

UIPR – Recinto de Arecibo
Programa de Ciencias de Computadora

COMP2052.WEB DEV SERV- SIDE&MICROSE BKE

Gestor de Biblioteca

Jeremy A. Ayende Santiago (R00658578)

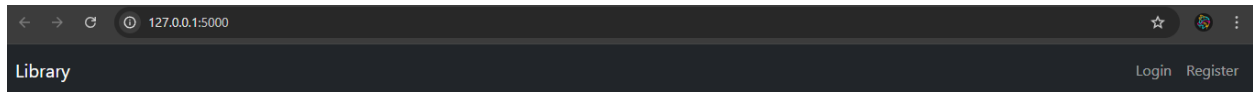
Eloy M. Toledo De Jesús (Y00651313)

Título: interfaces Front-End

Interfase del lector:

La pantalla del inicio:

Esta es la primera página que aparece cuando se corre el código y es una bienvenida para el usuario que valla a estar usando la página web.



Welcome to the Library

Please log in or register to continue.

Pantalla de Login,

En esta pantalla se ve la parte de login donde se pone el email y el password pero si no tienes una cuenta creada no te va a permitir a entrar a la pantalla de los libros.



User Login

Email

Enter your email

Password

Enter your password

Login

Don't have an account? [Register here.](#)

Esta es la pantalla para poder registrar su cuenta y dependiendo quien seas tienes que cambiar el Role al que le corresponde, por ejemplo, si soy un Lector selecciono el role de lector o si soy el bibliotecario escojo el role de bibliotecario con excepción del admin ya que ese role no lo puede usar cualquiera.

Library

LoginRegister

User Registration

Username

Enter your username

Email

Enter your email

Password

Enter your password

Confirm password

Confirm your password

Role

Lector

Register

Al usuario registrarse correctamente le aparecerá una notificación diciendo: 'User registered successfully' avisándole al usuario que su cuenta fue creada

127.0.0.1:5000/login

Library

LoginRegister

User registered successfully.

User Login

Email

Enter your email

Password

Enter your password

Login

Don't have an account? [Register here.](#)

En la imagen se presente el usuario entrando como lector después de crear su cuenta.

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/login'. The page title is 'Library'. In the top right corner, there are links for 'Login' and 'Register'. The main content area features a 'User Login' form. The form has two input fields: 'Email' with the value 'pruebafinal@example.com' and 'Password' with masked characters '*****'. A blue 'Login' button is positioned below the password field. To the right of the password field is an eye icon for toggling password visibility. Below the login form, there is a text prompt: 'Don't have an account? [Register here](#)'.

La interfaz del 'Dashboards' del lector donde se presenta información sobre los libros encontrados en la base de datos.

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/dashboard'. The page title is 'Library'. In the top right corner, there are links for 'Dashboard', 'Change Password', and 'Logout'. The main content area features a 'Welcome to the Library' heading followed by a table of books. The table has seven columns: Title, Author, ISBN, Category, Status, Year, and a lock icon. The table lists five books: 'Cronicas de Narnia', 'Los codigos', 'cuentos del antaño', 'prueba de bibliotecario', and 'book'. Below the table, there is a message: 'You do not have permission to create, update or delete courses.'

Title	Author	ISBN	Category	Status	Year	
Cronicas de Narnia	Alanis	25584889558632596596	Aventura	Disponible	2025	🔒
Los codigos	Pilato	5455487484514846	Misterio	Prestado	1959	🔒
cuentos del antaño	Pablo Escobar	0000000000001	Matanza	Disponible	1998	🔒
prueba de bibliotecario	bibliotecario	0000000000002	prueba	Prestado	1999	🔒
book	book	1111111	book	Disponible	1111	🔒

Aquí se presente la interfaz de cambio de contraseña.

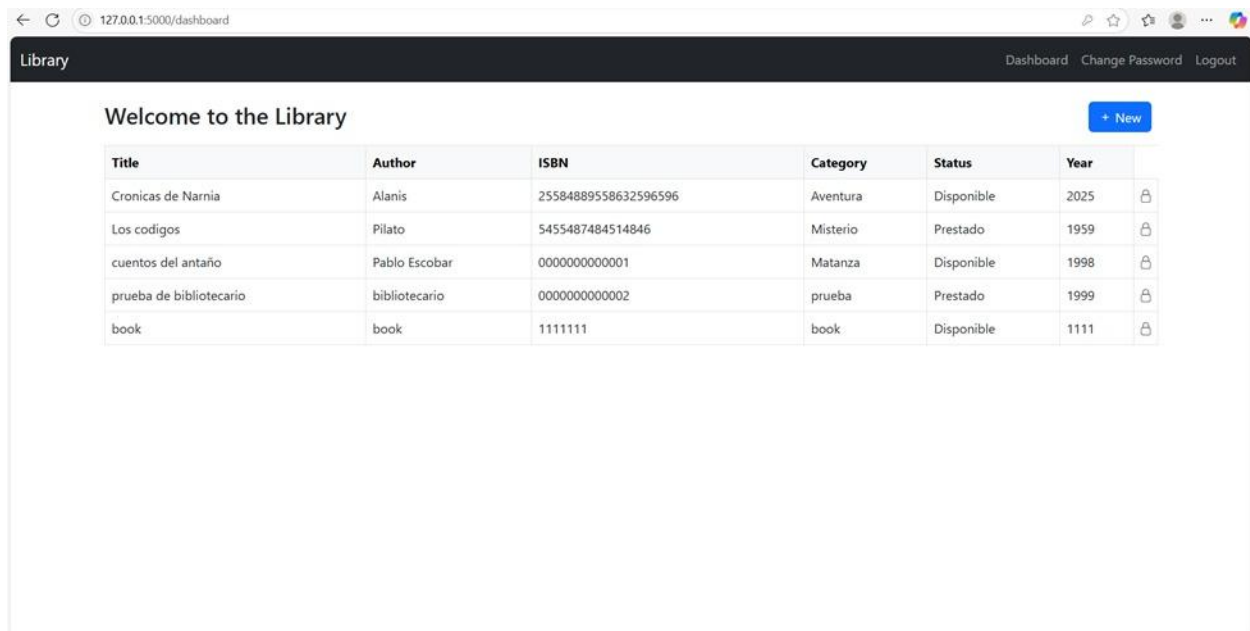
The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/cambiar-password'. The page has a dark header with 'Library' on the left and 'Dashboard', 'Change Password', and 'Logout' on the right. The main content area is titled 'Change Password' and contains three password input fields: 'Current password', 'New password', and 'Confirm new password'. Each field is filled with dots. The 'Confirm new password' field has a toggle icon on the right. Below the fields is a green 'Update Password' button.

Interfase del Bibliotecario:

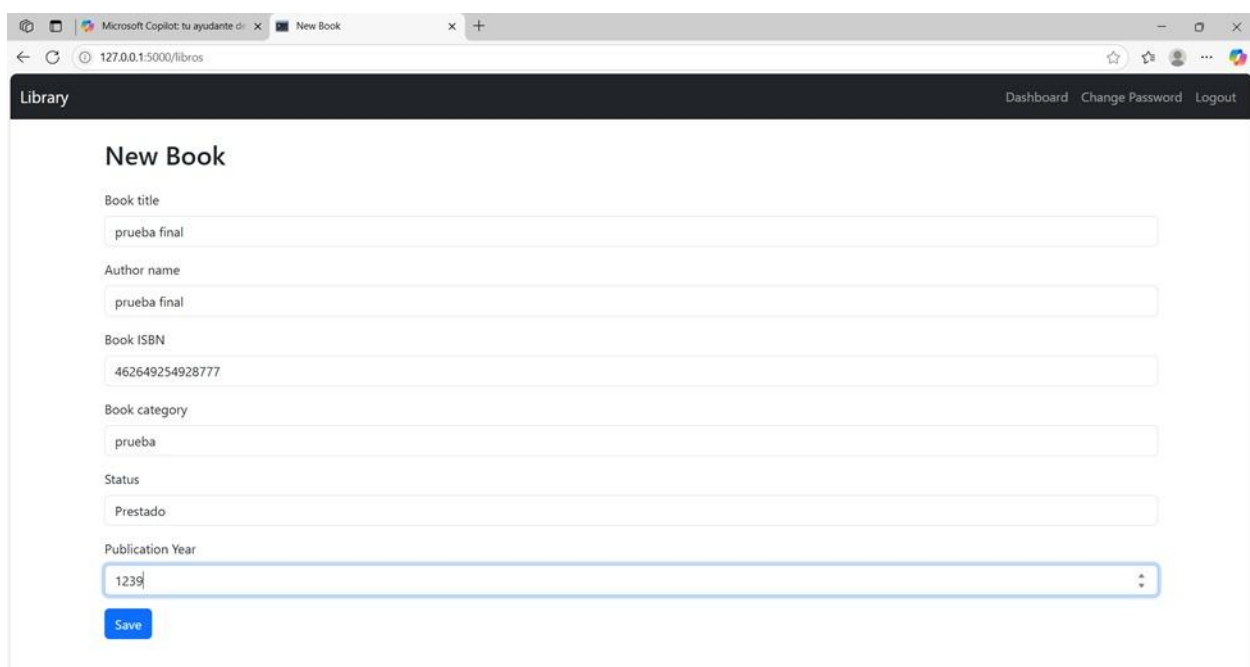
Aquí se presenta al usuario 'Bibliotecario' al iniciar sección.

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/login'. The page has a dark header with 'Library' on the left and 'Login' and 'Register' on the right. The main content area is titled 'User Login' and contains two input fields: 'Email' (with the value 'john@example.com') and 'Password' (filled with dots). Below the fields is a blue 'Login' button. At the bottom, there is a link that says 'Don't have an account? [Register here.](#)'.

Interfaz del ‘Dashboard’ del bibliotecario donde le presenta la información de los libre y le permite crear un libro nuevo.

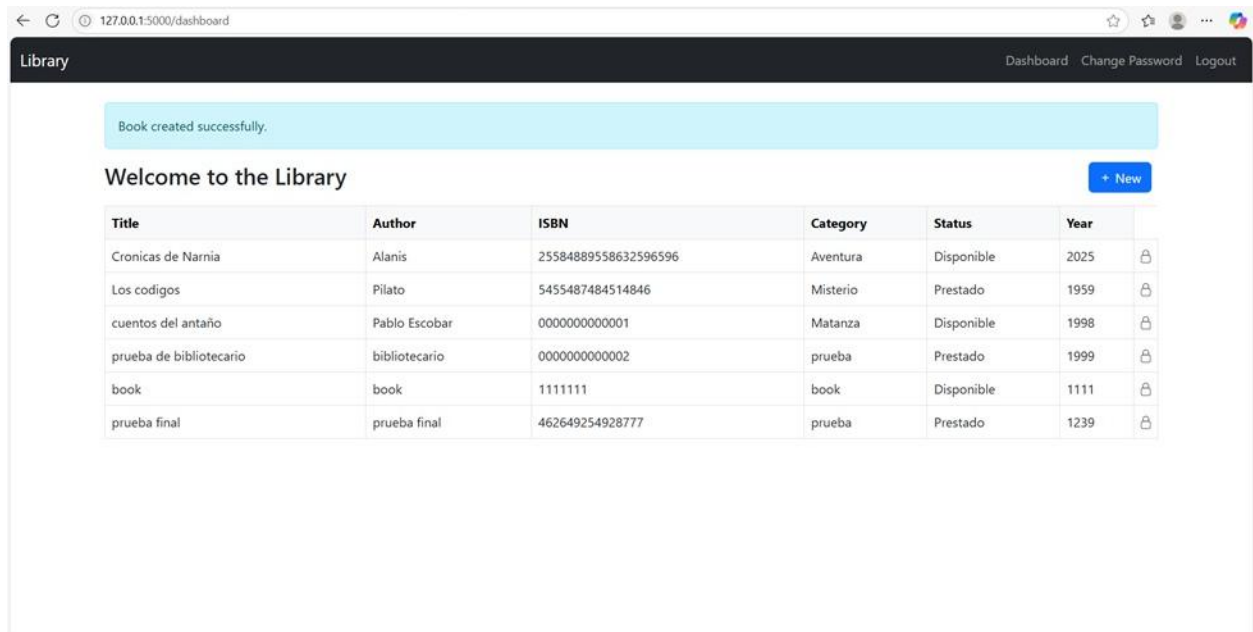


Al entrar en new el sistema le pide al bibliotecario que ingrese la información pertinente sobre el libro: Título, Autor, ISBN, Categoría del libro, Status, Año de publicación y un botón de ‘Save’ para enviar la información a la base de datos para ser almacenada y presentarla en el ‘Dashboard’.



Al crear el libro correctamente le aparecera un mensaje diciendo: ‘Libro creado exitosamente’

//Se intento de darle permiso al bibliotecario para poder poder editar y eliminar, pero no se logro incluso con la ayuda de los asistentes.//



De igual manera le aparece la pantalla de cambio de contraseña.



Interfase del Administrador:

Aquí se presenta al usuario entrando como 'Admin' al iniciar sección.

User Login

Email

dastas@example.com

Password













Login

Don't have an account? [Register here](#)

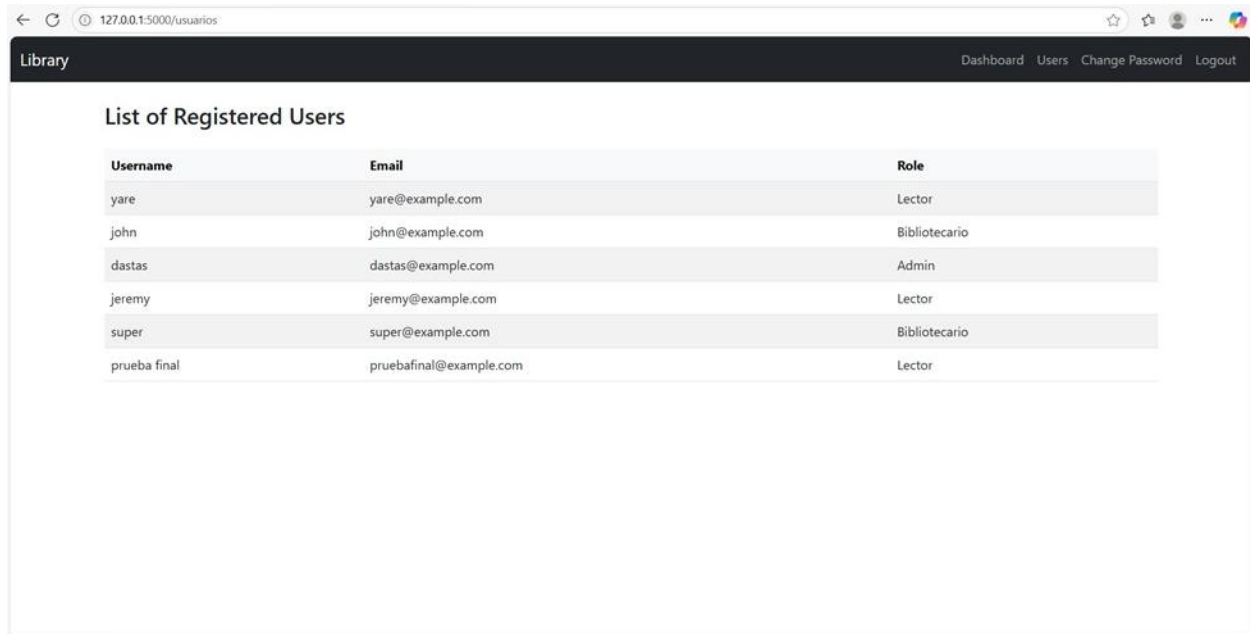
La interfaz del 'Dashboard' del administrador tiene unas opciones exclusivas para el cómo: tabla de usuarios, editiar y eliminar libros

Welcome to the Library

+ New

Title	Author	ISBN	Category	Status	Year	
Cronicas de Narnia	Alanis	25584889558632596596	Aventura	Disponible	2025	 
Los codigos	Pilato	5455487484514846	Misterio	Prestado	1959	 
cuentos del antaño	Pablo Escobar	0000000000001	Matanza	Disponible	1998	 
prueba de bibliotecario	bibliotecario	0000000000002	prueba	Prestado	1999	 
book	book	1111111	book	Disponible	1111	 
prueba final	prueba final	462649254928777	prueba	Prestado	1239	 

En esta interfaz se le presenta al 'admin' los usuarios registrados en la plataforma.



Username	Email	Role
yare	yare@example.com	Lector
john	john@example.com	Bibliotecario
dastas	dastas@example.com	Admin
jeremy	jeremy@example.com	Lector
super	super@example.com	Bibliotecario
prueba final	pruebafinal@example.com	Lector

Aquí se presenta la opción de editar la información de un libro con los siguientes campos: Título, Autor, ISBN, Categoría, Status, Año de publicación y ‘Save’



Edit Book

Book title
Cronicas de Narnia

Author name
Alanis

Book ISBN
25584889558632596596

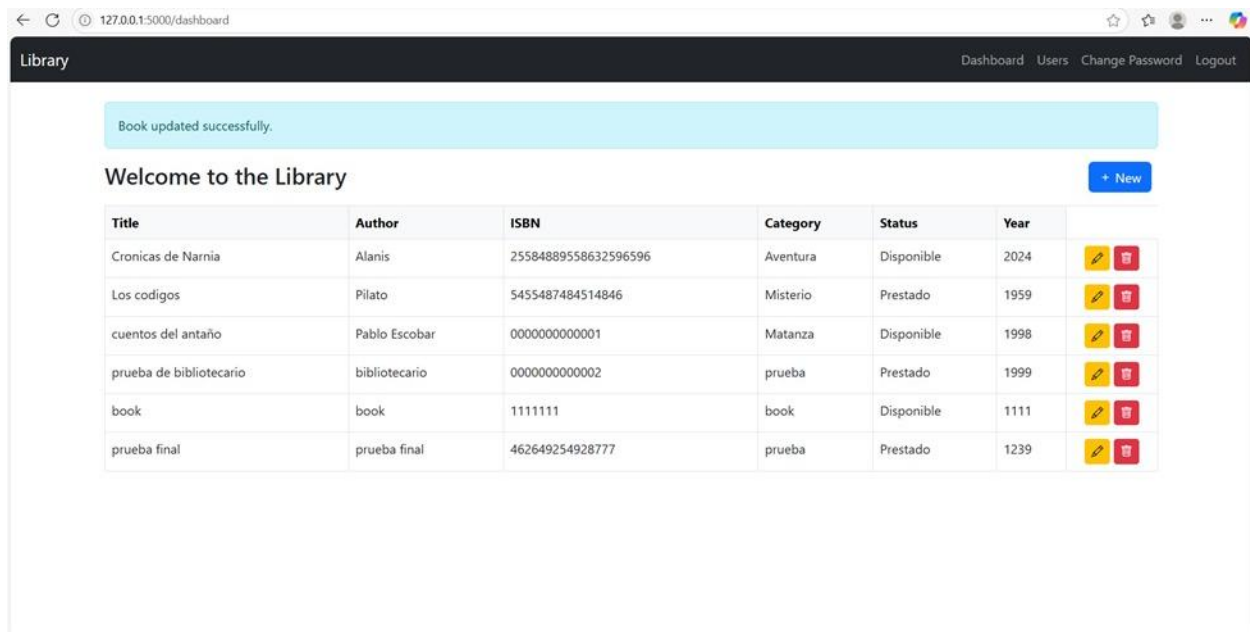
Book category
Aventura

Status
Disponible

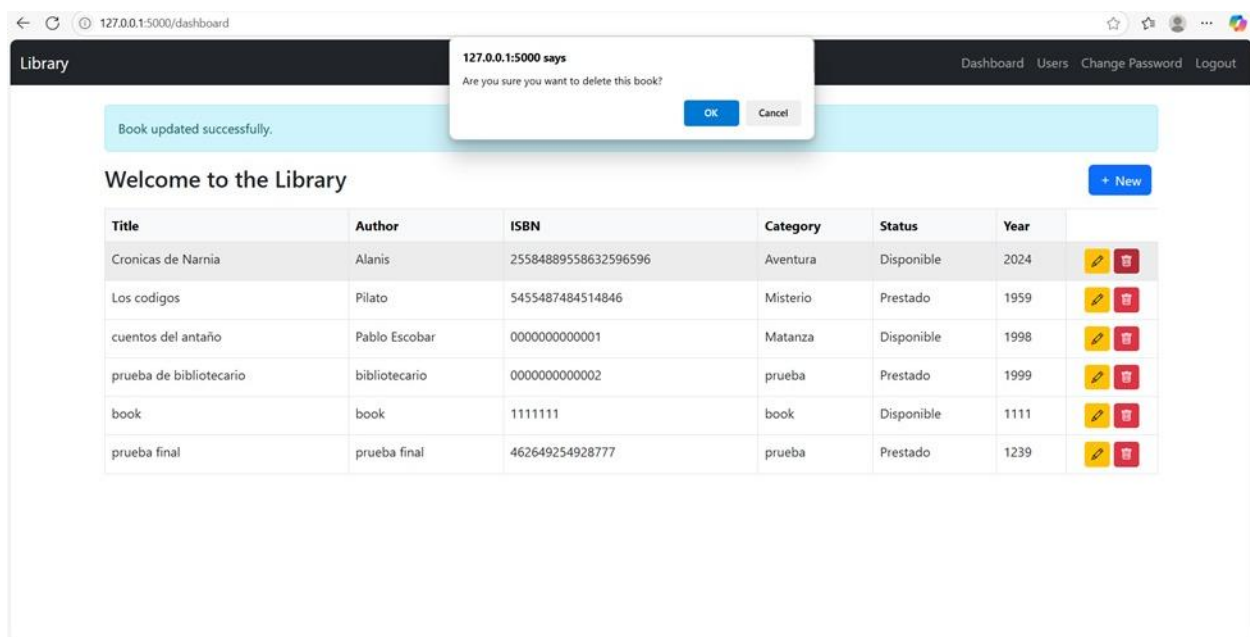
Publication Year
2025

Save

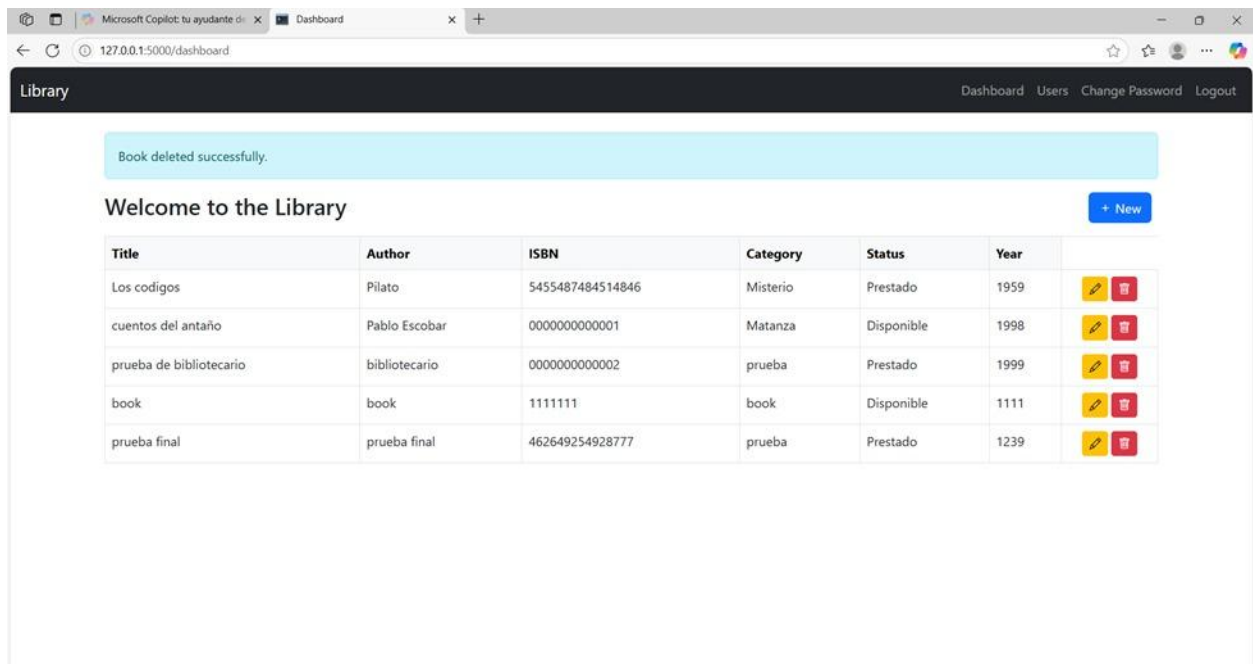
Luego de editar la información de un libro le aparecerá el siguiente mensaje: ‘Libro actualizado exitosamente’.













Al elegir la opcion de borrar le aparece un mensaje que dice: ‘Estas seguro que quieres eliminar este libro’.



Al eliminar el libro le aparece el mensaje: 'Libro eliminado exitosamente'.



The screenshot shows a web browser window with the URL `127.0.0.1:5000/dashboard`. The page is titled "Library" and has a navigation bar with links: "Dashboard", "Users", "Change Password", and "Logout". A light blue message box at the top states "Book deleted successfully." Below this, a heading "Welcome to the Library" is followed by a "+ New" button. A table lists books with columns: Title, Author, ISBN, Category, Status, Year, and action icons (edit and delete).

Title	Author	ISBN	Category	Status	Year	
Los codigos	Pilato	5455487484514846	Misterio	Prestado	1959	 
cuentos del antaño	Pablo Escobar	0000000000001	Matanza	Disponible	1998	 
prueba de bibliotecario	bibliotecario	0000000000002	prueba	Prestado	1999	 
book	book	1111111	book	Disponible	1111	 
prueba final	prueba final	462649254928777	prueba	Prestado	1239	 

También tendrá la página de cambio de contraseña



The screenshot shows a web browser window with the URL `127.0.0.1:5000/cambiar-password`. The page is titled "Library" and has a navigation bar with links: "Dashboard", "Users", "Change Password", and "Logout". The main content area is titled "Change Password" and contains three input fields: "Current password" (with placeholder "Enter current password"), "New password" (with placeholder "Enter new password"), and "Confirm new password" (with placeholder "Confirm new password"). A green "Update Password" button is at the bottom.

Título: Código de los ENDPOINTS

1. Este código permite que a través de una solicitud GET el cliente pueda recibir en formato JSON un listado de la información de los libros que se encuentran la base de datos.

```
15 @main.route('/listar_libro', methods=['GET'])
16 def listar_libro():
17     """
18     Retorna una lista de libros (JSON).
19     """
20     libro = Libro.query.all()
21
22     data = [
23         {'id': libro.id, 'titulo': libro.titulo, 'autor': libro.autor, 'isbn': libro.isbn, 'categoria': libro.categoria,
24          'estado': libro.estado, 'año_publicacion': libro.año_publicacion, 'bibliotecario_id': libro.bibliotecario_id}
25         for libro in libro
26     ]
27     return jsonify(data), 200
28
```

2. Este código permite que a través de una solicitud GET le presente al cliente la información de un libro basándose en el ID del libro que el usuario coloque.

```
30 @main.route('/libro/<int:id>', methods=['GET'])
31 def listar_un_libro(id):
32     """
33     Retorna un solo libro por su ID (JSON).
34     """
35     libro = Libro.query.get_or_404(id)
36
37     data = {
38         'id': libro.id,
39         'titulo': libro.titulo,
40         'autor': libro.autor,
41         'isbn': libro.isbn,
42         'categoria': libro.categoria,
43         'estado': libro.estado,
44         'año_publicacion': libro.año_publicacion,
45         'bibliotecario_id': libro.bibliotecario_id
46     }
47
48     return jsonify(data), 200
```

3. Este código permite que el cliente a través de una solicitud POST puede crear un libro nuevo ingresando la información pertinente del libro.

```
51 @main.route('/crear_libro', methods=['POST'])
52 def crear_libro():
53     """
54     Crea un libro sin validación.
55     Espera JSON con 'titulo', 'autor' y 'bibliotecario_id'.
56     """
57     data = request.get_json()
58
59     if not data:
60         return jsonify({'error': 'No input data provided'}), 400
61
62     nuevo_libro = Libro(
63         titulo=data.get('titulo'),
64         autor=data.get('autor'),
65         isbn=data.get('isbn'),
66         categoria=data.get('categoria'),
67         estado=data.get('estado'),
68         año_publicacion=data.get('año_publicacion'),
69         bibliotecario_id=data.get('bibliotecario_id') # sin validación de usuario
70     )
71
72     db.session.add(nuevo_libro)
73     db.session.commit()
74
75     return jsonify({'message': 'Libro creado', 'id': nuevo_libro.id, 'bibliotecario_id': nuevo_libro.bibliotecario_id}), 201
```

4. Este código permite que el usuario a través de una solicitud de PUT pueda actualizar la información de un libro ya existente basándose en el ID del libro que el cliente coloque.

```
77 @main.route('/actualizar_libro/<int:id>', methods=['PUT'])
78 def actualizar_libro(id):
79     """
80     Actualiza un libro sin validación de usuario o permisos.
81     """
82     libro = Libro.query.get_or_404(id)
83     data = request.get_json()
84
85     libro.titulo = data.get('titulo', libro.titulo)
86     libro.autor = data.get('autor', libro.autor)
87     libro.isbn = data.get('isbn', libro.isbn)
88     libro.categoria = data.get('categoria', libro.categoria)
89     libro.estado = data.get('estado', libro.estado)
90     libro.año_publicacion = data.get('año_publicacion', libro.año_publicacion)
91     libro.bibliotecario_id = data.get('bibliotecario_id', libro.bibliotecario_id)
92
93     db.session.commit()
94
95     return jsonify({'message': 'Libro actualizado', 'id': libro.id}), 200
96
```

5. Este código permite que el cliente a través de una solicitud DELETE pueda eliminar un libro basándose en el ID del libro que el cliente coloque.

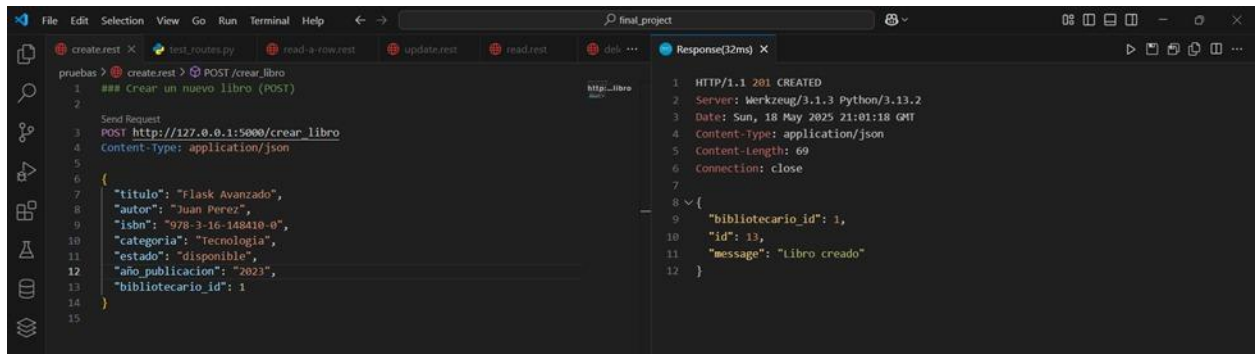
```
97  @main.route('/eliminar_libro/<int:id>', methods=['DELETE'])
98  ∨ def eliminar_libro(id):
99  ∨      """
100      Elimina un libro sin validación de permisos.
101      """
102      libro = Libro.query.get_or_404(id)
103      db.session.delete(libro)
104      db.session.commit()
105
106      return jsonify({'message': 'Libro eliminado', 'id': libro.id}), 200
```

Tabla de datos enviados y recibidos:

Nombre del Archivo	Valores Enviados	Valores Esperados
create.rest	POST http://127.0.0.1:5000/crear_libro Content-Type: application/json <pre>{ "titulo":, "autor":. "isbn":, "categoria":, "estado":, "año_publicacion":, "bibliotecario_id":#ID }</pre>	<pre>{ "bibliotecario_id":#ID "id"#ID Libro:, "message": "Libro creado" }</pre>
read-a-row-.rest	GET http://127.0.0.1:5000//libro/#ID Libro Content-Type: application/json	<pre>{ "autor":, "a\u00f1o_publicacion":, "bibliotecario_id": #ID, "categoria":, "estado":, "id": #ID Libro, "isbn":, "titulo": }</pre>
read.rest	GET http://127.0.0.1:5000/listar_libro Content-Type: application/json	Presentará la información de todos los libros que se encuentran en la base de datos.
update.rest	PUT http://localhost:5000/actualizar_libro/#ID Libro Content-Type: application/json <pre>{ Información que se quiere actualizar }</pre>	<pre>{ "id": #ID Libro, "message": "Libro actualizado" }</pre>
delete.rest	DELETE http://localhost:5000/eliminar_libro/#ID Libro Content-Type: application/json	<pre>{ "id": # ID Libro, "message": "Libro eliminado" }</pre>

Titulo: Pruebas de los ENDPOINTS del CRUD

1. La captura muestra una prueba de una API REST para crear un libro usando el método POST en una aplicación Flask. Al realizarse correctamente la solicitud de registro en el sistema la API respondió con éxito, asignando el ID 13 al nuevo libro.

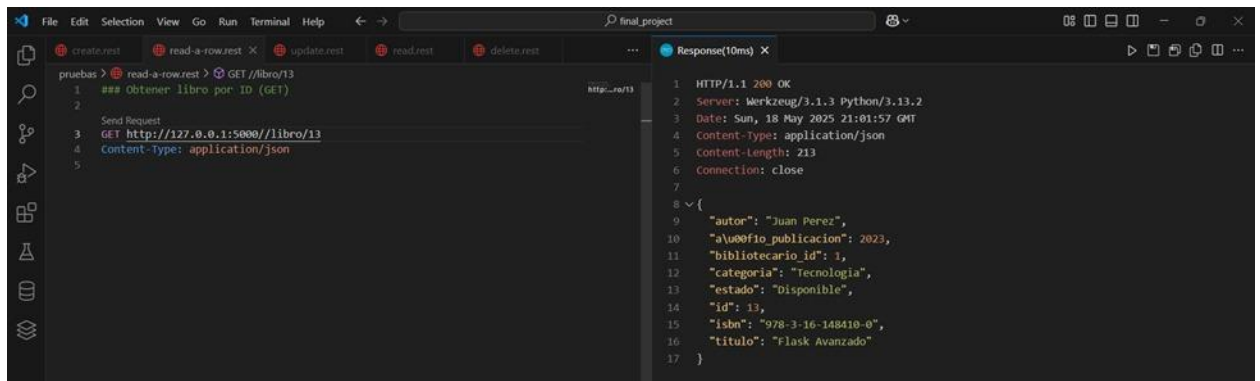


The screenshot shows a REST client interface with a tab for 'create.rest'. The request is a POST to 'http://127.0.0.1:5000/crear_libro' with a JSON body. The response is a 201 status code with a JSON body indicating the book was created with ID 13.

```
pruebas > create.rest > POST /crear_libro
1  ### Crear un nuevo libro (POST)
2
3  Send Request
4  POST http://127.0.0.1:5000/crear_libro
5  Content-Type: application/json
6
7  {
8    "titulo": "Flask Avanzado",
9    "autor": "Juan Perez",
10   "isbn": "978-3-16-148410-0",
11   "categoria": "Tecnologia",
12   "estado": "disponible",
13   "año_publicacion": "2023",
14   "bibliotecario_id": 1
15 }
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
1 HTTP/1.1 201 CREATED
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Sun, 18 May 2025 21:01:18 GMT
4 Content-Type: application/json
5 Content-Length: 69
6 Connection: close
7
8 {
9   "bibliotecario_id": 1,
10  "id": 13,
11  "message": "Libro creado"
12 }
```

2. La captura muestra una solicitud GET a una API REST para obtener información de un libro con ID 13. La solicitud se envía, y la respuesta JSON devuelve detalles exclusivos del libro, incluyendo su título, autor, ISBN, año de publicación, categoría y estado.



The screenshot shows a REST client interface with a tab for 'read-a-row.rest'. The request is a GET to 'http://127.0.0.1:5000/libro/13'. The response is a 200 status code with a JSON body containing the book's details.

```
pruebas > read-a-row.rest > GET /libro/13
1  ### Obtener libro por ID (GET)
2
3  Send Request
4  GET http://127.0.0.1:5000/libro/13
5  Content-Type: application/json
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Sun, 18 May 2025 21:01:57 GMT
4 Content-Type: application/json
5 Content-Length: 213
6 Connection: close
7
8 {
9   "autor": "Juan Perez",
10  "año_publicacion": 2023,
11  "bibliotecario_id": 1,
12  "categoria": "Tecnologia",
13  "estado": "Disponible",
14  "id": 13,
15  "isbn": "978-3-16-148410-0",
16  "titulo": "Flask Avanzado"
17 }
```


- [illegible]

-
- The screenshot shows a REST client interface with a menu bar (File, Edit, Selection, View, Go, Run, Terminal, Help) and a toolbar. The main workspace is divided into two panes. The left pane, titled 'pruebas > update.rest > ...', contains a REST client request for a POST method to 'http://localhost:5000/actualizar_libro/13'. The request body is a JSON object: { "titulo": "Generaciones de Flask", "autor": "Maria Lopez" }. The right pane, titled 'Response(20ms) X', displays the response status 'HTTP/1.1 200 OK' and the response body: { "id": 13, "message": "Libro actualizado" }. The interface also includes a sidebar with various tool icons on the left and a top toolbar with window and application controls.
- ```

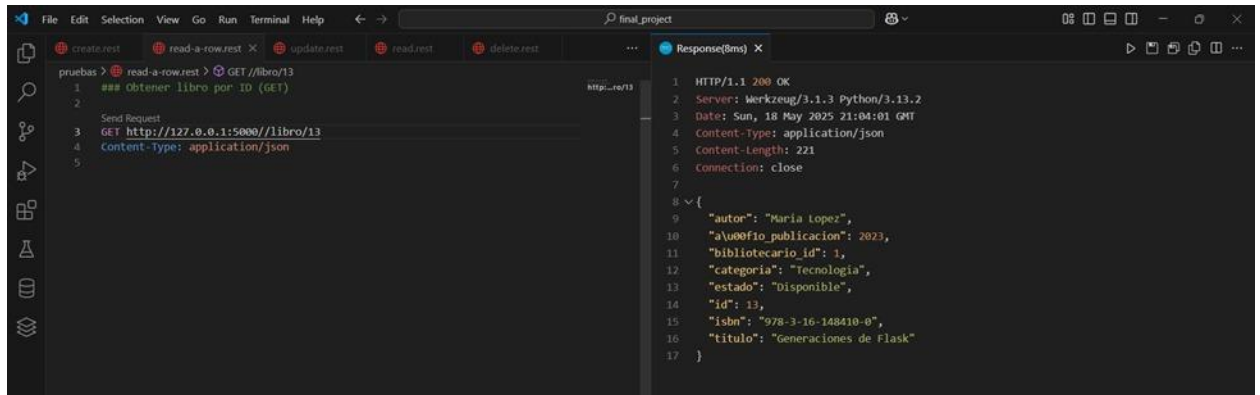
File Edit Selection View Go Run Terminal Help
create.rest read-a-row.rest update.rest X read.rest delete.rest ... Response(20ms) X

pruebas > update.rest > ...
1 ### Editar un libro (POST)
2
3 # Simular POST con nuevos datos del libro
4
5 Send Request
6 PUT http://localhost:5000/actualizar_libro/13
7 Content-Type: application/json
8
9 {
10 "titulo": "Generaciones de Flask",
11 "autor": "Maria Lopez"
12 }

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Sun, 18 May 2025 21:03:43 GMT
4 Content-Type: application/json
5 Content-Length: 49
6 Connection: close
7
8 {
9 "id": 13,
10 "message": "Libro actualizado"
11 }

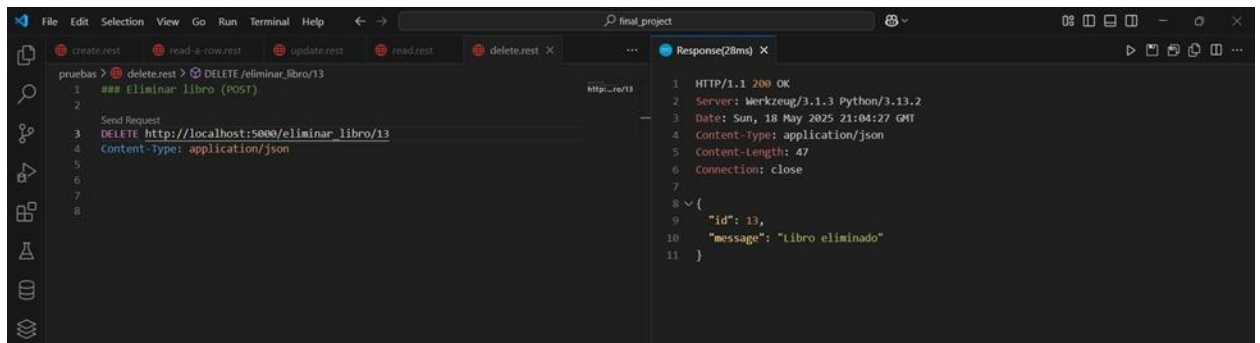
```

5. La captura presenta la información de la actualización del libro con ID 13.



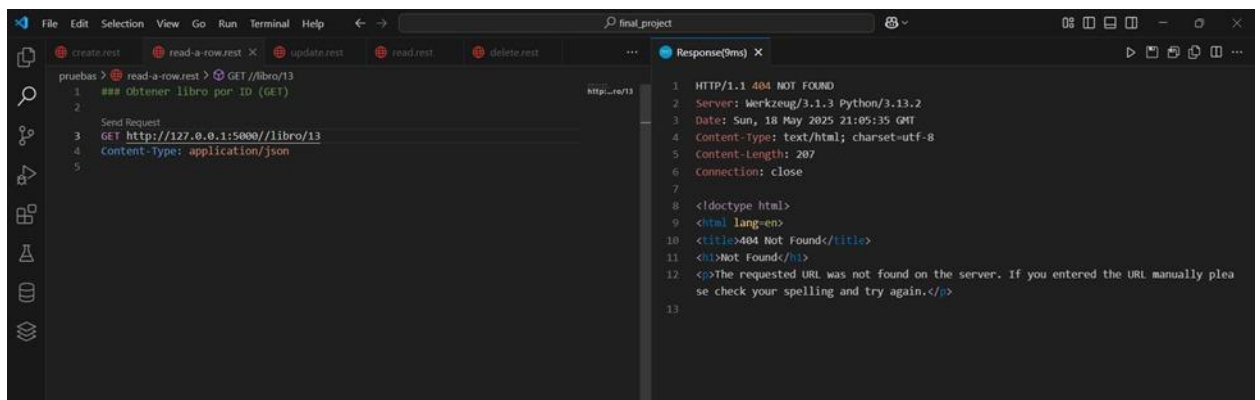
```
File Edit Selection View Go Run Terminal Help
final_project
create.rest read-a-row.rest update.rest read.rest delete.rest
pruebas > read-a-row.rest > GET /libro/13
1 ### Obtener libro por ID (GET)
2
3 Send Request
4 GET http://127.0.0.1:5000/libro/13
5 Content-Type: application/json
6
7
8
9
10
11
12
13
14
15
16
17
Response(8ms) X
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Sun, 18 May 2025 21:04:01 GMT
4 Content-Type: application/json
5 Content-Length: 221
6 Connection: close
7
8 {
9 "autor": "Maria Lopez",
10 "añoDePublicacion": 2023,
11 "bibliotecario_id": 1,
12 "categoria": "tecnologia",
13 "estado": "Disponible",
14 "id": 13,
15 "isbn": "978-3-16-148410-0",
16 "titulo": "Generaciones de Flask"
17 }
```

6. La siguiente captura presenta una solicitud para eliminar un libro utilizando el ID del libro a través de una solicitud DELETE. El API responde con el mensaje: 'Libro eliminado'.



```
File Edit Selection View Go Run Terminal Help
final_project
create.rest read-a-row.rest update.rest read.rest delete.rest
pruebas > delete.rest > DELETE /eliminar_libro/13
1 ### Eliminar libro (POST)
2
3 Send Request
4 DELETE http://localhost:5000/eliminar_libro/13
5 Content-Type: application/json
6
7
8
9
10
11
12
13
14
15
16
17
Response(28ms) X
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Sun, 18 May 2025 21:04:27 GMT
4 Content-Type: application/json
5 Content-Length: 47
6 Connection: close
7
8 {
9 "id": 13,
10 "message": "libro eliminado"
11 }
```

7. Aquí se presenta que al eliminar el libro no aparece nuevamente en la base de datos.



```
File Edit Selection View Go Run Terminal Help
final_project
create.rest read-a-row.rest update.rest read.rest delete.rest
pruebas > read-a-row.rest > GET /libro/13
1 ### Obtener libro por ID (GET)
2
3 Send Request
4 GET http://127.0.0.1:5000/libro/13
5 Content-Type: application/json
6
7
8
9
10
11
12
13
14
15
16
17
Response(9ms) X
1 HTTP/1.1 404 NOT FOUND
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Sun, 18 May 2025 21:05:35 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 207
6 Connection: close
7
8 <!doctype html>
9 <html lang=en>
10 <title>404 Not Found</title>
11 <h1>Not Found</h1>
12 <p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
13
```

# Integrantes:

| Nombre de los integrantes | Carpeta en GitHub                                                                                                                     |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Jeremy A. Ayende Santiago | <a href="https://github.com/ayende10/proyecto_final.git">https://github.com/ayende10/proyecto_final.git</a>                           |
| Eloy M. Toledo De Jesús   | <a href="https://github.com/Eloy841/proyecto_final_copia_de_Eloy.git">https://github.com/Eloy841/proyecto_final_copia_de_Eloy.git</a> |