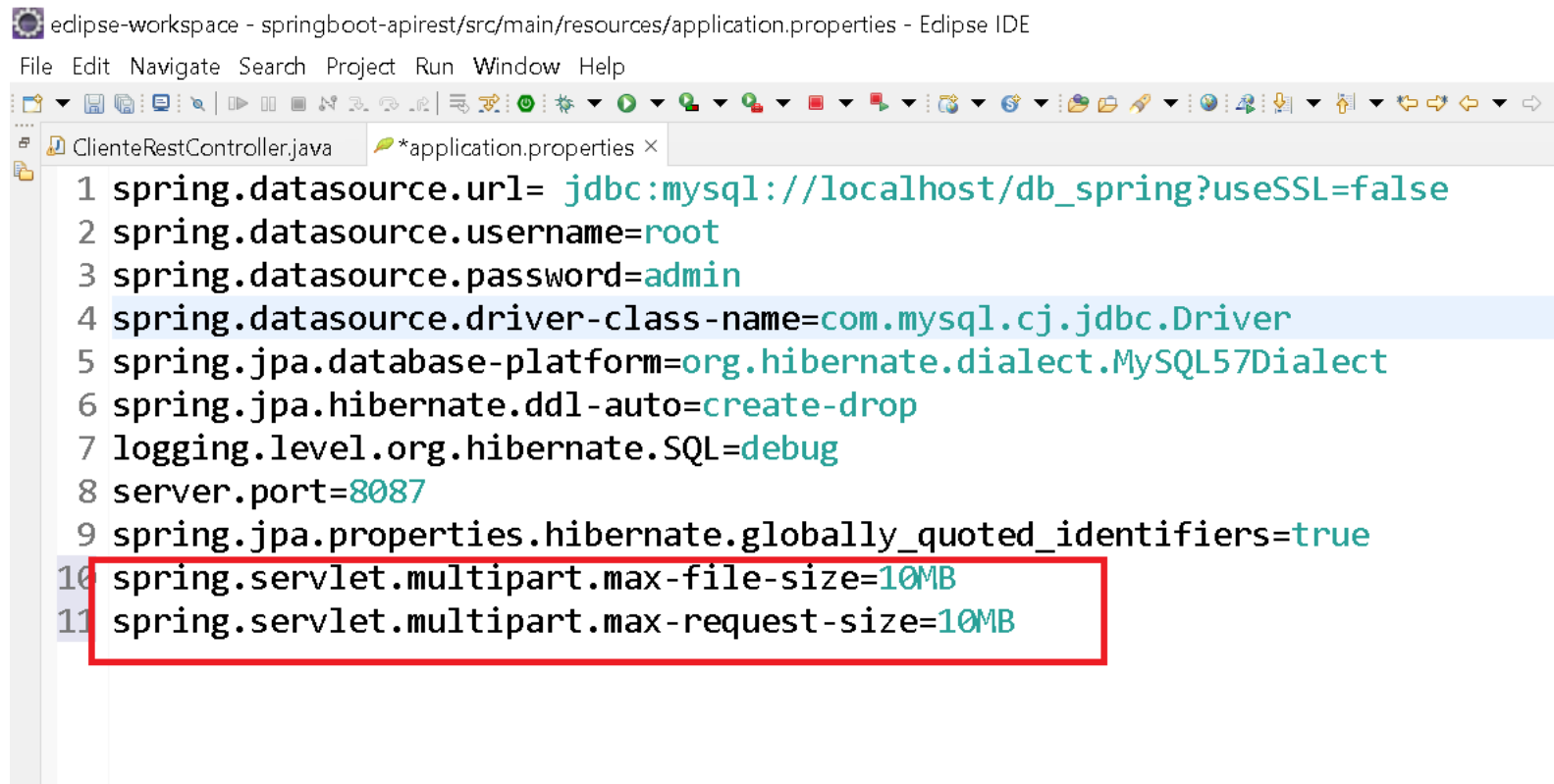
```
5  {
6    "apellido": "Perez",
7    "email": "jp@hotmail.com",
8    "telefono": 65433223,
9    "createdAt": "2021-10-01",
10   "imagen": "Image.PNG"
11 },
12 "mensaje": "Has subido correctamente la imagen: Image.PNG"
```



Realizamos mejoras a nuestra api

- ▶ Aumentar el tamaño permitido del archivo para subir por default esta limitado a 1MB
- ▶ Renombrar los archivos al subir para que no entren en conflicto al subir imágenes con el mismo nombre

Configuramos el tamaño permitido para subir archivos en application.priorerties

A screenshot of the Eclipse IDE interface. The title bar shows 'edipse-workspace - springboot-apirest/src/main/resources/application.priorerties - Edipse IDE'. The menu bar includes 'File', 'Edit', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations and development. The editor window has two tabs: 'ClienteRestController.java' and '*application.priorerties'. The 'application.priorerties' file is open and shows a list of configuration properties. Lines 10 and 11 are highlighted with a red rectangular box.

```
1 spring.datasource.url= jdbc:mysql://localhost/db_spring?useSSL=false
2 spring.datasource.username=root
3 spring.datasource.password=admin
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5 spring.jpa.database-platform=org.hibernate.dialect.MySQL57Dialect
6 spring.jpa.hibernate.ddl-auto=create-drop
7 logging.level.org.hibernate.SQL=debug
8 server.port=8087
9 spring.jpa.properties.hibernate.globally_quoted_identifiers=true
10 spring.servlet.multipart.max-file-size=10MB
11 spring.servlet.multipart.max-request-size=10MB
```

Agregamos la siguiente línea de código para generar nombre de archivos subidos

```
edipse-workspace - springboot-apirest/src/main/java/com/formacionjava/springboot/apirest/controllers/CienteRestController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

ClienteRestController.java x *application.properties

159 @PostMapping("clientes/upload")
160 public ResponseEntity<?> upload(@RequestParam("archivo") MultipartFile archivo, @RequestParam("id") Long id){
161     Map<String,Object> response = new HashMap<>();
162
163     Ciente cliente = clienteService.findById(id);
164
165     if(!archivo.isEmpty()) {
166         String nombreArchivo = UUID.randomUUID().toString()+"_"+ archivo.getOriginalFilename().replace(" ", "");
167         Path rutaArchivo = Paths.get("uploads").resolve(nombreArchivo).toAbsolutePath();
168
169         try {
170             Files.copy(archivo.getInputStream(),rutaArchivo);
171         } catch (IOException e) {
172             // TODO Auto-generated catch block
173             response.put("mensaje", "Error al subir la imagen del cliente");
174             response.put("error", e.getMessage().concat(": ").concat(e.getCause().getMessage()));
175             return new ResponseEntity<Map<String,Object>>(response,HttpStatus.INTERNAL_SERVER_ERROR);
176         }
177
178         cliente.setImagen(nombreArchivo);
179
180         clienteService.save(cliente);

```

Report

←

POST

GET h.

DEL h.

GET h.

POST

DEL h.

→

+

...

No Environment

...

http://localhost:8087/api/clientes/upload/

Save

POST

http://localhost:8087/api/clientes/upload/

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	archivo	WhatsApp Image 2021-04-21 at 09.5... x			
<input checked="" type="checkbox"/>	id	1			

Body

Cookies

Headers (5)

Test Results

201 Created

62 ms

511 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
6      "email": "jp@hotmail.com",
7      "telefono": 65433223,
8      "createdAt": "2021-10-01",
9      "imagen": "b58a2220-04ac-482b-8bef-37d1f7611976_WhatsAppImage2021-04-21at09.56.18(1).jpeg"
10    },
11    "mensaje": "Has subido correctamente la imagen:
12              b58a2220-04ac-482b-8bef-37d1f7611976_WhatsAppImage2021-04-21at09.56.18(1).jpeg"
```

Falta una ultima mejora

- ▶ Borrar los archivos asociados a un mismo usuario ya no se borra una vez reemplazado.

```
*ClienteRestController.java x *application.properties
160 @PostMapping("/clientes/upload")
161 public ResponseEntity<?> upload(@RequestParam("archivo") MultipartFile archivo, @RequestParam("id") Long id){
162     Map<String, Object> response = new HashMap<>();
163
164     Cliente cliente = clienteService.findById(id);
165
166     if(!archivo.isEmpty()) {
167         String nombreArchivo = UUID.randomUUID().toString()+"_"+ archivo.getOriginalFilename().replace(" ", "");
168         Path rutaArchivo = Paths.get("uploads").resolve(nombreArchivo).toAbsolutePath();
169
170         try {
171             Files.copy(archivo.getInputStream(), rutaArchivo);
172         } catch (IOException e) {
173             // TODO Auto-generated catch block
174             response.put("mensaje", "Error al subir la imagen del cliente");
175             response.put("error", e.getMessage().concat(": ").concat(e.getCause().getMessage()));
176             return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
177         }
178
179         String nombreFotoAnterior = cliente.getImagen();
180
181         if(nombreFotoAnterior != null && nombreFotoAnterior.length() > 0){
182             Path rutaFotoAnterior = Paths.get("uploads").resolve(nombreFotoAnterior).toAbsolutePath();
183             File archivoFotoAnterior = rutaFotoAnterior.toFile();
184
185             if(archivoFotoAnterior.exists() && archivoFotoAnterior.canRead()){
186
187                 archivoFotoAnterior.delete();
188             }
189         }
190     }
```

Agregamos el mismo código para eliminar imagen a la función delete

edipse-workspace - springboot-apirest/src/main/java/com/formacionjava/springboot/apirest/controllers/ClienteRestController.java - Edipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

*ClienteRestController.java × *application.properties

```
141
142 @DeleteMapping("clientes/{id}")
143 public ResponseEntity<?> delete(@PathVariable Long id) {
144     Map<String, Object> response = new HashMap<>();
145
146     try {
147
148         Cliente cliente = clienteService.findById(id);
149         String nombreFotoAnterior = cliente.getImagen();
150
151         if(nombreFotoAnterior != null && nombreFotoAnterior.length() > 0){
152             Path rutaFotoAnterior = Paths.get("uploads").resolve(nombreFotoAnterior).toAbsolutePath();
153             File archivoFotoAnterior = rutaFotoAnterior.toFile();
154
155             if(archivoFotoAnterior.exists() && archivoFotoAnterior.canRead()){
156
157                 archivoFotoAnterior.delete();
158             }
159         }
160
161         clienteService.delete(id);
162     } catch (DataAccessException e) {
163         response.put("mensaje", "Error al eliminar el cliente en la base de datos");
164         response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
165         return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
166     }
167     response.put("mensaje", "El cliente ha sido eliminado con éxito!");
168
169
170     return new ResponseEntity<Map<String, Object>>(response, HttpStatus.OK);
171 }
```

Postman

File Edit View Help

Home

Workspaces

API Network

Reports

Explore

Search Postman

Invite

Upgrade

My Workspace

New

Import

POST

GET h.

DEL h.

GET h.

POST

DEL h.

No Environment

Collections

Postman Echo

PruebasApiRest

APIs

Environments

Mock Servers

Monitors

Flows

History

http://localhost:8087/api/clientes/upload/

Save

Send

POST

http://localhost:8087/api/clientes/upload/

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION		Bulk Edit
<input checked="" type="checkbox"/>	archivo	Tumble Road Multicolor.png			
<input checked="" type="checkbox"/>	id	1			

Body

Cookies

Headers (5)

Test Results

201 Created

41 ms

477 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
4      "nombre": "Jose",
5      "apellido": "Perez",
6      "email": "jp@hotmail.com",
7      "telefono": 65433223,
8      "createdAt": "2021-10-01",
9      "imagen": "ab122359-53af-453d-870f-febdb865c074_TumbleRoadMulticolor.png"
10    },
11    "mensaje": "Has subido correctamente la imagen: ab122359-53af-453d-870f-febdb865c074_TumbleRoadMulticolor"
```