

# **Sistema de biblioteca**

TRABALHO DE BACK-END FRAMEWORKS

# Membros

Aline Bianca Arantes da Silva – 01617756

- Documentadora e Desenvolvedora

Emerson Wallace Barcelos de Araújo – 01609132

- Gerente de configuração e Desenvolvedor

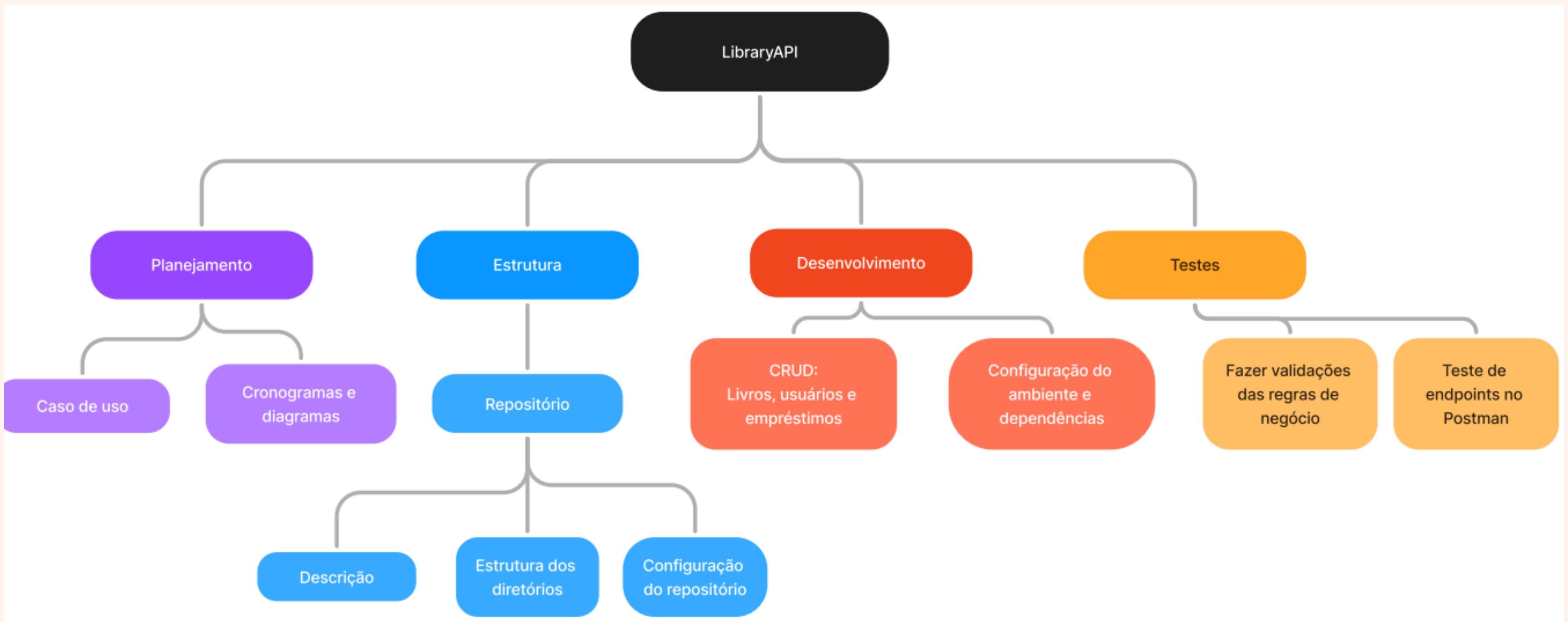
Maria Karolina Rodrigues de Andrade – 01631286

- Documentadora e Scrum Master

# Sumário

- Estrutura Analítica do Projeto
- Custos e Investimentos
- Cronograma de Gantt
- Trello
- Fluxo de trabalho do repositório
- Caso de Uso
- Extratos do código
- Demonstração de Uso
- Lições Aprendidas
- Referência

# Estrutura Analítica do Projeto



# Custos e Investimentos

<b>NOME</b>	<b>Maria Karolina Rodrigues de Andrade</b>	
<b>CARGO</b>	<b>Scrum Master</b>	
VALOR POR HORA	R\$ 56,00	Inserir um valor médio
TOTAL DE HORAS NO MÊS	160 Horas	Horas mensais dedicadas
SALÁRIO BASE	R\$ 8.960,00	Salário base do funcionário
IMPOSTOS TOTAIS (% MÉDIO)	20% Média dos impostos sobre salário	
IMPOSTOS TOTAIS A PAGAR	R\$ 1.792,00	Valor dos impostos sobre salário
BENEFÍCIOS (VR, VT, SAÚDE)	R\$ 896,00	Benefícios ao funcionário
APROVISIONAMENTOS	R\$ 139,78	Processos, seguros, despesas
<b>CUSTO TOTAL MENSAL</b>	<b>R\$ 11.787,78</b>	<i>Custo total para a empresa</i>

<b>NOME</b>	<b>Emerson Wallace Barcelos de Araújo</b>	
<b>CARGO</b>	<b>Desenvolvedor</b>	
VALOR POR HORA	R\$ 35,00	Inserir um valor médio
TOTAL DE HORAS NO MÊS	160 Horas	Horas mensais dedicadas
SALÁRIO BASE	R\$ 5.600,00	Salário base do funcionário
IMPOSTOS TOTAIS (% MÉDIO)	20% Média dos impostos sobre salário	
IMPOSTOS TOTAIS A PAGAR	R\$ 1.120,00	Valor dos impostos sobre salário
BENEFÍCIOS (VR, VT, SAÚDE)	R\$ 560,00	Benefícios ao funcionário
APROVISIONAMENTOS	R\$ 87,36	Processos, seguros, despesas
<b>CUSTO TOTAL MENSAL</b>	<b>R\$ 7.367,36</b>	<i>Custo total para a empresa</i>

<b>NOME</b>	<b>Aline Bianca Arantes da Silva</b>	
<b>CARGO</b>	<b>Desenvolvedora</b>	
VALOR POR HORA	R\$ 35,00	Inserir um valor médio
TOTAL DE HORAS NO MÊS	160 Horas	Horas mensais dedicadas
SALÁRIO BASE	R\$ 5.600,00	Salário base do funcionário
IMPOSTOS TOTAIS (% MÉDIO)	20% Média dos impostos sobre salário	
IMPOSTOS TOTAIS A PAGAR	R\$ 1.120,00	Valor dos impostos sobre salário
BENEFÍCIOS (VR, VT, SAÚDE)	R\$ 560,00	Benefícios ao funcionário
APROVISIONAMENTOS	R\$ 87,36	Processos, seguros, despesas
<b>CUSTO TOTAL MENSAL</b>	<b>R\$ 7.367,36</b>	<i>Custo total para a empresa</i>

# Custos e Investimentos

LICENÇAS DE SOFTWARE			
Microsoft Visual Studio	R\$	375,00	Valor mensal
MySQL Enterprise	R\$	420,00	Valor mensal
Postman	R\$	350,00	Valor mensal
Figma Premium	R\$	1.350,00	Valor mensal
<b>CUSTO TOTAL MENSAL</b>	<b>R\$</b>	<b>2.495,00</b>	<i>Custo total para a empresa</i>

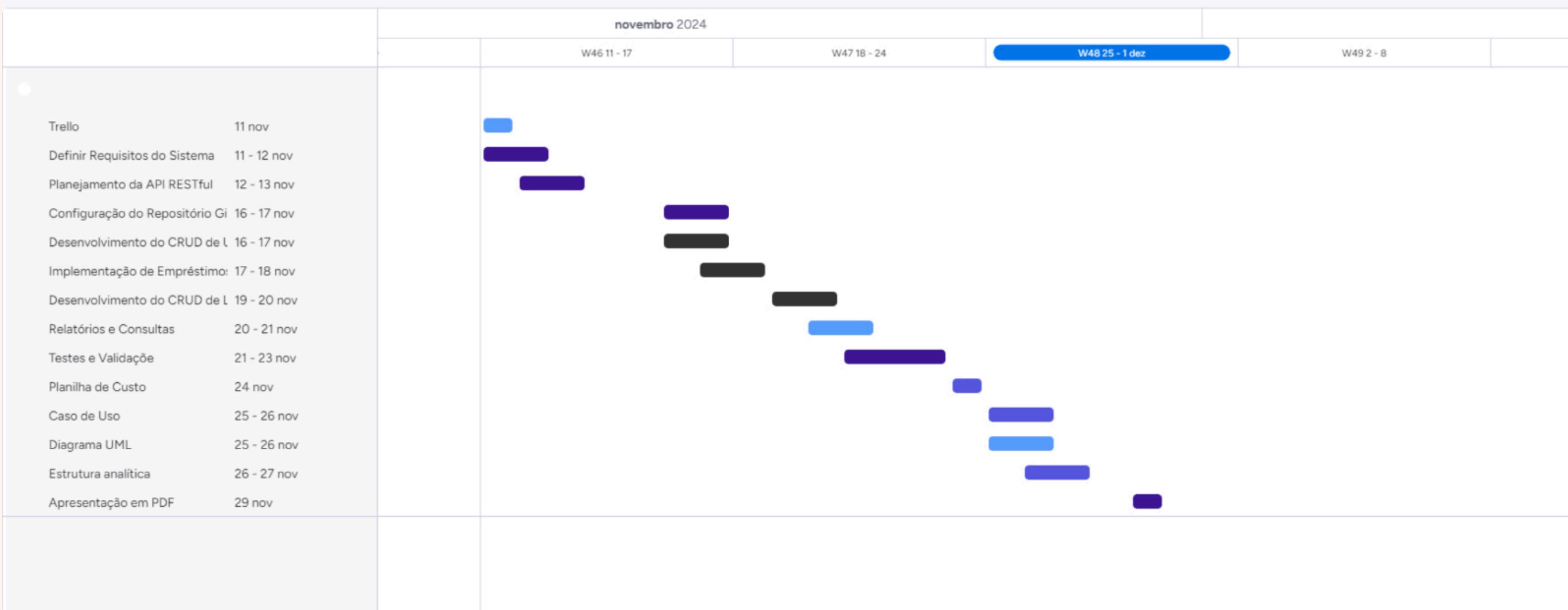
ASSINATURA DE SERVIÇOS			
Udemy Business	R\$	1.000,00	Valor mensal
GitHub Enterprise	R\$	576,00	Valor mensal
Amazon Web Services	R\$	3.000,00	Valor mensal
Trello	R\$	86,00	Valor mensal
Office 365	R\$	105,00	Valor mensal
<b>CUSTO TOTAL MENSAL</b>	<b>R\$</b>	<b>4.767,00</b>	<i>Custo total para a empresa</i>

INFRAESTRUTURA E HARDWARE			
Link Internet 1Gbps	R\$	500,00	Valor mensal
Roteador TP-Link	R\$	300,00	Valor fixo
Notebooks i7	R\$	7.000,00	Valor fixo
Mesas de escritório	R\$	1.200,00	Valor fixo
Cadeiras de escritório	R\$	2.400,00	Valor fixo
<b>CUSTO TOTAL MENSAL</b>	<b>R\$</b>	<b>11.400,00</b>	<i>Custo total para a empresa</i>

# Cronograma de Gantt

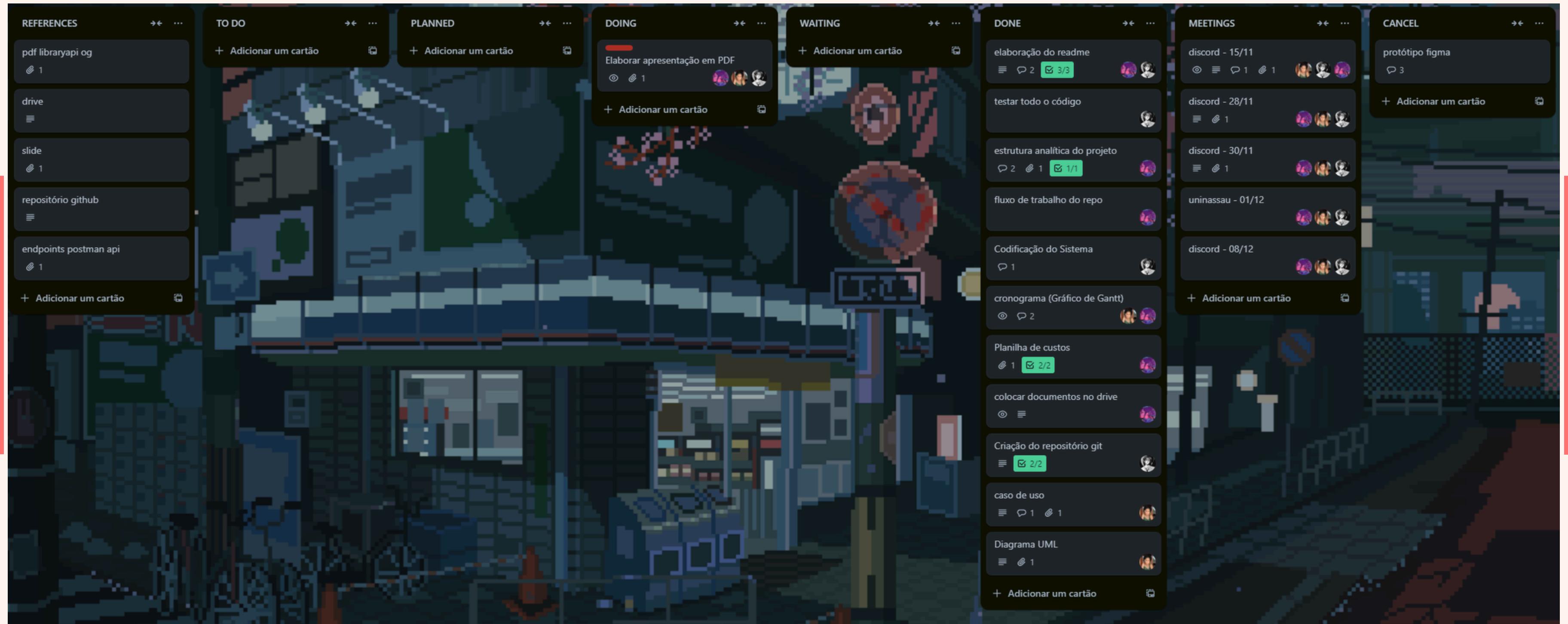
Gantt

01 dezembro, 2024 | 18:13:09



Powered by [monday.com](#)

# Trello



# Casos de Uso

## Recuperar Senha

ID	UC002
Título	Recuperar senha
Autor principal	Usuário
Objetivo	Permitir que o usuário redefina a senha de acesso ao sistema caso a tenha esquecido ou perdido.
Pré-condição	O usuário deve ter um e-mail válido previamente cadastrado no sistema.

## Gerenciamento de Livros

ID	UC003
Título	Gerenciamento de livros
Autor principal	Usuário
Objetivo	Permitir que usuários visualize a lista de livros disponíveis.
Pré-condição	Usuários devem estar cadastrados e autenticados para acessar funcionalidades restritas.
Pós-condição	O usuário consegue visualizar a lista de livros disponíveis ou os detalhes de um livro específico.

## Registro de Empréstimos e Devoluções

ID	UC004
Título	Empréstimos e Devoluções
Autor principal	Usuário
Objetivo	Permitir que o usuário solicite o empréstimo de livros disponíveis no acervo, garantindo o controle de prazos e limites.
Pré-condição	O usuário deve estar autenticado no sistema, o livro solicitado deve estar disponível no acervo, e o usuário não pode ter excedido o limite máximo de empréstimos ativos permitido.
Pós-condição	O empréstimo do livro é registrado com sucesso, incluindo a data prevista para devolução.

## Cadastro e Login

ID	UC001
Título	Login e cadastro
Autor principal	Usuário
Objetivo	Permitir que usuários se cadastrem, realizem login, acessem livros, façam empréstimos e gerenciem suas ações no sistema.
Pré-condição	Usuários devem estar cadastrados e autenticados para acessar funcionalidades restritas.
Pós-condição	Usuários autenticados podem acessar os livros, registrar empréstimos e devoluções

# Fluxo de trabalho do repositório

-o	Commits on Dec 1, 2024
	<b>models</b> itswall committed 4 minutes ago 680216c ⚡ ↗
	<b>Update README.md</b> aline-exe authored 5 minutes ago Verified 4f13bd9 ⚡ ↗
	<b>Create CONTRIBUTING.md</b> aline-exe authored 5 minutes ago Verified 6a3859b ⚡ ↗
-o	Commits on Nov 26, 2024
	<b>diretorios</b> itswall committed last week ac1154c ⚡ ↗
	<b>fix README</b> itswall committed last week ee9c0e8 ⚡ ↗
	<b>update README.md</b> aline-exe authored last week Verified 1fa542a ⚡ ↗
	<b>fix README.md git commands</b> aline-exe authored last week Verified e0050b2 ⚡ ↗
	<b>add README.md content</b> aline-exe authored last week Verified 60f29a7 ⚡ ↗
	<b>Add files via upload</b> aline-exe authored last week Verified a3e0099 ⚡ ↗
	<b>all code for library api</b> itswall committed last week c883c1c ⚡ ↗
-o	Commits on Nov 15, 2024
	Merge pull request #1 from aline-exe/dev

# Work tree

```
D:.  
  .vscode  
  library  
    .mvn  
      wrapper  
    .vscode  
    src  
      main  
        java  
          com  
            libraryApi  
              library  
                config  
                controllers  
                entities  
                  dto  
                exceptions  
                  ex  
                repositories  
                services  
        resources  
          static  
          templates  
      test  
        java  
          com  
            libraryApi  
              library  
      target  
        classes  
          com  
            libraryApi  
              library  
                config  
                controllers  
                entities  
                  dto  
                exceptions  
                  ex  
                repositories  
                services  
        generated-sources  
          annotations  
        generated-test-sources  
          test-annotations  
        maven-status  
          maven-compiler-plugin  
            compile  
              default-compile  
            testCompile  
              default-testCompile  
        test-classes  
          com  
            libraryApi  
              library  
        librarydocs  
          models  
            classes  
            database  
              mindmap  
            sql  
            usecase
```

# **Extratos do código**

(Partes mais importantes)

# Classes dos Rest Controllers

```
1  @RestController
2  @RequestMapping("/books")
3  @RequiredArgsConstructor
4  public class BookController {
5
6      private final BooksService booksService;
7
8      @PostMapping("/create")
9      public ResponseEntity<Books> createNewBook(@RequestBody Books booksDto){
10         return ResponseEntity.status(201).body(booksService.addNewBook(booksDto));
11     }
12
13     @GetMapping
14     public ResponseEntity<List<Books>> findAllBooks(){
15         return ResponseEntity.ok(booksService.findAllBooks());
16     }
17
18     @GetMapping("/{id}")
19     public ResponseEntity<Books> finBook(@PathVariable Long id){
20         return ResponseEntity.ok(booksService.findBook(id));
21     }
22
23     @PatchMapping("/update/{id}")
24     public ResponseEntity<Books> updateBook(@PathVariable Long id, @RequestBody Books booksDto){
25         return ResponseEntity.ok(booksService.updateBook(id, booksDto));
26     }
27
28     @DeleteMapping("/delete/{id}")
29     public ResponseEntity<Void> deleteBook(@PathVariable Long id){
30         booksService.deleteBook(id);
31         return ResponseEntity.ok().build();
32     }
33
34     @GetMapping("/mostBorrowed")
35     public ResponseEntity<List<Books>> findMostBorrowedBooks(){
36         return ResponseEntity.ok(booksService.mostBorroweBooks());
37     }
38
39     @GetMapping("/mostBorrowed")
40     public ResponseEntity<List<Books>> findMostBorrowedBooks(){
41         return ResponseEntity.ok(booksService.mostBorroweBooks());
42     }
43
44 }
45
```

```
1  @RestController
2  @RequestMapping("/user")
3  @RequiredArgsConstructor
4  public class UserController {
5
6      private final UserService userService;
7
8      @PostMapping("/create")
9      public ResponseEntity<User> createNewUser(@Valid @RequestBody User user){
10         return ResponseEntity.status(201).body(userService.addUser(user));
11     }
12
13     @GetMapping
14     public ResponseEntity<List<User>> findAllUsers(){
15         return ResponseEntity.ok(userService.findAllUsers());
16     }
17
18     @GetMapping("/{id}")
19     public ResponseEntity<User> findUser(@PathVariable Long id){
20         return ResponseEntity.ok(userService.findUser(id));
21     }
22
23     @PatchMapping("/update/{id}")
24     public ResponseEntity<User> updateUser(@PathVariable Long id, @RequestBody @Valid User user){
25         return ResponseEntity.ok(userService.updateUser(id, user));
26     }
27
28     @DeleteMapping("/delete/{id}")
29     public ResponseEntity<Void> deleteUser(@PathVariable Long id){
30         userService.deleteUser(id);
31         return ResponseEntity.ok().build();
32     }
33
34     @DeleteMapping("/delete/{id}")
35     public ResponseEntity<Void> deleteUser(@PathVariable Long id){
36         userService.deleteUser(id);
37         return ResponseEntity.ok().build();
38     }
39
40     @DeleteMapping("/delete/{id}")
41     public ResponseEntity<Void> deleteUser(@PathVariable Long id){
42         userService.deleteUser(id);
43         return ResponseEntity.ok().build();
44     }
45 }
```

# Classes dos Rest Controllers

```
1 @RestController
2 @RequestMapping("/borrows")
3 @RequiredArgsConstructor
4 public class BorrowedBooksController {
5
6     private final BorrowedBooksService borrowedBooksService;
7
8     @PostMapping("/newBorrow")
9     public ResponseEntity<BorrowedBooks> borrowNewBook(@RequestBody BorrowedBooksDto borrowedBooksDto) {
10         return ResponseEntity.status(201)
11             .body(borrowedBooksService.borrowNewBook(
12                 borrowedBooksDto.userId(),
13                 borrowedBooksDto.bookId(),
14                 borrowedBooksDto.returnDate()));
15     }
16
17     @GetMapping
18     public ResponseEntity<List<BorrowedBooks>> findAllBorrowedBooks(){
19         return ResponseEntity.ok(borrowedBooksService.findAllBorrowedBooks());
20     }
21
22     @DeleteMapping("/return/{userId}/{bookId}")
23     public ResponseEntity<Void> returnBook(@PathVariable Long userId, @PathVariable Long bookId ){
24         borrowedBooksService.returnBook(userId, bookId);
25         return ResponseEntity.ok().build();
26     }
27 }
```

```
1 @RestController
2 @RequiredArgsConstructor
3 @RequestMapping("/PendingBooks")
4 public class PendingBooksController {
5
6     private final PendingBooksService pendingBooksService;
7
8     @PostMapping("/NewPending")
9     public ResponseEntity<PendingBooks> newPending(@RequestBody PendingBooksDto pendingBooksDto){
10
11         return ResponseEntity.status(201).body(pendingBooksService.newPendingBook(pendingBooksDto.userId(),
12             pendingBooksDto.bookId()));
13     }
14
15     @GetMapping
16     public ResponseEntity<List<PendingBooks>> findAllPendingBooks(){
17         return ResponseEntity.ok(pendingBooksService.findAllPendingBooks());
18     }
19
20     @DeleteMapping("/returnPending/{userId}/{bookId}")
21     public ResponseEntity<Void> returnBook(@PathVariable Long userId, @PathVariable Long bookId ){
22         pendingBooksService.returnPendingBook(userId, bookId);
23         return ResponseEntity.ok().build();
24     }
25 }
26 }
```

# Classes dos Serviços

```
● ● ●  
1  @Service  
2  @RequiredArgsConstructor  
3  public class UserService {  
4  
5      private final UserRepository userRepository;  
6  
7      public User addUser(User user){  
8          return userRepository.save(user);  
9      }  
10     public List<User> findAllUsers(){  
11         return userRepository.findAll();  
12     }  
13  
14     public User findUser(Long id){  
15  
16         return userRepository.findById(id).orElseThrow(  
17             () -> new ContentNotFoundException("Id not Found")  
18         );  
19     }  
20  
21     public User updateUser(Long id, User user){  
22         User userToUpdate = this.findUser(id);  
23  
24         if (user.getEmail() != null){  
25             userToUpdate.setEmail(user.getEmail());  
26         }  
27         if (user.getAddress() != null){  
28             userToUpdate.setAddress(user.getAddress());  
29         }  
30         if (user.getName() != null){  
31             userToUpdate.setName(user.getName());  
32         }  
33         if (user.getNumber() != null) {  
34             userToUpdate.setNumber(user.getNumber());  
35         }  
36  
37         return userRepository.save(userToUpdate);  
38     }  
39  
40     public void deleteUser(Long id){  
41         var user = this.findUser(id);  
42         userRepository.delete(user);  
43  
44         try {  
45             userRepository.delete(user);  
46         }  
47         catch (DataIntegrityViolationException e){  
48             throw new UserAlreadyBorrowed("User cannot be deleted with borrowed books!");  
49         }  
50     }  
51  
52 }  
53  
54 }  
55 }
```

```
● ● ●  
1  @Service  
2  @RequiredArgsConstructor  
3  public class BooksService {  
4  
5      private final BooksRepository booksRepository;  
6  
7      public Books addNewBook(Books books){  
8          return booksRepository.save(books);  
9      }  
10  
11     public List<Books> findAllBooks(){  
12         return booksRepository.findAll();  
13     }  
14  
15     public Books findBook(Long id){  
16  
17         return booksRepository.findById(id).orElseThrow(  
18             () -> new ContentNotFoundException("Id not found")  
19         );  
20     }  
21  
22     public Books updateBook(Long id, Books books){  
23         Books bookToUpdate = this.findBook(id);  
24  
25         if (books.getPublishYear() != null){  
26             bookToUpdate.setPublishYear(books.getPublishYear());  
27         }  
28         if (books.getAuthor() != null){  
29             bookToUpdate.setAuthor(books.getAuthor());  
30         }  
31         if (books.getGenre() != null){  
32             bookToUpdate.setGenre(books.getGenre());  
33         }  
34         if (books.getTitle() != null) {  
35             bookToUpdate.setTitle(books.getTitle());  
36         }  
37         return booksRepository.save(bookToUpdate);  
38     }  
39  
40     public void deleteBook(Long id){  
41         var book = this.findBook(id);  
42         booksRepository.delete(book);  
43     }  
44  
45     public List<Books> mostBorrowedBooks(){  
46         var list = this.findAllBooks();  
47  
48         return list.stream()  
49             .sorted(Comparator.comparing(Books::getBorrowedTimes).reversed())  
50             .collect(Collectors.toList());  
51     }  
52  
53 }  
54 }
```

# Classes dos Serviços

```
1  @Service
2  @RequiredArgsConstructor
3  public class BorrowedBooksService {
4
5      private final BorrowedBooksRepository borrowedBooksRepository;
6      private final BooksService booksService;
7      private final UserService userService;
8
9      public List<BorrowedBooks> findAllBorrowedBooks(){
10         return borrowedBooksRepository.findAll();
11     }
12
13
14     public BorrowedBooks borrowNewBook(Long user_id, Long book_id, LocalDate returnDate){
15
16         var user = userService.findUser(user_id);
17         var book = booksService.findBook(book_id);
18         var borrowPerUser = 0;
19
20         for (BorrowedBooks l: findAllBorrowedBooks()){
21             if (Objects.equals(l.getUser().getId(), user.getId())){
22                 borrowPerUser += 1;
23             }
24             if (borrowPerUser>=2){
25                 throw new MaxBorrows("Users already borrow the max limit");
26             }
27         }
28
29         BorrowedBooks borrowedBooks = new BorrowedBooks();
30         borrowedBooks.setBooks(book);
31         borrowedBooks.setUser(user);
32         borrowedBooks.setReturnDate(returnDate);
33
34         book.setBorrowedTimes(book.getBorrowedTimes()+1);
35         return borrowedBooksRepository.save(borrowedBooks);
36
37     }
38
39     public void returnBook(Long user_id, Long book_id){
40
41         for (BorrowedBooks l : this.findAllBorrowedBooks()){
42             if (Objects.equals(l.getUser().getId(), user_id) && Objects.equals(l.getBooks().getId(), book_id)){
43                 borrowedBooksRepository.delete(l);
44                 return;
45             }
46         }
47     }
48
49     throw new ContentNotFoundException(
50         "No users found."
51     );
52 }
53 }
```

```
1  @Service
2  @RequiredArgsConstructor
3  public class PendingBooksService {
4
5      private final UserService userService;
6      private final BooksService booksService;
7      private final PendingBooksRepository pendingBooksRepository;
8
9      public List<PendingBooks> findAllPendingBooks(){
10         return pendingBooksRepository.findAll();
11     }
12
13     public PendingBooks newPendingBook(Long userId, Long bookId){
14
15         var user = userService.findUser(userId);
16         var book = booksService.findBook(bookId);
17
18         for (PendingBooks e: this.findAllPendingBooks()){
19             if (e.getUser() == user){
20                 throw new MaxBorrows("Limite maximo de emprestimos pendentes foi atingido");
21             }
22         }
23
24         PendingBooks pendingBooks = new PendingBooks();
25         pendingBooks.setUser(user);
26         pendingBooks.setBooks(book);
27
28         return pendingBooksRepository.save(pendingBooks);
29     }
30
31     public void returnPendingBook(Long user_id, Long book_id){
32
33         for (PendingBooks l : this.findAllPendingBooks()){
34             if (Objects.equals(l.getUser().getId(), user_id) && Objects.equals(l.getBooks().getId(), book_id)){
35                 pendingBooksRepository.delete(l);
36                 return;
37             }
38         }
39
40         throw new ContentNotFoundException(
41             "No users found."
42         );
43     }
44 }
45 }
```

# Demonstração de uso

HTTP LibraryApi / Create new user

POST http://localhost:8080/user/create

Params Authorization Headers (8) **Body** Scripts Settings

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

```
1 {
2   "name": "João",
3   "email": "joao@gmail.com",
4   "number": "8198888888",
5   "address": "Rua das Flores, 123"
6 }
```

Body Cookies Headers (8) Test Results 201 Created

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "name": "João",
4   "email": "joao@gmail.com",
5   "number": "8198888888",
6   "address": "Rua das Flores, 123"
7 }
```

# Demonstração de uso

HTTP LibraryApi / Get all users

GET http://localhost:8080/user

Params Authorization Headers (8) Body Scripts Settings

Body Cookies Headers (8) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
1 [  
2 {  
3     "id": 2,  
4     "name": "roberto",  
5     "email": "pato2@gmail.com",  
6     "number": "8198888888",  
7     "address": "muito longe"  
8 },  
9 {  
10    "id": 3,  
11    "name": "João",  
12    "email": "joao@gmail.com",  
13    "number": "8198888888",  
14    "address": "Rua das Flores, 123"  
15 }  
16 ]
```

# Demonstração de uso

The screenshot shows a POSTMAN API client interface. At the top, there's a header bar with the text "HTTP LibraryApi / Get user by ID". Below this, a method dropdown shows "GET" and a URL field contains "http://localhost:8080/user/3". A navigation bar below the URL includes tabs for "Params", "Authorization", "Headers (6)", "Body", "Scripts", and "Settings", with "Headers (6)" currently selected. To the right of the URL, a status indicator shows "200 OK". Below the navigation bar, another set of tabs includes "Body", "Cookies", "Headers (8)", and "Test Results", with "Body" selected. Under the "Body" tab, there are four options: "Pretty", "Raw", "Preview", and "Visualize", with "Pretty" selected. A dropdown menu next to "Pretty" shows "JSON" and "Text", with "JSON" selected. The main content area displays a JSON response with line numbers from 1 to 7. The JSON data is:

```
1 {  
2   "id": 3,  
3   "name": "João",  
4   "email": "joao@gmail.com",  
5   "number": "8198888888",  
6   "address": "Rua das Flores, 123"  
7 }
```

# Demonstração de uso

The screenshot shows a POSTMAN interface with the following details:

- HTTP:** LibraryApi / Update user
- Method:** PATCH
- URL:** http://localhost:8080/user/update/2
- Body (raw JSON):**

```
1 {  
2   "name": "roberto donald",  
3   "email": "roberto@gmail.com",  
4   "number": "8198888888",  
5   "address": "mais longe ainda"  
6 }
```
- Response Status:** 200 OK
- Body (Pretty JSON):**

```
1 {  
2   "id": 2,  
3   "name": "roberto donald",  
4   "email": "roberto@gmail.com",  
5   "number": "8198888888",  
6   "address": "mais longe ainda"  
7 }
```

# Demonstração de uso

The screenshot shows a POST request to `http://localhost:8080/books/create`. The request body is a JSON object representing a book:

```
1 {  
2   "title": "Pride and Prejudice",  
3   "author": "Jane Austen",  
4   "genre": "Romance",  
5   "publishYear": "1813",  
6   "borrowedTimes": 5  
7 }
```

The response status is 201 Created. The response body is identical to the request body:

```
1 {  
2   "id": 3,  
3   "title": "Pride and Prejudice",  
4   "author": "Jane Austen",  
5   "genre": "Romance",  
6   "publishYear": "1813",  
7   "borrowedTimes": 5  
8 }
```

# Demonstração de uso

The screenshot shows a POSTMAN API client interface. At the top, it displays the URL `http://localhost:8080/books`. Below the URL, there are tabs for **Params**, **Authorization**, **Headers (6)**, **Body**, **Scripts**, and **Settings**. The **Body** tab is selected, showing sub-tabs for **Pretty**, **Raw**, **Preview**, and **Visualize**. The **Pretty** tab is active, displaying the JSON response with line numbers. The response body is:

```
1 [  
2 {  
3   "id": 2,  
4   "title": "The Great Gatsby",  
5   "author": "F. Scott Fitzgerald",  
6   "genre": "Fiction",  
7   "publishYear": "1925",  
8   "borrowedTimes": 2  
9 },  
10 {  
11   "id": 3,  
12   "title": "Pride and Prejudice",  
13   "author": "Jane Austen",  
14   "genre": "Romance",  
15   "publishYear": "1813",  
16   "borrowedTimes": 5  
17 }  
18 ]
```

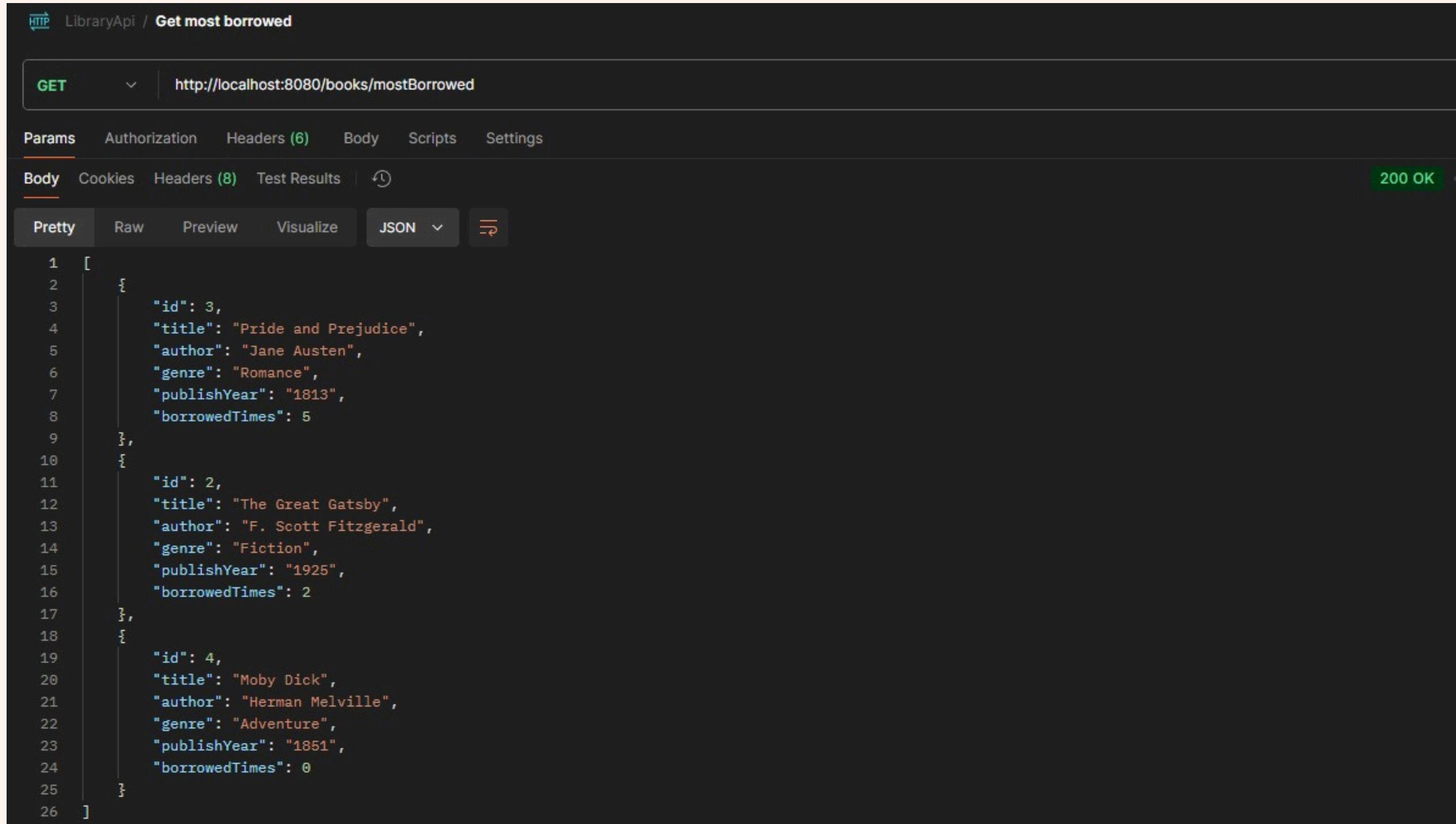
To the right of the response body, a green button indicates a **200 OK** status.

# Demonstração de uso

The screenshot shows a POSTMAN API client interface. At the top, it displays the URL `http://localhost:8080/books/3`. Below the URL, there are tabs for **Params**, **Authorization**, **Headers (6)**, **Body**, **Scripts**, and **Settings**. The **Body** tab is currently selected. On the right side of the interface, a green box indicates a **200 OK** response. The **Body** section has tabs for **Pretty**, **Raw**, **Preview**, and **Visualize**, with **Pretty** being the active tab. The JSON response is displayed in a multi-line code block:

```
1 {  
2   "id": 3,  
3   "title": "Pride and Prejudice",  
4   "author": "Jane Austen",  
5   "genre": "Romance",  
6   "publishYear": "1813",  
7   "borrowedTimes": 5  
8 }
```

# Demonstração de uso



HTTP LibraryApi / Get most borrowed

GET <http://localhost:8080/books/mostBorrowed>

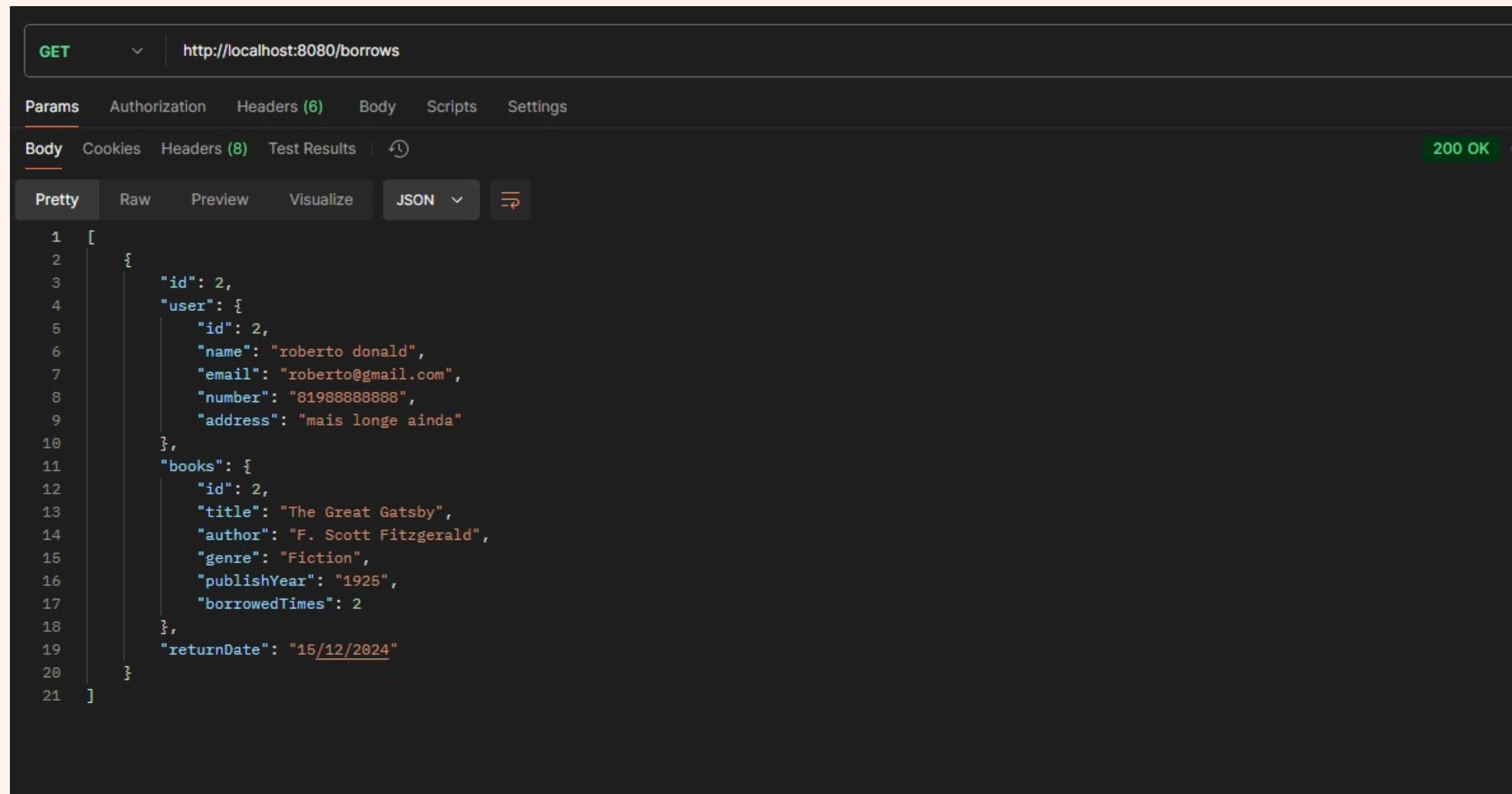
Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (8) Test Results ⏪ 200 OK

Pretty Raw Preview Visualize JSON ⏪

```
1 [  
2 {  
3   "id": 3,  
4   "title": "Pride and Prejudice",  
5   "author": "Jane Austen",  
6   "genre": "Romance",  
7   "publishYear": "1813",  
8   "borrowedTimes": 5  
9 },  
10 {  
11   "id": 2,  
12   "title": "The Great Gatsby",  
13   "author": "F. Scott Fitzgerald",  
14   "genre": "Fiction",  
15   "publishYear": "1925",  
16   "borrowedTimes": 2  
17 },  
18 {  
19   "id": 4,  
20   "title": "Moby Dick",  
21   "author": "Herman Melville",  
22   "genre": "Adventure",  
23   "publishYear": "1851",  
24   "borrowedTimes": 0  
25 }  
26 ]
```

# Demonstração de uso



GET <http://localhost:8080/borrows>

Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (8) Test Results | ⚙️ 200 OK

Pretty Raw Preview Visualize JSON ↻

```
1 [  
2 {  
3     "id": 2,  
4     "user": {  
5         "id": 2,  
6         "name": "roberto donald",  
7         "email": "roberto@gmail.com",  
8         "number": "8198888888",  
9         "address": "mais longe ainda"  
10    },  
11    "books": {  
12        "id": 2,  
13        "title": "The Great Gatsby",  
14        "author": "F. Scott Fitzgerald",  
15        "genre": "Fiction",  
16        "publishYear": "1925",  
17        "borrowedTimes": 2  
18    },  
19    "returnDate": "15/12/2024"  
20}  
21]
```

# Lições Aprendidas

Aprendemos a importância de dividir o trabalho em grupo, definir prioridades e separar atividades para que cada membro tenha um foco. Além disso, a organização do trabalho com o Trello utilizando o método Kanban ajuda muito na organização e do alinhamento da equipe com as atividades de cada um.

# Referências

- YouTube
- GitHub
- ChatGPT
- Figma – EAP
- Documentação Spring Boot
- Monday – Gráfico Gantt

**Obrigado!**