



**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
CARRERA DE INGENIERÍA EN SOFTWARE
PRÁCTICA PRE PROFESIONAL NO REMUNERADA**

INFORME DE:

☐

Pasantía

☐

Ayudante de Cátedra

☒

Práctica Pre Profesional No Remunerada

☐

Ayudante de Investigación

☐

Servicio a la comunidad

CÓDIGO SOLUTE S.A

ESTUDIANTE: STEVEN JEFFERSON POZO ANALUISA
TUTOR ACADÉMICO: JENNY ALEXANDRA RUIZ ROBALINO

CALIFICACIÓN DEL INFORME

19

JENNY ALEXANDRA RUIZ ROBALINO STEVEN JEFFERSON POZO ANALUISA
Tutora académica Estudiante

JAIME LEOPOLDO VELA MARTÍNEZ
Tutor Empresarial

Quito, 07/02/2025

1. INTRODUCCIÓN

En este informe se describe el desarrollo de las prácticas preprofesionales realizadas en la empresa CODIGO SOLUTE S.A. Durante este período, el trabajo estuvo enfocado en el desarrollo del backend de una aplicación de escritorio para la biblioteca del colegio *Saint Patrick School*. Además, se llevaron a cabo tareas relacionadas con la manipulación de bases de datos relacionales y la codificación de scripts en el sistema operativo Windows.

Las actividades realizadas se centraron en la programación del backend utilizando el framework Spring Boot, lo que permitió construir un servidor robusto para atender solicitudes desde el cliente a través de una API RESTful. También se trabajó con la base de datos relacional MySQL y se desarrollaron archivos .bat para la gestión de tareas en Windows.

Como se mencionó anteriormente, estas prácticas se desarrollaron en la empresa CODIGO SOLUTE S.A., especializada en la implementación de sistemas de seguridad como biométricos y cámaras de vigilancia. En particular, el área de desarrollo de software, en la que se participó, se dedica a crear soluciones para terceros, como fue el caso de la aplicación de escritorio solicitada por la institución educativa *Saint Patrick School*.

El desarrollo del proyecto fue una tarea colectiva en la que se trabajó estrechamente con el supervisor encargado del proyecto, quien además actuó como enlace con el cliente, representado por la institución educativa quien es el jefe de sistemas de dicha institución.

Las prácticas tuvieron una duración de tres meses, iniciando el 30 de septiembre de 2024 y finalizando el 20 de diciembre de 2024. Inicialmente, las reuniones se realizaron de manera presencial, pero debido a una crisis energética, se optó por reuniones virtuales para dar continuidad al trabajo.

Las actividades realizadas durante las prácticas preprofesionales están alineadas con el perfil de egreso de un ingeniero en software, ya que se enfocaron en el desarrollo de software, específicamente en la implementación APIs que se consumen para realizar peticiones desde cliente y devolver la información recuperada desde la base de datos hacia el sistema. Además, se aplicaron conocimientos adquiridos en la universidad, como el uso de sistemas operativos, y documentación, lo que permitió afrontar las tareas asignadas sin dificultades.

Por último, cabe destacar que el sistema desarrollado tiene como objetivo principal optimizar la gestión de la biblioteca de la institución educativa, mediante la creación de APIs, las cuales cubre las necesidades de registrar libros y usuarios a partir de un control de id único, con la finalidad de evitar registros redundantes. Así también, la posibilidad de realizar préstamos y devoluciones de dichos libros. En la sección de desarrollo se detalla las actividades y funcionalidad del backend del sistema.

2. DESARROLLO

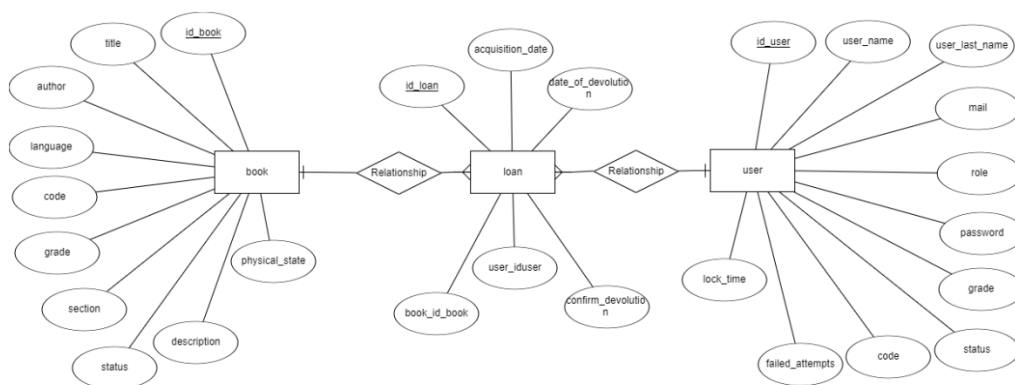
Durante el desarrollo del sistema de gestión de biblioteca, se llevaron a cabo varios procesos. A continuación, se detallan dichos procesos.

2.1. Base de datos

En este punto, se trabajó con la base de datos MySQL, el cual permitió generar un modelo lógico que establece la lógica del comportamiento del sistema. Tal como se observa en la Figura 1, se elaboró un diagrama ER que establecía las entidades participantes del sistema y sus atributos respectivos.

Figura 1

Diagrama ER del sistema de gestión de biblioteca

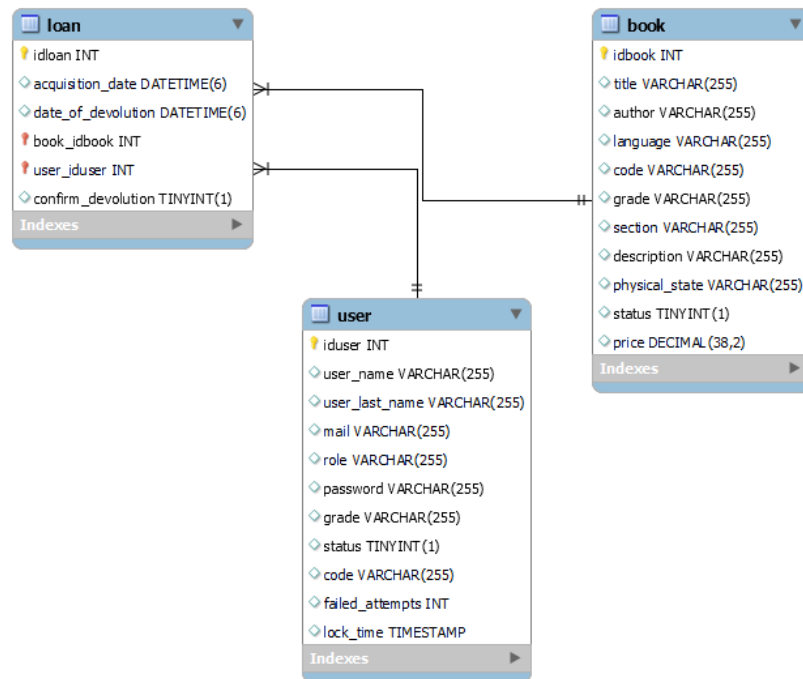


Nota: Diagrama entidad relación Realizado con ERDPlus online. Elaborado por Steven Pozo, 2025, Quito.

Así también, con el uso del gestor de base de datos de Workbench se obtuvo el diagrama físico del sistema a nivel de base de datos, dicho diagrama se lo muestra a continuación, en la figura 2.

Figura 2

Diagrama Físico del sistema de gestión de biblioteca



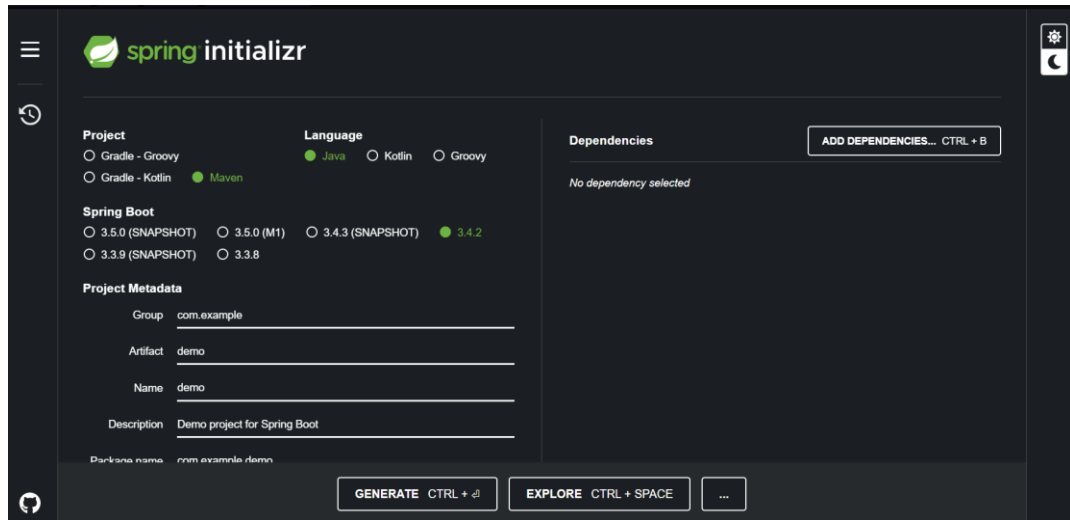
Nota: Diagrama Físico de las tablas y atributos realizado en el gestor de base de datos Workbench. Elaborado por Steven Pozo, 2025, Quito.

2.2.Backend Springboot

Para el desarrollo del backend usando el Framework de Springboot, se creó un proyecto utilizando una herramienta que permite gestionar dependencia; lenguajes de programación como, java, kotlin y Groovy, siendo Java el lenguaje que se implementó en la construcción del backend. La herramienta que permitió realizar esto fue Spring initializr, el cual se lo muestra en la figura 3.

Figura 3.

Spring inicializr



Nota: Herramienta spring inicializr usado para generar proyectos Springboot.

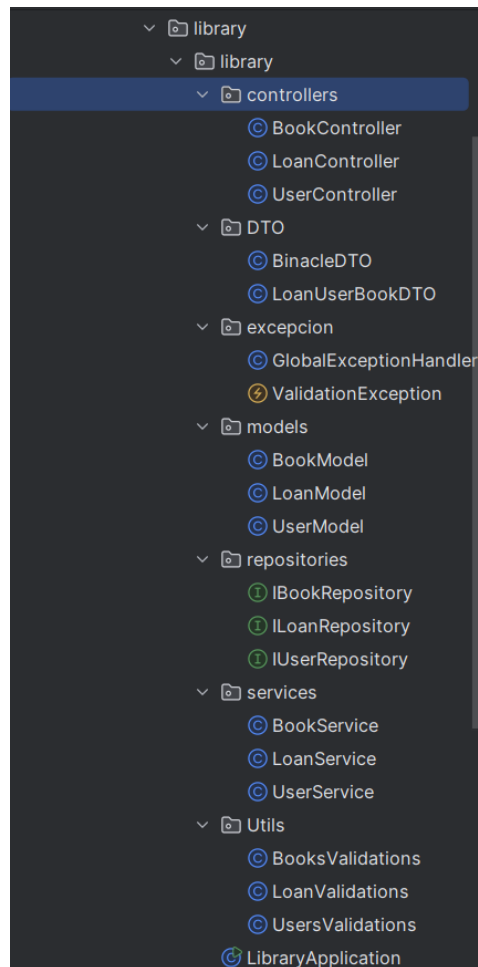
Elaborado por Steven Pozo, 2025, Quito.

Además, el uso de un IDE de programación especializado en algunos lenguajes, especialmente java, fue de utilidad. Este proyecto fue desarrollado en IntelliJ IDEA Community Edition, siendo una versión gratuita y muy llevadera.

Al acoplar el proyecto generado con el IDE, se implementó una arquitectura en capas, ya que, es sencillo de implementar y se alinea al nivel de dificultad del sistema de gestión de biblioteca. Para ello, se generó 7 capas que permitieron llevar a cabo un trabajo específico en cada uno de ellos, tal como se muestra en la figura 4.

Figura 4

Arquitectura en capas del sistema



Nota: Estructura del backend del sistema de gestión de biblioteca, donde se muestran las capas que complementan su funcionamiento. Elaborado por Steven Pozo, 2025, Quito.

2.3.Funcionalidad de las APIs

- **User**

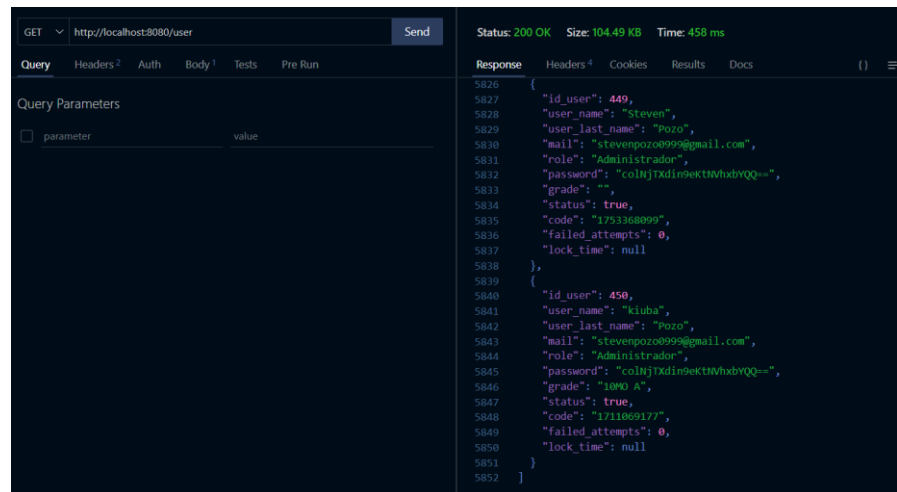
Para la gestión de usuarios se construyó 8 APIs, las cuales se las describe a continuación.

1) Obtención de todos los usuarios

Tal como se observa en la figura 5, se obtiene todos los usuarios del sistema. En este caso se devuelven todos los usuarios con todos sus datos.

Figura 5

API para obtener todos los usuarios con todos sus datos



```

GET http://localhost:8080/user
Status: 200 OK Size: 104.49 KB Time: 458 ms

Response
{
  "id_user": 449,
  "user_name": "Steven",
  "user_last_name": "Pozo",
  "mail": "stevenpozo0999@gmail.com",
  "role": "Administrador",
  "password": "colhj7Xdin9ekTlWxbvQQ==",
  "grade": "",
  "status": true,
  "code": "1753368099",
  "failed_attempts": 0,
  "lock_time": null
},
{
  "id_user": 450,
  "user_name": "kiuba",
  "user_last_name": "Pozo",
  "mail": "stevenpozo0999@gmail.com",
  "role": "Administrador",
  "password": "colhj7Xdin9ekTlWxbvQQ==",
  "grade": "10MO A",
  "status": true,
  "code": "1711069177",
  "failed_attempts": 0,
  "lock_time": null
}

```

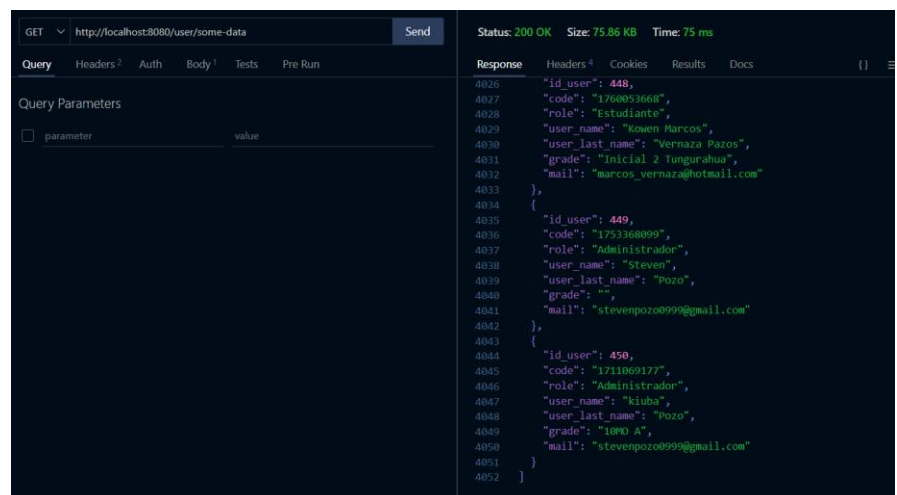
Nota: APIRest para obtener todos los usuarios registrados en el sistema. Elaborado por Steven Pozo, 2025, Quito.

2) Filtro de obtención de usuarios

Este api, actúa como un filtro, devolviendo solo algunos datos específicos del usuario. Al ejecutarlo se genera la respuesta, tal como se muestra en la figura 6.

Figura 6

API para datos específicos de los usuarios



```

GET http://localhost:8080/user/some-data
Status: 200 OK Size: 75.86 KB Time: 75 ms

Response
{
  "id_user": 448,
  "code": "1768053668",
  "role": "Estudiante",
  "user_name": "Kowen Marcos",
  "user_last_name": "Vernaza Pazos",
  "grade": "Inicial 2 Tungurahua",
  "mail": "marcos_vernaza@hotmail.com"
},
{
  "id_user": 449,
  "code": "1753368099",
  "role": "Administrador",
  "user_name": "Steven",
  "user_last_name": "Pozo",
  "grade": "",
  "status": true,
  "code": "1711069177",
  "failed_attempts": 0,
  "lock_time": null
},
{
  "id_user": 450,
  "code": "1711069177",
  "role": "Administrador",
  "user_name": "kiuba",
  "user_last_name": "Pozo",
  "grade": "10MO A",
  "status": true,
  "code": "1711069177",
  "failed_attempts": 0,
  "lock_time": null
}

```

Nota: APIRest para obtener todos los usuarios registrados en el sistema con datos personalizados en la respuesta del JSON.

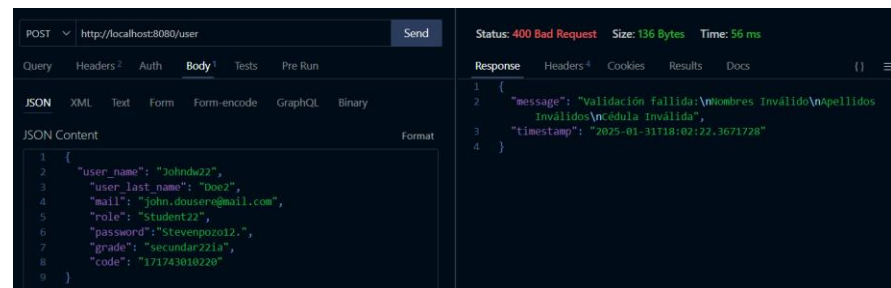
Elaborado por Steven Pozo, 2025, Quito.

3) Guardar un nuevo usuario (administrativo)

Para guardar a un usuario, administrativo, la única diferencia es que este tendrá contraseña. Para ello, se usa la API con el método POST. Además, se valida los campos de entrada siendo la respuesta un Bad Request, en caso de que haya excepciones, adicionalmente se devuelven los errores en una sola lista, tal como se muestra en la figura 7.

Figura 7

Validación de campos al guardar un usuario

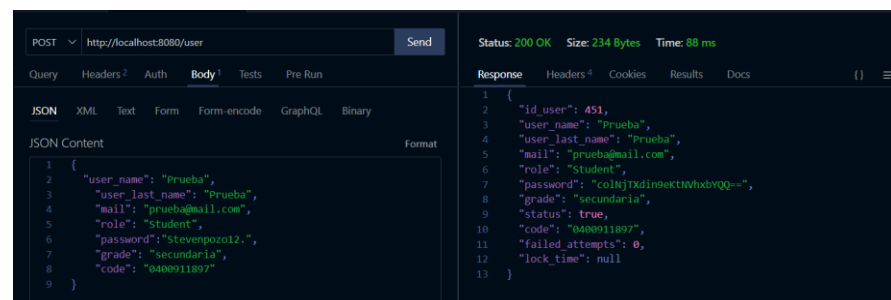


Nota: API Rest para guardar un nuevo usuario, pero con validación de campos. Elaborado por Steven Pozo, 2025, Quito.

Por el contrario, si los datos de entrada son correctos, se guarda exitosamente, tal como se muestra en la figura 8.

Figura 8

Agregación de un nuevo usuario administrativo con éxito.



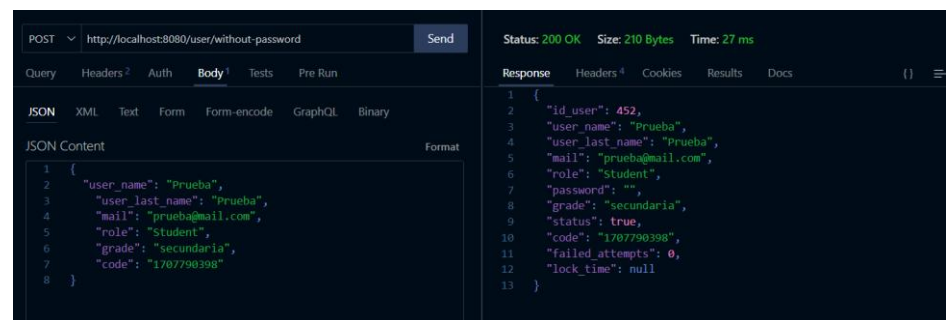
Nota: API Rest para guardar un nuevo usuario, pero sin excepciones. Elaborado por Steven Pozo, 2025, Quito.

4) Guardar un nuevo usuario (estudiantes, profesores y personal institucional)

Para guardar un nuevo usuario de tipo estudiante, profesor o personal, se lo hace sin la contraseña, ya que no tienen acceso al sistema. Para ello, la API valida los campos de entrada y después de hacerlo, guarda el registro si no existen excepciones. Esto se lo muestra en la figura 9.

Figura 9

Agregación de un usuario que no es administrativo



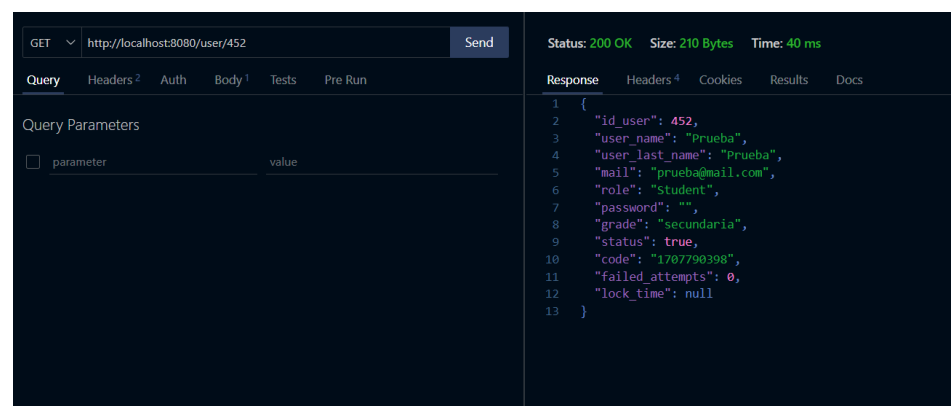
Nota: APIRest para guardar un nuevo usuario que no es administrativo. Elaborado por Steven Pozo, 2025, Quito.

5) Obtener usuario por ID

Esta API permite obtener un usuario por su id, tal como se muestra en la figura 10.

Figura 10

Obtención de usuario por id



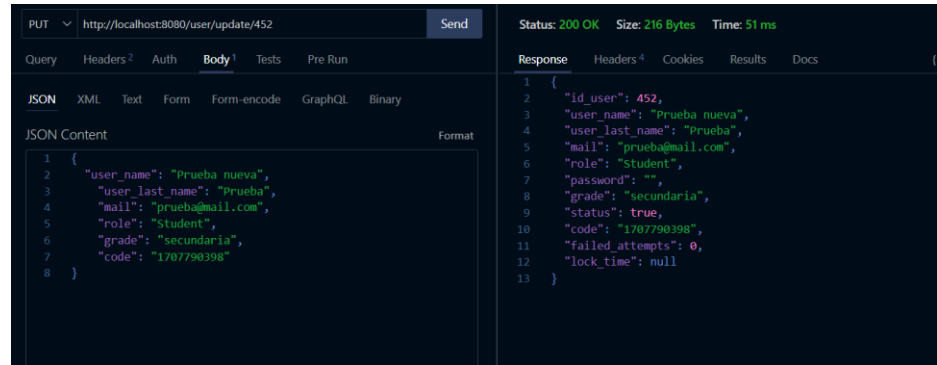
Nota: APIRest para obtener un usuario por id. Elaborado por Steven Pozo, 2025, Quito.

6) Editar usuario

Este permite editar la información del usuario, tal como se muestra en la figura 11.

Figura 11

Edición del nombre del usuario 452



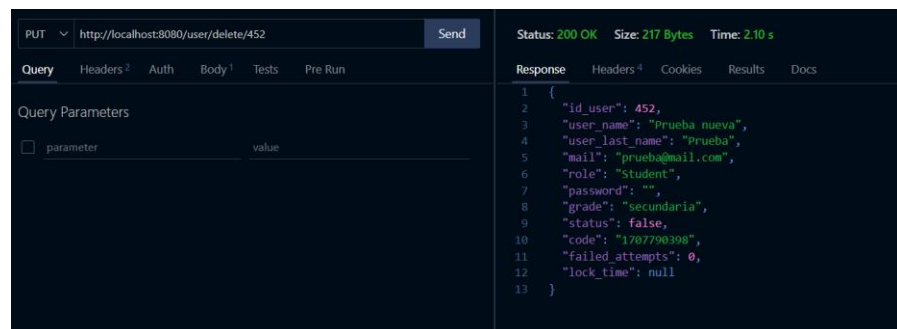
Nota: APIRest para editar los datos de cualquier usuario. Elaborado por Steven Pozo, 2025, Quito.

7) Deshabilitar un usuario

Este permite inhabilitar a al usuario, cambiando su estatus de true a false, tal como se muestra en la figura 11.

Figura 11

Usuario 452 inhabilitado del sistema



Nota: APIRest para inhabilitar a un usuario dentro del sistema. Elaborado por Steven Pozo, 2025, Quito.

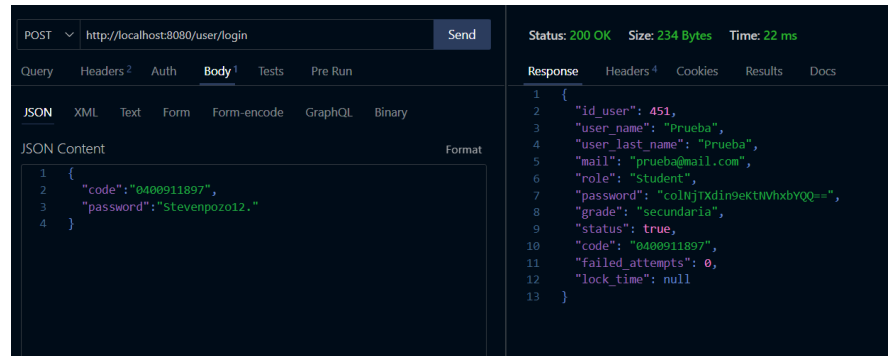
8) Iniciar sesión en el sistema

En el api de inicio de sesión, permite ingresar al sistema, mediante el ingreso de la cédula y de la contraseña, si la verificación es correcta,

este devolverá los datos de dicho usuario registrado y un OK de parte del servidor, tal como se muestra en la figura 12.

Figura 12

Acceso exitoso al sistema



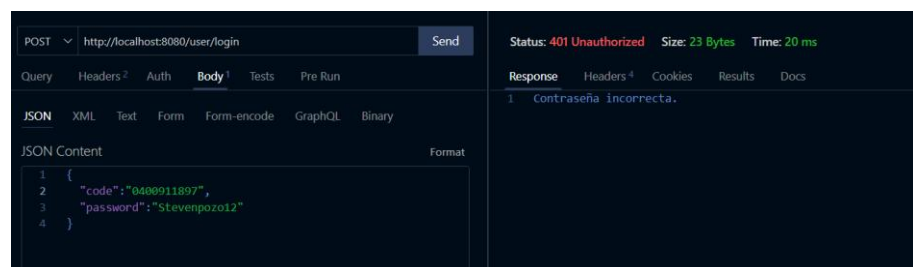
Nota: API Rest para acceder al sistema de manera exitosa.

Elaborado por Steven Pozo, 2025, Quito.

Si en el caso se ingresa mal las credenciales, por defecto se envía un mensaje desde el servidor diciendo que la contraseña está incorrecta, tal como se muestra en la figura 13.

Figura 13

Excepción de credenciales incorrectas

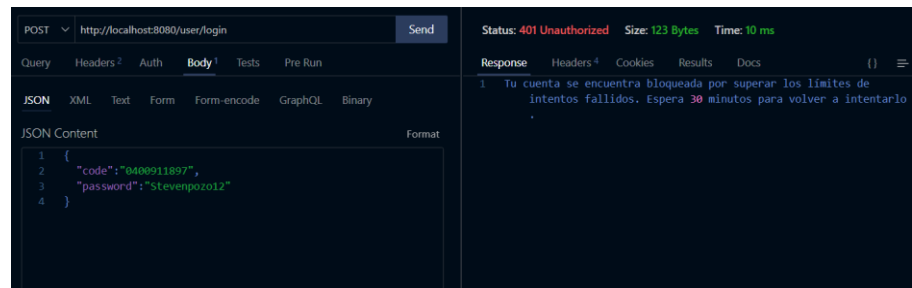


Nota: Se genera una excepción en el inicio de sesión si se ingresa credenciales inválidas. Elaborado por Steven Pozo, 2025, Quito.

Y si los intentos del usuario sobrepasan los 3 intentos, el usuario se bloquea por 30 minutos, esto con la finalidad de evitar ataques de fuerza bruta. Esto se lo muestra en la figura 14.

Figura 14

Acceso del usuario bloqueado



Nota: El usuario se bloquea, después de haber realizado tres intentos erróneos para ingresar al sistema. Elaborado por Steven Pozo, 2025, Quito.

- **Book**

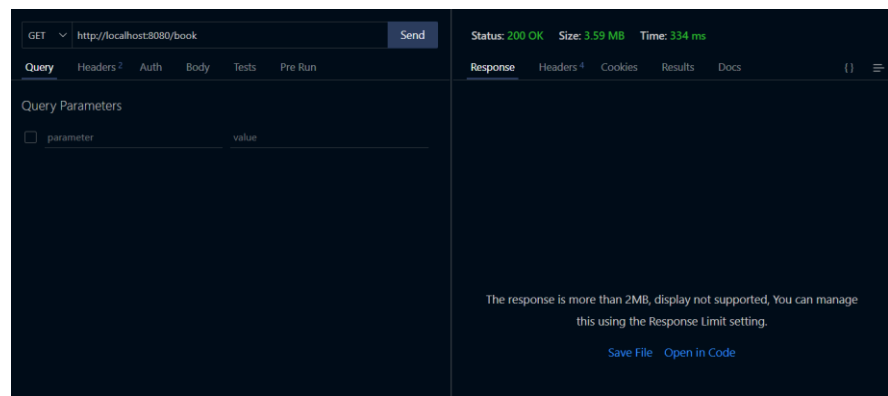
En la gestión de libros, se generaron 7 APIs, las cuales se las describe a continuación.

1) Obtener todos los libros

Para obtener los libros se usa el método GET, junto a su API correspondiente, tal como se muestra en la figura 15.

Figura 15

Obtención de todos los libros



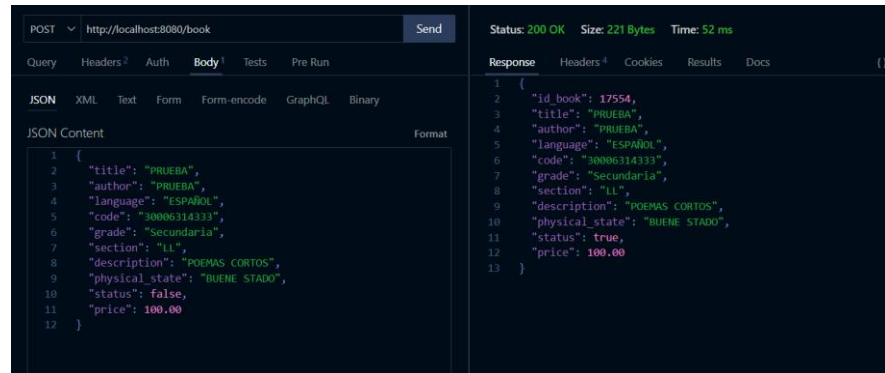
Nota: APIRest para obtener la información de todos los libros, en este caso solo se puede observar la respuesta OK, le cual indica que se obtuvo sin problemas, aunque no se muestre los registros, esto se debe a que Thunder Client solo permite mostrar 2MB en la respuesta, y actualmente la biblioteca cuenta con más de 17 mil libros. Elaborado por Steven Pozo, 2025, Quito.

2) Guardar un nuevo libro

Para guardar un libro se usa el método POST junto a su API, y también con el JSON que inserta los datos del libro, tal como se muestra en la figura 16.

Figura 16

Agregación de un nuevo libro



Nota: APIRest para guardar un nuevo libro con todos sus datos.

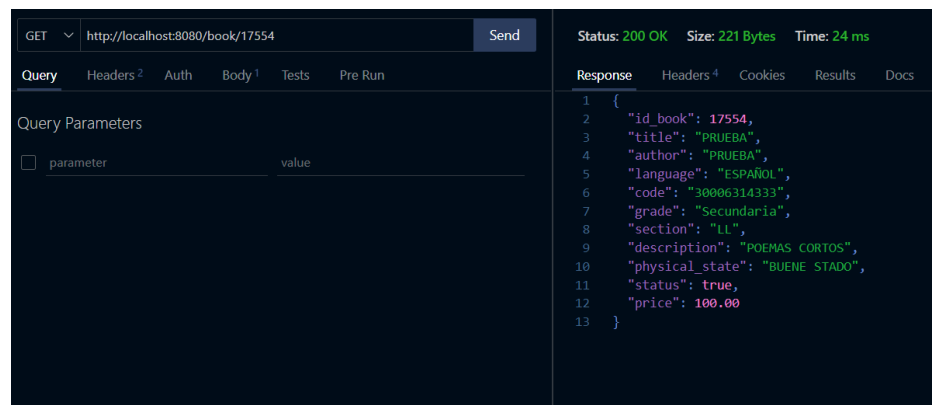
Elaborado por Steven Pozo, 2025, Quito.

3) Obtener un libro por id

Para obtener el libro usando su id, se adjunta el id del libro que se desea buscar a la url base de la API, tal como se muestra en la figura 17.

Figura 17

Obtención de libros por id



Nota: APIRest para obtener un libro por su id. Elaborado por

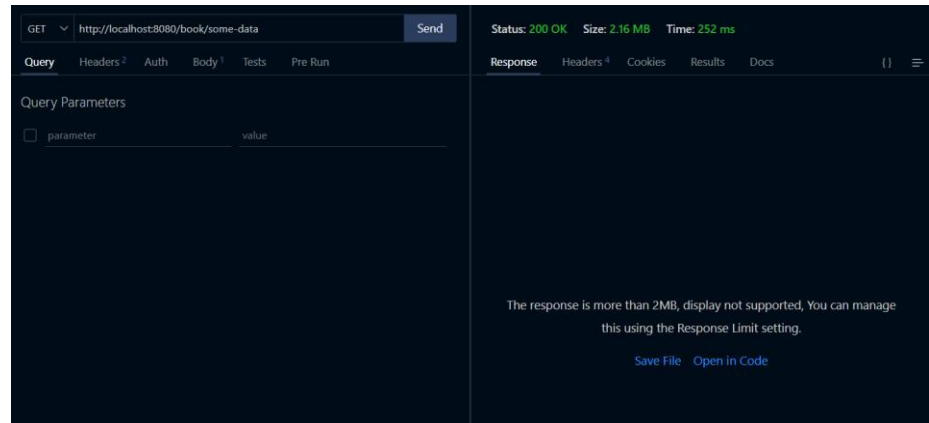
Steven Pozo, 2025, Quito.

4) Obtención de filtro de algunos datos de libro

Este es el mismo caso de obtener todos los libros, por restricciones de Thunder Client, no se puede mostrar debido a su tamaño, pero su respuesta sigue siendo un OK, tal como se muestra en la figura 18.

Figura 18

Obtención de algunos datos de libros mediante filtros



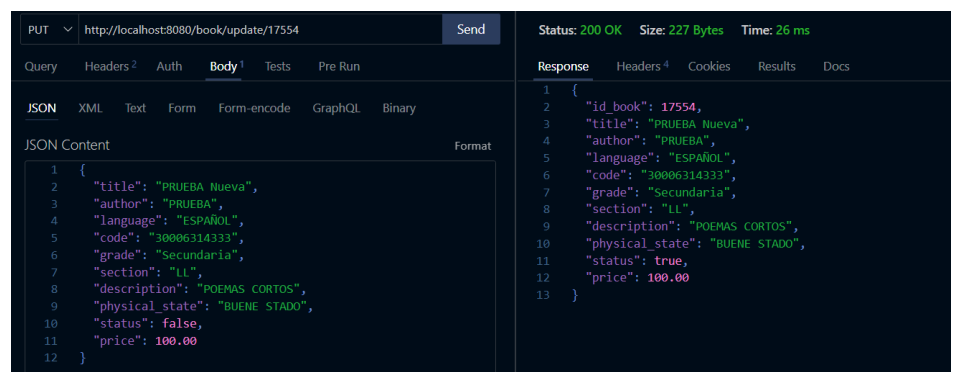
Nota: APIRest para obtener la información de todos los libros, en este caso solo se puede observar la respuesta OK, la cual indica que se obtuvo sin problemas, aunque no se muestre los registros, esto se debe a que Thunder Client solo permite mostrar 2MB en la respuesta, y actualmente la biblioteca cuenta con más de 17 mil libros. Elaborado por Steven Pozo, 2025, Quito.

5) Editar libros

En este caso se editó el título del libro que se ha venido mostrando en estos ejemplos, en la figura 19, se muestra su resultado exitoso.

Figura 19

Edición de datos de un libro



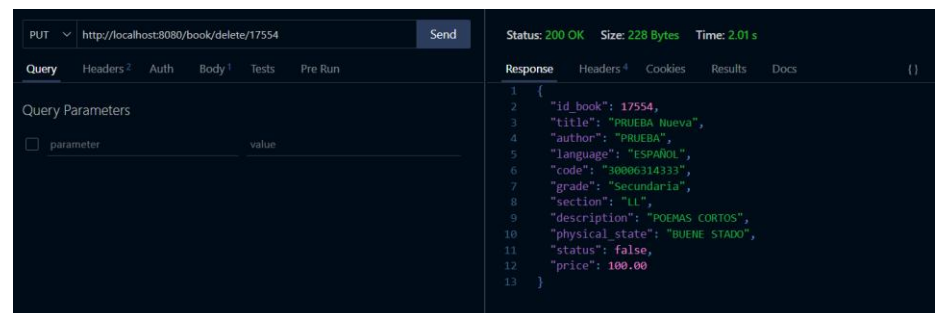
Nota: APIRest para editar la información de un libro. Elaborado por Steven Pozo, 2025, Quito.

6) Inhabilitar un libro

Para sacar a un libro de los registros, ya sea, por viejo o falta de uso se usa esta API, la cual inhabilita al libro, mas no lo elimina, tal como se muestra en la figura 20.

Figura 20

Inhabilitación de un libro del sistema



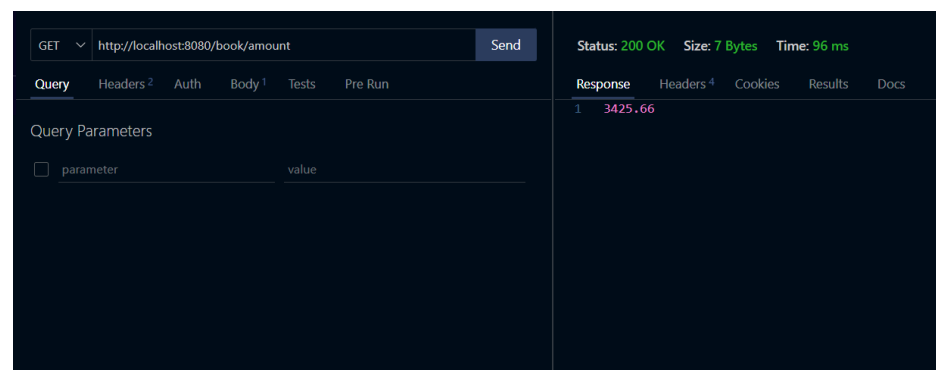
Nota: APIRest para inhabilitar un libro del sistema, más no lo elimina. Elaborado por Steven Pozo, 2025, Quito.

7) Obtención del monto total de los libros de la biblioteca.

Este api permite obtener el monto total de todos los libros del sistema, tal como se muestra en la figura 21, esto para uso de contabilidad de la institución.

Figura 21

Obtención del monto total de todos los libros registrados del sistema



Nota: APIRest para obtener el monto total de todos los libros registrados. Elaborado por Steven Pozo, 2025, Quito.

- **Loan**

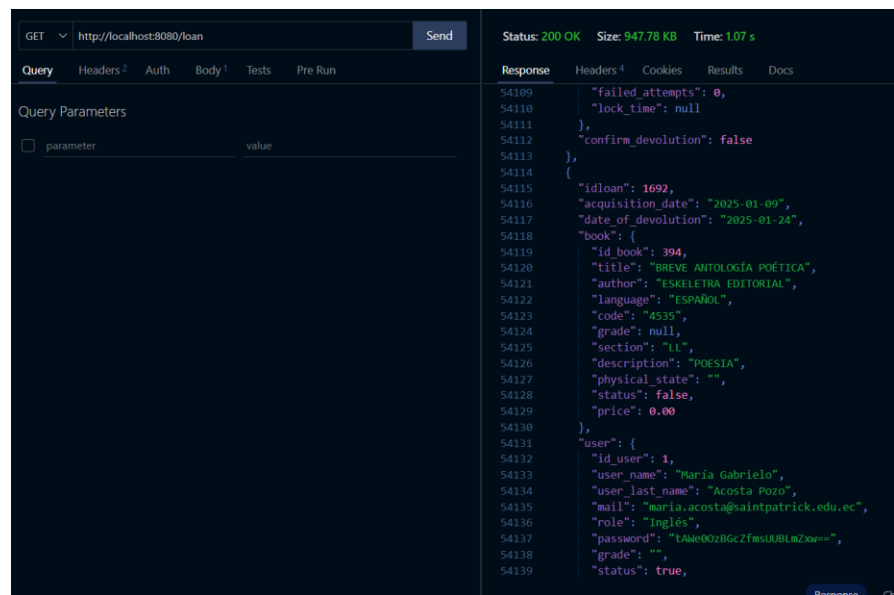
Para la gestión de préstamos se creó 8 APIs, las cuales se los explica a continuación.

1) Obtención de préstamos

Para obtener todos los préstamos se usa el método GET junto a su API correspondiente, tal como se muestra en la figura 22.

Figura 22

Obtención de todos los préstamos realizados



Nota: APIRest para obtener todos los préstamos. Elaborado por Steven Pozo, 2025, Quito.

2) Filtro para obtener datos del usuario, libro y el préstamo

En la figura 23, se muestra como cada préstamo se muestra con datos personalizados.

Figura 23

Préstamos con datos personalizados

| | | | | | |
|--------------------------|--------------------------------------|---------|-------------------|-----------------|--------------|
| GET | http://localhost:8080/loan/some-data | Send | Status: 200 OK | Size: 212.43 KB | Time: 1.28 s |
| Query | Headers ² | Auth | Body ¹ | Tests | Pre Run |
| Query Parameters | | | | | |
| <input type="checkbox"/> | parameter | | value | | |
| Response | Headers ⁴ | Cookies | Results | Docs | |
| 9887 | "author": "EDGARR ALLAN GARCÍA", | | | | |
| 9888 | "acquisition_date": "2025-01-09", | | | | |
| 9889 | "date_of_devolution": "2025-01-24" | | | | |
| 9890 | }, | | | | |
| 9891 | { | | | | |
| 9892 | "id_loan": 1691, | | | | |
| 9893 | "codeUser": "1718556648", | | | | |
| 9894 | "user_name": "María Gabrielo", | | | | |
| 9895 | "user_last_name": "Acosta Pozo", | | | | |
| 9896 | "codeBook": "4200222", | | | | |
| 9897 | "title": "LA EXTRAÑA DAMA INGLESA", | | | | |
| 9898 | "author": "ELIECER CARDENAS", | | | | |
| 9899 | "acquisition_date": "2025-01-09", | | | | |
| 9900 | "date_of_devolution": "2025-01-24" | | | | |
| 9901 | }, | | | | |
| 9902 | { | | | | |
| 9903 | "id_loan": 1692, | | | | |
| 9904 | "codeUser": "1718556648", | | | | |
| 9905 | "user_name": "María Gabrielo", | | | | |
| 9906 | "user_last_name": "Acosta Pozo", | | | | |
| 9907 | "codeBook": "4535", | | | | |
| 9908 | "title": "BREVE ANTOLOGÍA POÉTICA", | | | | |
| 9909 | "author": "ESKELETRA EDITORIAL", | | | | |
| 9910 | "acquisition_date": "2025-01-09", | | | | |
| 9911 | "date_of_devolution": "2025-01-24" | | | | |
| 9912 | } | | | | |
| 9913 | } | | | | |

Nota: APIRest para obtener datos de un préstamo con información personalizada. Elaborado por Steven Pozo, 2025, Quito.

3) Filtro que muestra la bitácora de todos los préstamos

Esta API, permite mostrar la bitácora con respecto a los préstamos del sistema. Esto se lo muestra en la figura 24.

Figura 24

Bitácora del sistema con la información de los préstamos activos e inactivos

| | | | | | |
|--------------------------|---|---------|-------------------|-----------------|--------------|
| GET | http://localhost:8080/loan/binnacle | Send | Status: 200 OK | Size: 561.78 KB | Time: 983 ms |
| Query | Headers ² | Auth | Body ¹ | Tests | Pre Run |
| Query Parameters | | | | | |
| <input type="checkbox"/> | parameter | | value | | |
| Response | Headers ⁴ | Cookies | Results | Docs | |
| 25356 | "title": "LA EXTRAÑA DAMA INGLESA", | | | | |
| 25357 | "language": "ESPAÑOL", | | | | |
| 25358 | "section": "LG", | | | | |
| 25359 | "user_name": "María Gabrielo", | | | | |
| 25360 | "user_last_name": "Acosta Pozo", | | | | |
| 25361 | "mail": "maria.acosta@saintpatrick.edu.ec", | | | | |
| 25362 | "role": "Ingles", | | | | |
| 25363 | "acquisition_date": "2025-01-09", | | | | |
| 25364 | "date_of_devolution": "2025-01-24", | | | | |
| 25365 | "confirm_devolution": false | | | | |
| 25366 | }, | | | | |
| 25367 | { | | | | |
| 25368 | "codeBook": "4535", | | | | |
| 25369 | "author": "ESKELETRA EDITORIAL", | | | | |
| 25370 | "grade": null, | | | | |
| 25371 | "title": "BREVE ANTOLOGÍA POÉTICA", | | | | |
| 25372 | "language": "ESPAÑOL", | | | | |
| 25373 | "section": "LL", | | | | |
| 25374 | "user_name": "María Gabrielo", | | | | |
| 25375 | "user_last_name": "Acosta Pozo", | | | | |
| 25376 | "mail": "maria.acosta@saintpatrick.edu.ec", | | | | |
| 25377 | "role": "Ingles", | | | | |
| 25378 | "acquisition_date": "2025-01-09", | | | | |
| 25379 | "date_of_devolution": "2025-01-24", | | | | |
| 25380 | "confirm_devolution": false | | | | |
| 25381 | } | | | | |
| 25382 | } | | | | |

Nota: APIRest para generar y obtener los datos de la bitácora.

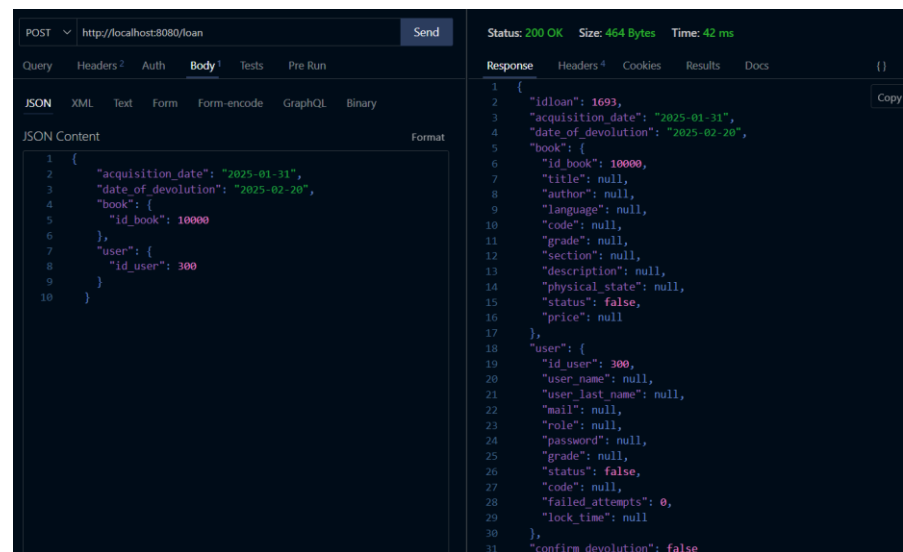
Elaborado por Steven Pozo, 2025, Quito.

4) Realizar un préstamo

En la figura 25, se observa que para crear un préstamo se necesita crear un JSON que envíe el id del usuario y del libro, los cuales se actualizan con false, mientras el préstamo no sea finalizado, para evitar prestar un mismo libro a varios estudiantes sin antes ser devuelto.

Figura 25

Creación de un préstamo



```

POST http://localhost:8080/loan
Content-Type: application/json

{
  "acquisition_date": "2025-01-31",
  "date_of_devolution": "2025-02-20",
  "book": {
    "id_book": 10000
  },
  "user": {
    "id_user": 300
  }
}

Status: 200 OK Size: 464 Bytes Time: 42 ms

{
  "idloan": 1693,
  "acquisition_date": "2025-01-31",
  "date_of_devolution": "2025-02-20",
  "book": {
    "id_book": 10000,
    "title": null,
    "author": null,
    "language": null,
    "code": null,
    "grade": null,
    "section": null,
    "description": null,
    "physical_state": null,
    "status": false,
    "price": null
  },
  "user": {
    "id_user": 300,
    "user_name": null,
    "user_last_name": null,
    "mail": null,
    "role": null,
    "password": null,
    "grade": null,
    "status": false,
    "code": null,
    "failed_attempts": 0,
    "lock_time": null
  },
  "confirm_devolution": false
}

```

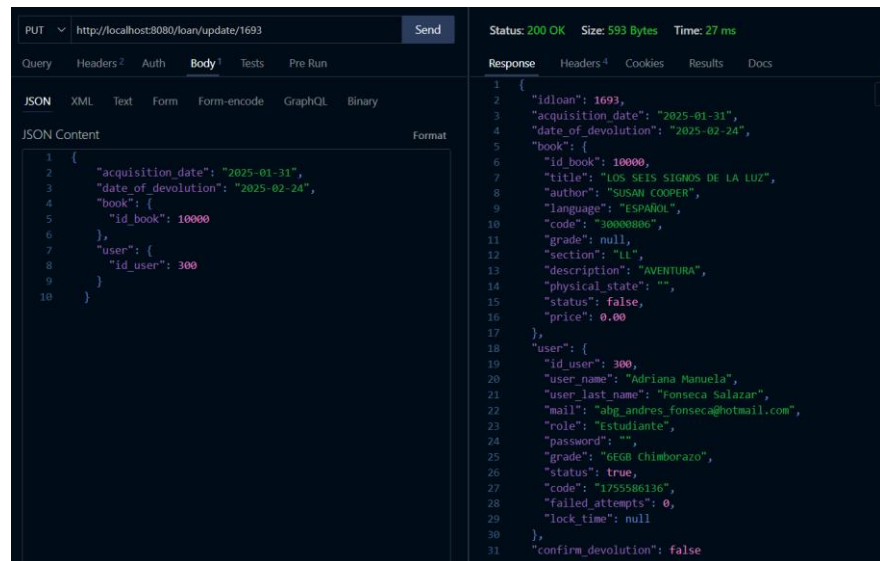
Nota: APIRest para crear un préstamo. Elaborado por Steven Pozo, 2025, Quito.

5) Actualizar un préstamo

Esta API se usa principalmente para actualizar la fecha de devolución del préstamo, tal como se muestra en la figura 26.

Figura 26

Actualización de la fecha del préstamo



The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/loan/update/1693`. The request body is a JSON object with the following content:

```
1 {
2   "acquisition_date": "2025-01-31",
3   "date_of_devolution": "2025-02-24",
4   "book": {
5     "id_book": 10000
6   },
7   "user": {
8     "id_user": 300
9   }
10 }
```

The response is a 200 OK status with 593 bytes and a time of 27 ms. The response body is a JSON object with the following content:

```
1 {
2   "idloan": 1693,
3   "acquisition_date": "2025-01-31",
4   "date_of_devolution": "2025-02-24",
5   "book": {
6     "id_book": 10000,
7     "title": "LOS SEIS SIGNOS DE LA LUZ",
8     "author": "SUSAN COOPER",
9     "language": "ESPAÑOL",
10    "code": "30000806",
11    "grade": null,
12    "section": "LI",
13    "description": "AVENTURA",
14    "physical_state": "",
15    "status": false,
16    "price": 0.00
17  },
18  "user": {
19    "id_user": 300,
20    "user_name": "Adriana Manuela",
21    "user_last_name": "Fonseca Salazar",
22    "mail": "abg_andres_fonseca@hotmail.com",
23    "role": "Estudiante",
24    "password": "",
25    "grade": "6EGB Chimborazo",
26    "status": true,
27    "code": "1755586136",
28    "failed_attempts": 0,
29    "lock_time": null
30  },
31  "confirm_devolution": false
32 }
```

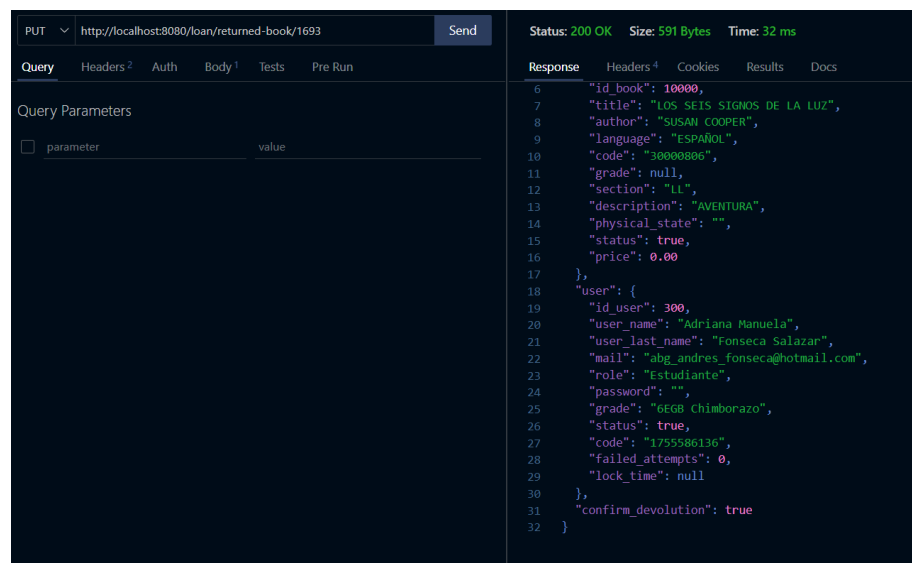
Nota: APIRest para actualizar los datos de un préstamo, específicamente la fecha de devolución. Elaborado por Steven Pozo, 2025, Quito.

6) Devolución de un libro

En este caso, la API se usa para actualizar el estado del préstamo, este pasa a ser true, después de ser devuelto, así también el libro vuelve a habilitarse para realizar otro préstamo. Esto se lo puede ver en la figura 27.

Figura 27

Devolución de un libro y actualización del estatus.



The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/loan/returned-book/1693`. The request body is a JSON object with the following content:

```
1 {
2   "id_book": 10000,
3   "title": "LOS SEIS SIGNOS DE LA LUZ",
4   "author": "SUSAN COOPER",
5   "language": "ESPAÑOL",
6   "code": "30000806",
7   "grade": null,
8   "section": "LI",
9   "description": "AVENTURA",
10  "physical_state": "",
11  "status": true,
12  "price": 0.00
13 },
14 "user": {
15   "id_user": 300,
16   "user_name": "Adriana Manuela",
17   "user_last_name": "Fonseca Salazar",
18   "mail": "abg_andres_fonseca@hotmail.com",
19   "role": "Estudiante",
20   "password": "",
21   "grade": "6EGB Chimborazo",
22   "status": true,
23   "code": "1755586136",
24   "failed_attempts": 0,
25   "lock_time": null
26 },
27 "confirm_devolution": true
28 }
```

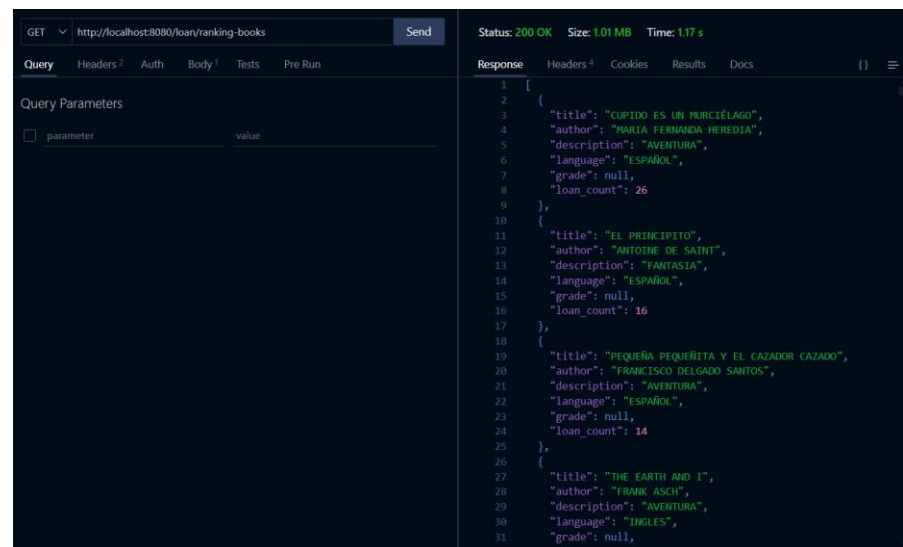
Nota: APIRest para devolver un libro y cambiar el estatus del préstamo. Elaborado por Steven Pozo, 2025, Quito.

7) Ranking de libros

Este api permite mostrar los libros que son más leídos, con el fin de actualizar el inventario y libros dentro de la biblioteca, tal como se muestra en la figura 28.

Figura 28

Obtención de los libros más leídos en la biblioteca



```

GET http://localhost:8080/loan/ranking-books
Send

Query Parameters
parameter value

Response
Status: 200 OK Size: 1.01 MB Time: 1:17 s

1 {
2   {
3     "title": "CUPIDO ES UN MURCIÉLAGO",
4     "author": "MARIA FERNANDA HEREDIA",
5     "description": "AVENTURA",
6     "language": "ESPAÑOL",
7     "grade": null,
8     "loan_count": 26
9   },
10  {
11    "title": "EL PRINCIPITO",
12    "author": "ANTOINE DE SAINT",
13    "description": "FANTASIA",
14    "language": "ESPAÑOL",
15    "grade": null,
16    "loan_count": 16
17  },
18  {
19    "title": "PEQUEÑA PEQUERITA Y EL CAZADOR CAZADO",
20    "author": "FRANCISCO DELGADO SANTOS",
21    "description": "AVENTURA",
22    "language": "ESPAÑOL",
23    "grade": null,
24    "loan_count": 14
25  },
26  {
27    "title": "THE EARTH AND I",
28    "author": "FRANK ASCH",
29    "description": "AVENTURA",
30    "language": "INGLES",
31    "grade": null,
  
```

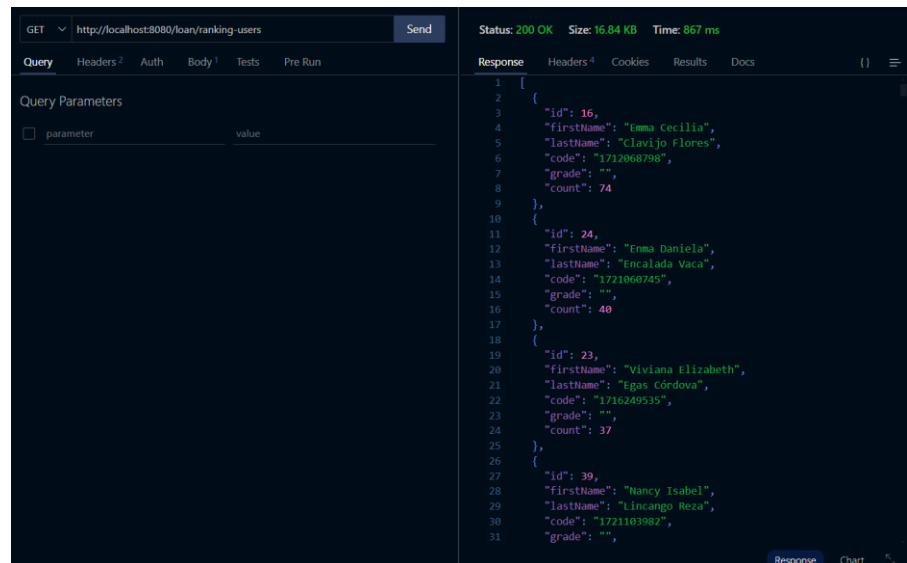
Nota: APIRest para obtener los libros más leídos. Elaborado por Steven Pozo, 2025, Quito.

8) Ranking de usuarios

En la figura 29, se observa como de la misma manera que el ranking de libros, se obtiene el ranking de usuarios, en este caso se muestra los usuarios que más han leídos libros.

Figura 29

Ranking de los usuarios que más han leído un libro



Nota: APIRest para obtener los usuarios que han leído más libros.

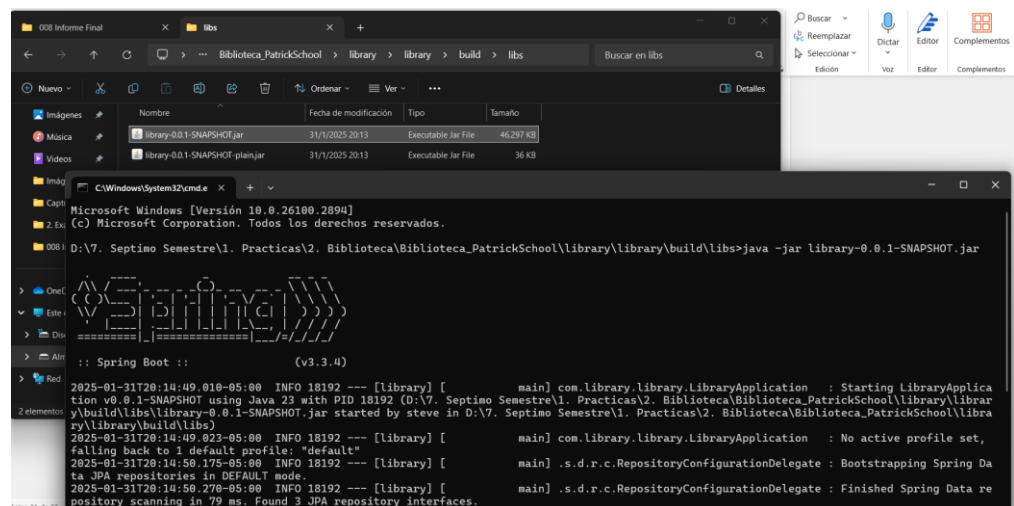
Elaborado por Steven Pozo, 2025, Quito.

- **Ejecución del backend**

Para la ejecución del backend se crea un usa el comando java -jar. El cual ejecuta el servidor desde un ejecutable accesible desde cualquier computador. Esto se lo muestra en la figura 30.

Figura 30

Ejecutable del backend



Nota: Ejecutable del backend y ejecución de este usando java -jar mediante el CMD de Windows. Elaborado por Steven Pozo, 2025, Quito.

2.4. Manipulación del sistema operativo Windows

En este punto se interactuó con las tareas del sistema, como es el caso del Programador de tareas de Windows, donde se necesita crear archivos .bat para poder indicar al sistema que haga una tarea especializada. En este caso se crearon dos archivos .bat, uno centrado en ejecutar el backend y el frontend de manera local, mediante un acceso directo y el otro enfocado en ejecutar una copia de seguridad de la base de datos. A continuación, en la figura 31 y 32 se muestran dichos archivos.

Figura 31

Archivo .bat para gestionar el arranque del aplicativo

```
1 @echo off
2
3 :: Inicia el frontend en segundo plano
4 start javaw -jar "C:\ruta\Biblioteca-1.0-SNAPSHOT-jar-with-dependencies.jar"
5
6 :: Espera un momento para asegurar que el frontend esté ejecutándose
7 timeout /t 2
8
9 :: Inicia el backend en segundo plano (sin consola)
10 start javaw -jar "C:\ruta\library-0.0.1-SNAPSHOT.jar"
11
12 :: Monitorea si el proceso del frontend sigue ejecutándose
13 :monitor
14 tasklist /FI "IMAGENAME eq javaw.exe" | findstr /i "Biblioteca-1.0-SNAPSHOT-jar-with-dependencies.jar" > nul
15 if %ERRORLEVEL% equ 0 (
16     :: Si el frontend está ejecutándose, espera 5 segundos y vuelve a comprobar
17     timeout /t 5
18     goto monitor
19 ) else (
20     :: Si el frontend se cerró, termina el backend
21     taskkill /F /IM "javaw.exe" /T
22     echo El frontend se cerró. El backend también se detuvo.
23 )
24
25 exit
```

Nota: Archivo .bat que arranca el backend y el frontend al mismo tiempo, para generar el uso de aplicativo de escritorio. Elaborado por Steven Pozo, 2025, Quito.

En la figura 31, se observa que el archivo .bat, permite arrancar el front y el back al mismo tiempo, para que el servicio funcione a la par del cliente.

Figura 32

Archivo .bat de backup de la base de datos

```
@echo off
:: Obtiene la fecha en formato YYYYMMDD
for /f "tokens=2-4 delims=/ " %a in ('echo %date%') do set fecha=%c%%b%%a

cd C:\Program Files\MySQL\MySQL Server 9.0\bin
mysqldump -u root --password=170311 library > C:\Backup\library_%fecha%.sql
```

Nota: Archivo .bat que permite generar una copia de seguridad de la base de datos de manera local. Elaborado por Steven Pozo, 2025, Quito.

En la figura 32, se observa que se extrae la fecha del sistema para añadirlo al backup de la base de datos local generado.

2.5.Actividades diarias

Como último punto de este desarrollo, se presentan las actividades realizadas, sus horas empleadas y la fecha de realización de cada una de ellas. Estas se reflejan en la tabla 1.

Tabla 1

Actividades diarias

| Actividad | Fecha de inicio | Fecha final | Hora inicio | Hora final | Detalles | Nro. horas | Horas |
|---|-----------------|-------------|-------------|------------|---|------------|-------|
| introducción del proyecto y documentación de casos de usos V1.0.0 | 30/09/2024 | 4/10/2024 | 8:00 | 14:00 | Configuración de Redmin VPN, creación de usuario y acceso a la máquina virtual compartida | 6 | 30 |
| | | | 8:00 | 14:00 | Capacitación y manipulación experimental de APIS HICKCENTER - Fase I | 6 | |
| | | | 8:00 | 14:00 | Capacitación y manipulación experimental de APIS HICKCENTER - Fase II Introducción al proyecto de biblioteca | 6 | |

| | | | | | | | |
|---|------------|------------|-------|-------|--|---|----|
| | | | 8:00 | 14:00 | Planificación de actividades, selección de framework para el Backend y Frontend | 6 | |
| | | | 8:00 | 14:00 | Casos de usos de los RF del sistema | 6 | |
| Creación de la base de datos. Introducción a Java Spring Boot, creación de proyecto, creación de arquitectura en capas e inicio de construcción de APIs básicas CRUD. | 7/10/2024 | 10/10/2024 | 8:00 | 14:00 | Análisis y creación de la base de datos, instalación y configuración de Mysql, Creación de diagrama ER del sistema | 6 | 30 |
| | | | 8:00 | 14:00 | Creación de proyecto Springboot, adición de dependencias y configuración del JDK Java del sistema | 6 | |
| | | | 8:00 | 14:00 | Creación de la arquitectura en capas del backend del sistema, conexión a la base de datos mysql | 6 | |
| | | | 8:00 | 14:00 | Codificación del CRUD en la capa de modelo, Repositorio, Servicio y Controlador de la entidad User | 6 | |
| Continuación de construcción de | 14/10/2024 | 18/10/2024 | 11:00 | 17:00 | Codificación del CRUD en la capa de modelo, Repositorio, Servicio y | 6 | 30 |

| | | | | | | | |
|--|------------|------------|-------|-------|---|---|----|
| APIS y validación de datos | | | | | Controlador de la entidad Book | | |
| | | | 11:00 | 17:00 | Codificación del CRUD en la capa de modelo, Repositorio, Servicio y Controlador de la entidad Loan. Creación de la capa de Utils para validar datos de entrada. | 6 | |
| | | | 11:00 | 17:00 | Creación e implementación de métodos para validar los campos de user - Fase I | 6 | |
| | | | 11:00 | 17:00 | Creación e implementación de métodos para validar los campos de user - Fase II | 6 | |
| | | | 11:00 | 17:00 | Creación e implementación de métodos para validar los campos de book - Fase I | 6 | |
| Continuación con la validación de datos. | 21/10/2024 | 25/10/2024 | 8:00 | 14:00 | Creación e implementación de métodos para validar los campos de book - Fase II | 6 | 30 |
| | | | 8:00 | 14:00 | Creación e implementación de métodos para validar los campos de loan | 6 | |
| | | | 8:00 | 14:00 | Método de encriptación | 6 | |

| | | | | | | | |
|--|-----------|------------|-------|-------|--|---|----|
| | | | | | simétrica AES para guardar las contraseñas de los administradores. | | |
| | | | 8:00 | 14:00 | Filtración de datos en User y encriptación de datos APIs de user | 6 | |
| | | | 8:00 | 14:00 | Construcción de API para Logearse al sistema, tanto para administrativos, como para los que no lo son. | 6 | |
| Cambios y construcción de nuevas APIs necesitadas en el FrontEnd | | 31/10/2024 | 7:00 | 11:00 | Filtrado de datos y formateo de fechas en book. | 4 | 18 |
| | | | 15:00 | 19:00 | Filtrado de datos y cambio de tipo de datos de loan. | 4 | |
| | | | 7:00 | 11:00 | Transformación y filtrado de datos dentro de la respuesta en APIs de users. | 4 | |
| | | | 8:00 | 14:00 | Correcciones de validaciones de APIs. | 6 | |
| Cambios y construcción de nuevas APIs necesitadas en el FrontEnd - Fase II | 5/11/2024 | 8/11/2024 | 15:00 | 19:00 | Construcción y validación de APIs (ranks users y books) | 4 | 18 |
| | | | 7:00 | 11:00 | Validaciones de APIs faltantes | 4 | |
| | | | 15:00 | 19:00 | Creación de respaldo.bat para la base de datos | 4 | |

| | | | | | | | |
|---|------------|------------|-------|-------|---|---|----|
| Backup de la base de datos | | | 13:00 | 19:00 | Reestructuración del archivo.bat del respaldo de la base de datos. | 6 | |
| Refactorización de APIS (si es necesario) | 11/11/2024 | 15/11/2024 | 7:00 | 11:00 | Inicio de documento de SRS | 4 | 22 |
| | | | 15:00 | 19:00 | Continuación del documento SRS | 4 | |
| | | | 7:00 | 11:00 | Finalización del documento SRS | 4 | |
| | | | 15:00 | 19:00 | Reunión con el supervisor | 4 | |
| | | | 8:00 | 14:00 | Acompañamiento al Front | 6 | |
| Pruebas Unitarias | 18/11/2024 | 22/11/2024 | 7:00 | 11:00 | Corrección y ajuste de requisitos | 4 | 22 |
| | | | 15:00 | 19:00 | Corrección y ajuste de requisitos Fase II | 4 | |
| | | | 7:00 | 11:00 | Corrección de casos de uso | 4 | |
| | | | 15:00 | 19:00 | Creación de Unit test User y Book | 4 | |
| | | | 8:00 | 14:00 | Creación de Unit test Loan | 6 | |
| Pruebas de vulnerabilidad | 25/11/2024 | 29/11/2024 | 7:00 | 11:00 | Creación de Unit test Loan II | 4 | 22 |
| | | | 15:00 | 19:00 | Corrección de 2 Unit test Loan | 4 | |
| | | | 7:00 | 11:00 | Análisis de control de acceso (Autenticación y Autorización) | 4 | |
| | | | 15:00 | 19:00 | Análisis de control de acceso (Autenticación y Autorización) -Fase II | 4 | |
| | | | 8:00 | 14:00 | Análisis de control de acceso | 6 | |

| | | | | | | | |
|---|------------|------------|-------|-------|---|---|-----|
| | | | | | (Autenticación y Autorización) - Fase III | | |
| Pruebas de vulnerabilidad - Fase II | 2/12/2024 | 5/12/2024 | 7:00 | 11:00 | Cifrado de datos | 4 | 16 |
| | | | 15:00 | 19:00 | Manejo de errores y excepciones | 4 | |
| | | | 7:00 | 11:00 | Pruebas de persistencia de datos | 4 | |
| | | | 15:00 | 19:00 | Respaldo ante fallos y recuperación ante desastres | 4 | |
| Carga de información inicial al sistema | 16/12/2024 | 20/12/2024 | 7:00 | 11:00 | Transformación de datos e inserción de datos a la base de datos mysql. | 4 | 18 |
| | | | 14:00 | 18:00 | Configuración del .bat del respaldo de la bd en el programador de tareas windows. | 4 | |
| | | | 15:00 | 19:00 | Corrección de LocalTime en la base de datos y servidor de Springboot | 4 | |
| | | | 8:00 | 14:00 | Crear un archivo.bat de Windows, que genere un ejecute y acceso directo al sistema de la biblioteca, tal que ejecute el backend ejecutable .jar y el frontend ejecutable .jar, al mismo tiempo. | 6 | |
| TOTAL | | | | | | | 250 |

3. CONCLUSIONES

- A través del desarrollo de las prácticas preprofesionales, se fortalecieron conocimientos técnicos como el uso del framework Spring Boot, la manipulación de bases de datos relacionales con MySQL y la creación de scripts en Windows. Estos elementos permitieron construir un sistema robusto y funcional, aplicando conceptos teóricos adquiridos durante la formación universitaria en un entorno práctico y profesional.
- La implementación de una arquitectura en capas en el backend favoreció la organización del código, el desacoplamiento de componentes y la facilidad de mantenimiento del sistema. Además, la creación de documentación completa, incluyendo un documento SRS y diagramas de casos de uso, garantizó la claridad en los requisitos del sistema y la alineación con los objetivos del cliente.
- El sistema desarrollado no solo cumplió su objetivo de optimizar la gestión de la biblioteca escolar, sino que también generó satisfacción tanto en los usuarios como en el cliente. Adicionalmente, la experiencia permitió al practicante adaptarse a dinámicas de trabajo en equipo, resolver problemas reales y ganar confianza en sus capacidades profesionales dentro de un entorno laboral.

4. RECOMENDACIONES

- Es de vital importancia llevar un registro de las actividades que se deben realizar por días, esto con el fin de mantener una línea de trabajo ordenada, tal como lo especifico la tutora académica encarga.
- Realizar las actividades empresariales con respeto, ya que el estudiante es el reflejo de la Universidad y con ello dar la mejor impresión de profesionalismo.
- Como futura recomendación para la institución, es realizar más convenios con otras empresas, para que los estudiantes puedan tener más apertura para realizar prácticas preprofesionales en el tiempo que se estipula el proceso de prácticas. Así también, para que el estudiante adquiriera conocimientos de calidad que sean de ayuda para la institución y ejercer de la mejor manera a nivel profesional.