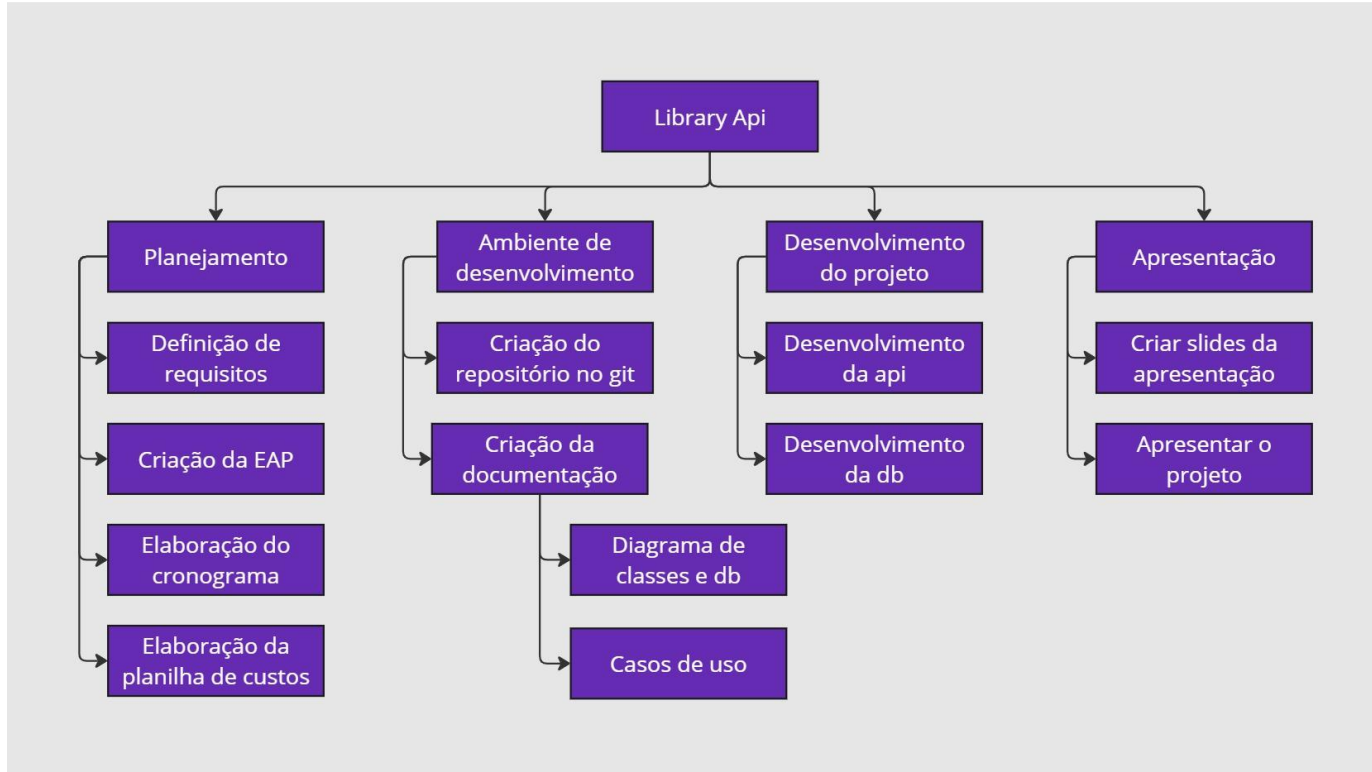


Library Api

Equipe

- Cláudio Cordeiro
- Valdeck Gomes

Estrutura Analítica do Projeto - EAP



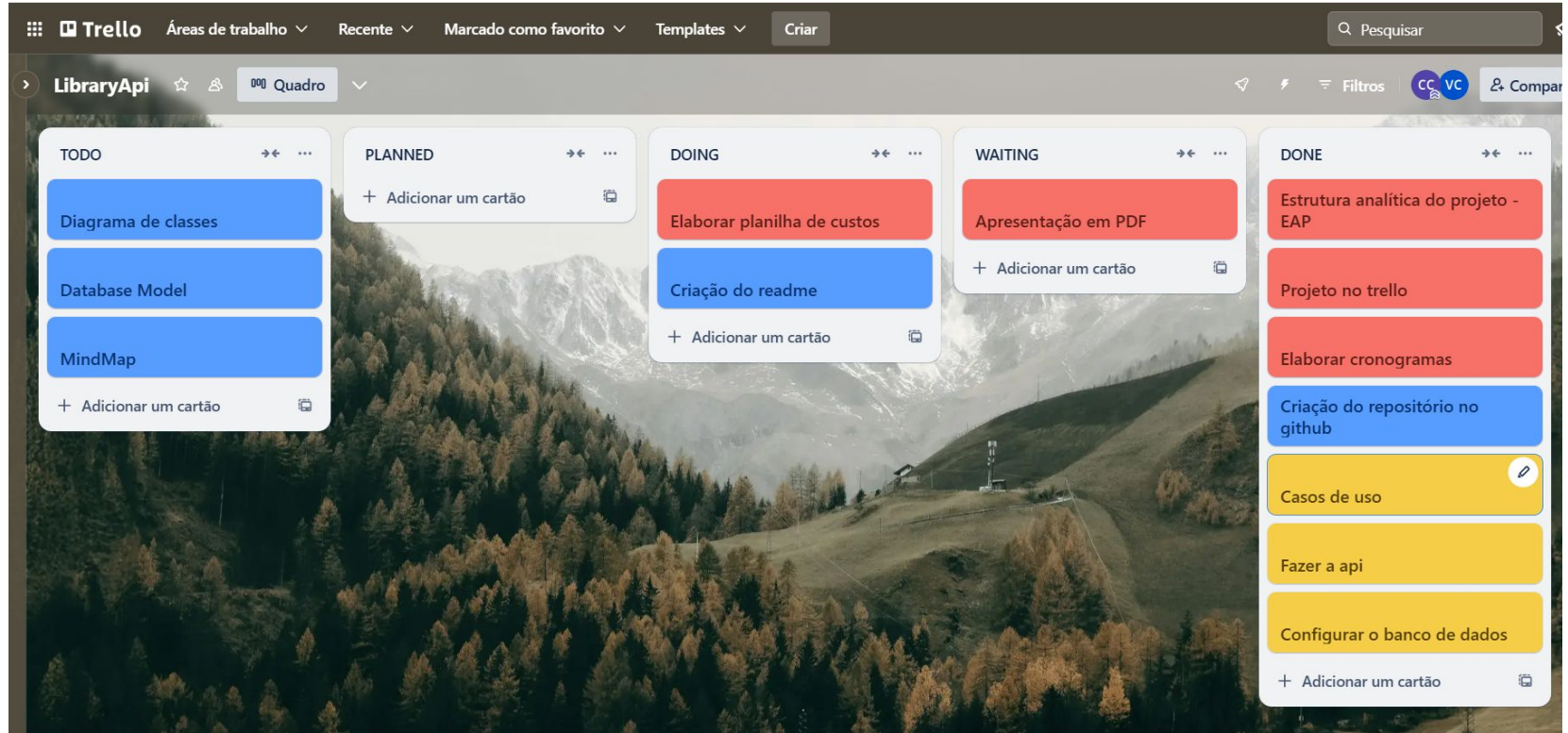
Cronograma com Gantt

[illegible]

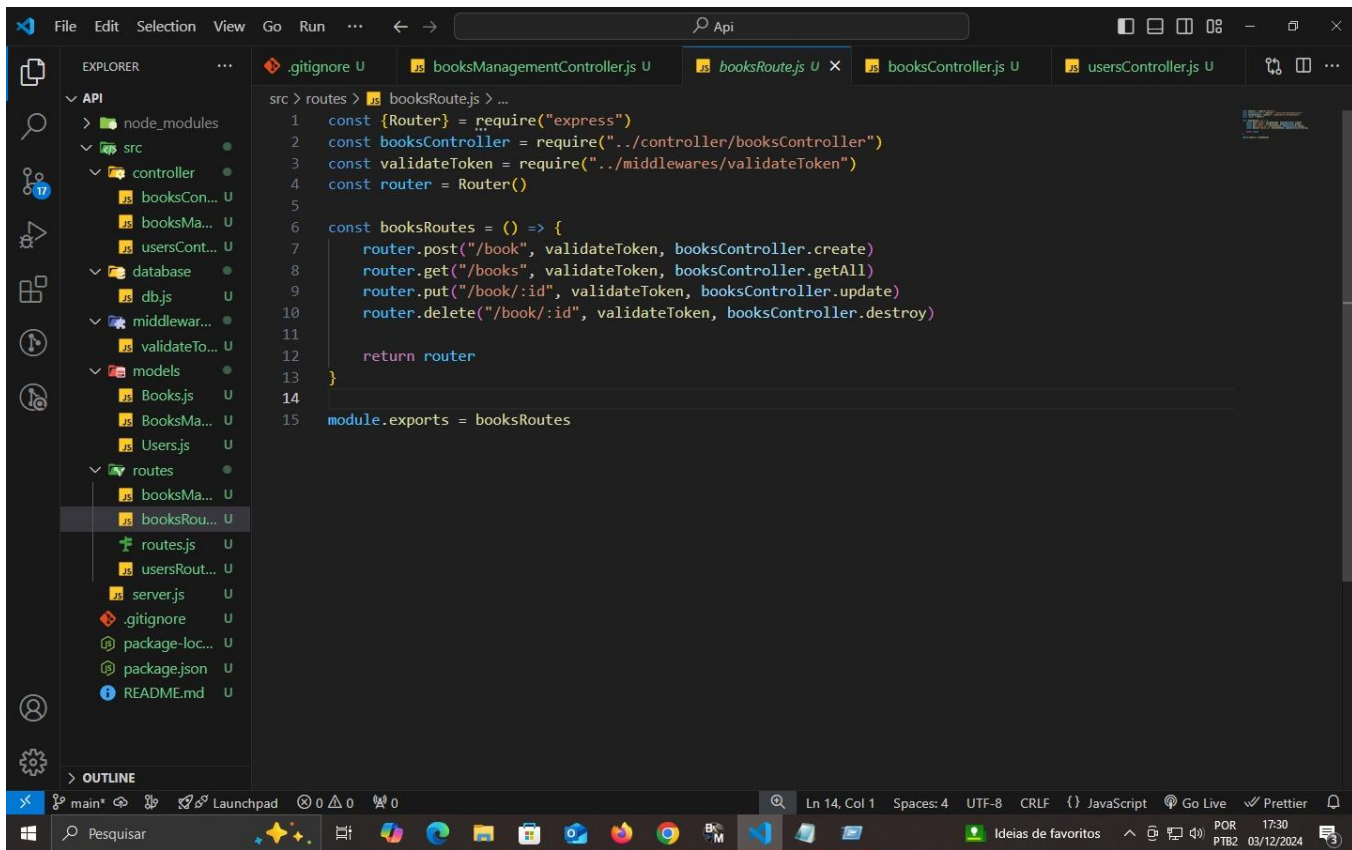
Custos e investimentos

Despesas	Costos diários	Custo final
Internet	R\$ 5,00	R\$ 50,00
Energia Elétrica	R\$ 10,00	R\$ 100,00
Impressão	R\$ 3,00	R\$ 30,00
Valor do Serviço	R\$ 220,67	R\$ 2.206,67
ISS	R\$ 11,03	R\$ 110,33

Trello



Extratos do Código

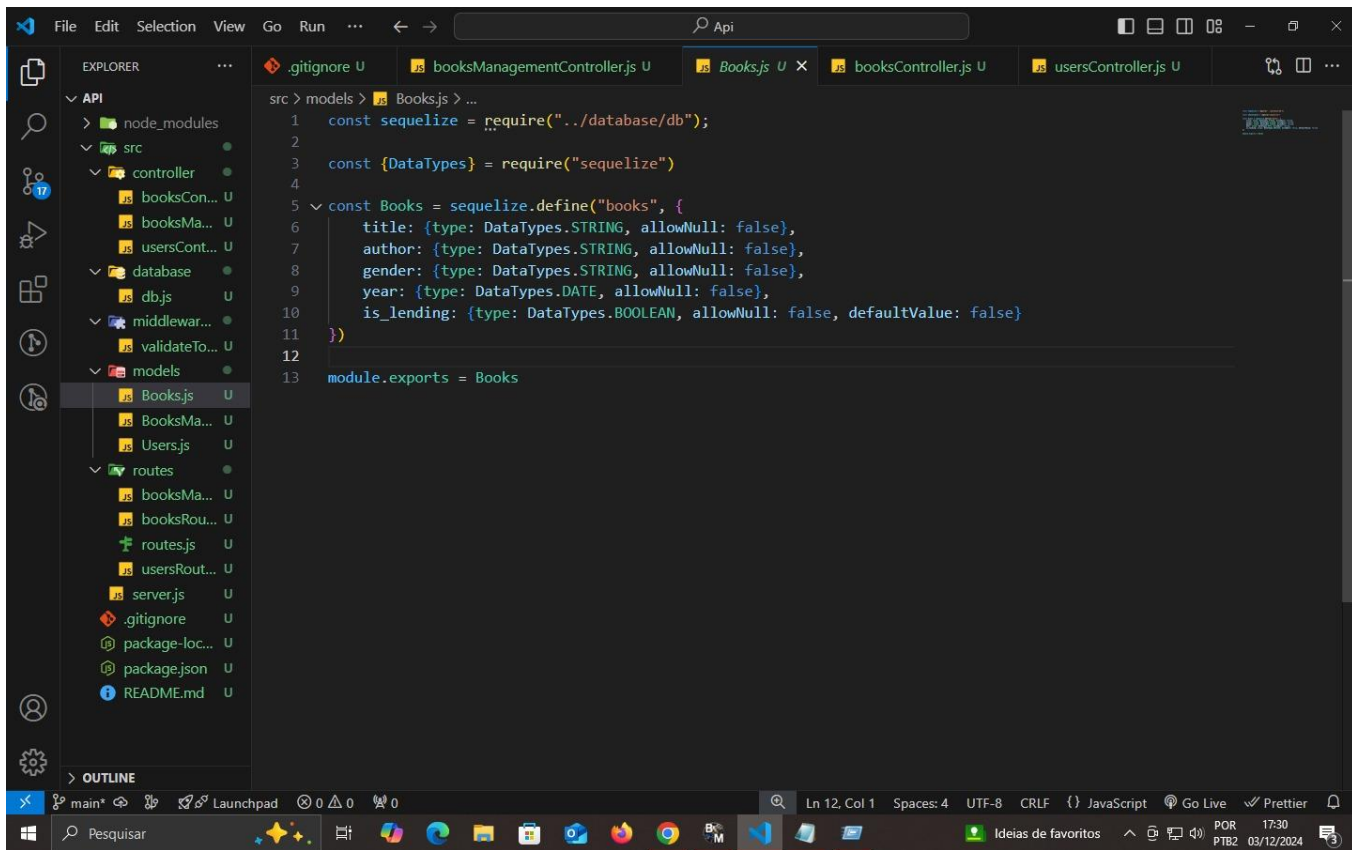


The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor in the center. The file explorer shows the project structure with folders like node_modules, src, controller, database, middleware, models, and routes. The routes folder is expanded, showing booksRoute.js selected. The code editor displays the content of booksRoute.js, which defines the router and its routes.

```
src > routes > booksRoute.js > ...
1  const {Router} = require("express")
2  const booksController = require("../controller/booksController")
3  const validateToken = require("../middlewares/validateToken")
4  const router = Router()
5
6  const booksRoutes = () => {
7      router.post("/book", validateToken, booksController.create)
8      router.get("/books", validateToken, booksController.getAll)
9      router.put("/book/:id", validateToken, booksController.update)
10     router.delete("/book/:id", validateToken, booksController.destroy)
11
12     return router
13 }
14
15 module.exports = booksRoutes
```

The status bar at the bottom indicates the current file is main*.js, the editor is using Launchpad, and the code is in JavaScript with Prettier formatting. The system tray shows the date and time as 03/12/2024 at 17:30.

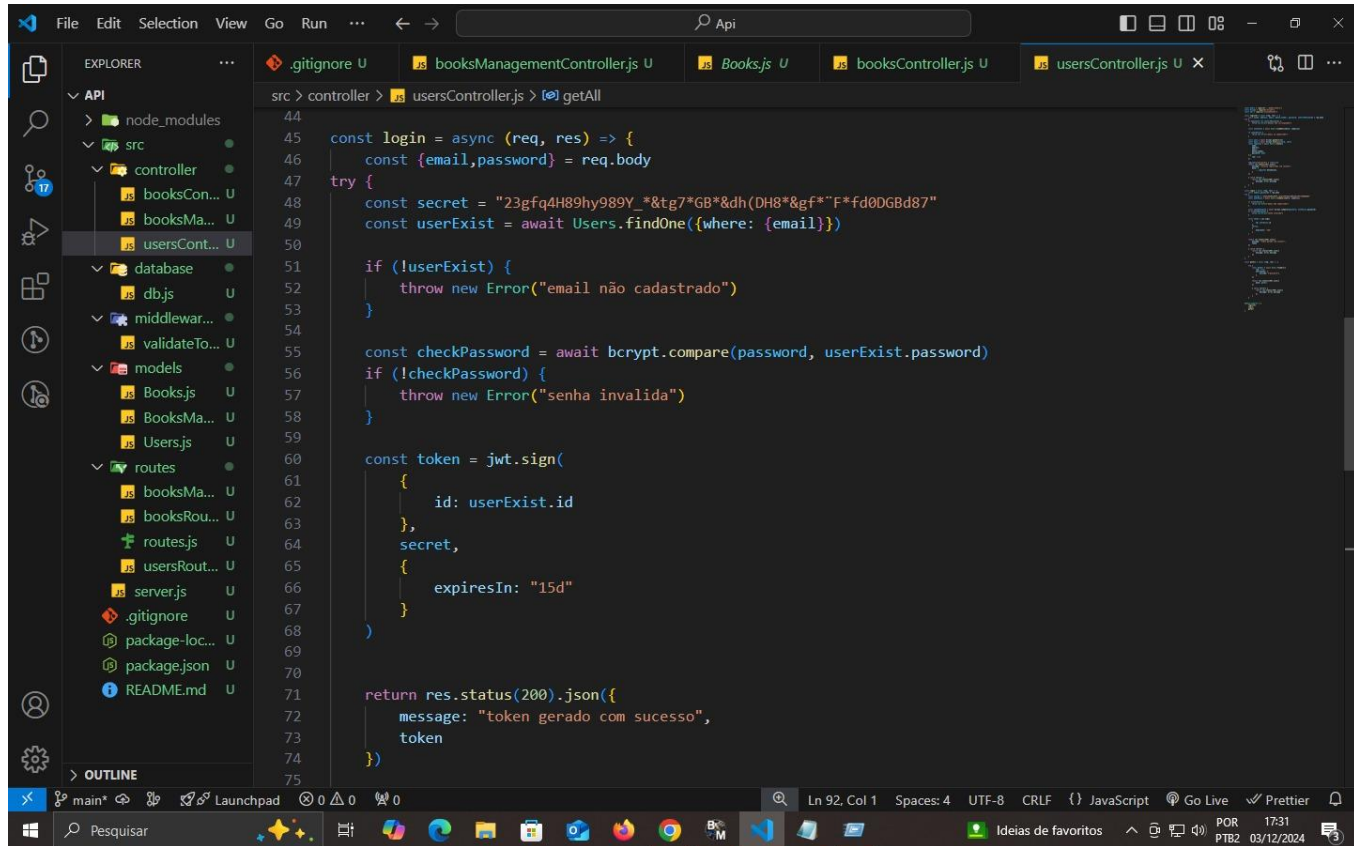
Extratos do Código



The image shows a screenshot of the Visual Studio Code editor. On the left, the Explorer sidebar displays a project structure for an API. The 'models' directory is expanded, showing files like 'Books.js', 'BooksMa...', 'Users.js', and 'routes'. The main editor area shows the content of 'Books.js', which defines a Sequelize model named 'Books'. The code includes imports for 'sequelize' and 'DataTypes', and a definition for the 'Books' model with attributes: title, author, gender, year, and is_lending. The model is then exported as 'Books'.

```
src > models > Books.js > ...
1  const sequelize = require("../database/db");
2
3  const {DataTypes} = require("sequelize")
4
5  const Books = sequelize.define("books", {
6    title: {type: DataTypes.STRING, allowNull: false},
7    author: {type: DataTypes.STRING, allowNull: false},
8    gender: {type: DataTypes.STRING, allowNull: false},
9    year: {type: DataTypes.DATE, allowNull: false},
10   is_lending: {type: DataTypes.BOOLEAN, allowNull: false, defaultValue: false}
11 })
12
13 module.exports = Books
```


Extratos do Código



The image shows a screenshot of a Visual Studio Code editor interface. The Explorer panel on the left displays a project structure with the following files and folders:

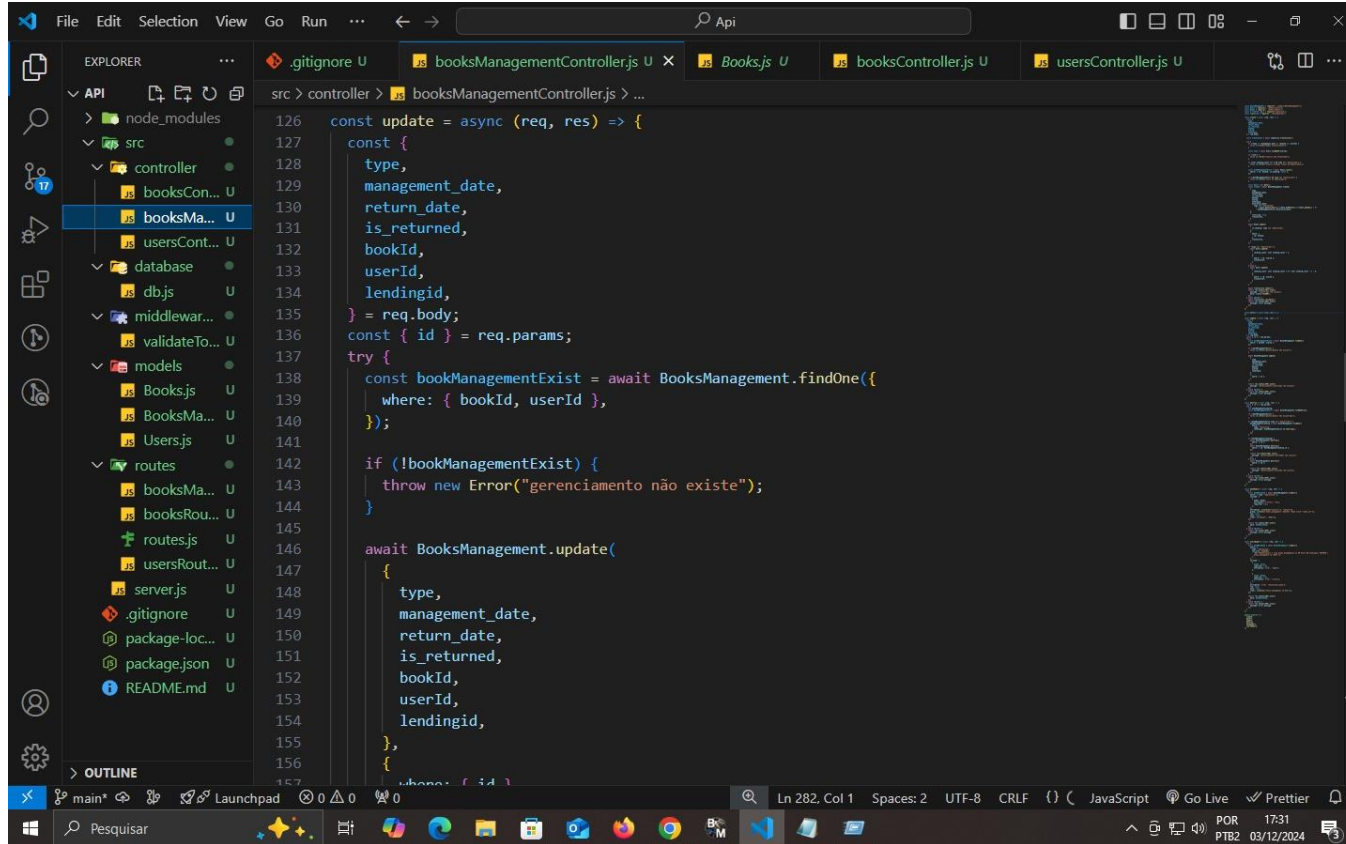
- API
 - node_modules
 - src
 - controller
 - booksCon... U
 - booksMa... U
 - usersCont... U
 - database
 - db.js U
 - middlewar... U
 - validateTo... U
 - models
 - Books.js U
 - BooksMa... U
 - Users.js U
 - routes
 - booksMa... U
 - booksRou... U
 - routes.js U
 - usersRout... U
 - server.js U
 - .gitignore U
 - package-loc... U
 - package.json U
 - README.md U
- OUTLINE

The main editor area shows the code for the `usersController.js` file, specifically the `getAll` method. The code is as follows:

```
src > controller > usersController.js > getAll
44
45 const login = async (req, res) => {
46   const {email,password} = req.body
47   try {
48     const secret = "23gfg4H89hy989Y_&tg7*6B*&dh(DH8*&gf*"F*fd0DGBd87"
49     const userExist = await Users.findOne({where: {email}})
50
51     if (!userExist) {
52       throw new Error("email não cadastrado")
53     }
54
55     const checkPassword = await bcrypt.compare(password, userExist.password)
56     if (!checkPassword) {
57       throw new Error("senha invalida")
58     }
59
60     const token = jwt.sign(
61       {
62         id: userExist.id
63       },
64       secret,
65       {
66         expiresIn: "15d"
67       }
68     )
69
70     return res.status(200).json({
71       message: "token gerado com sucesso",
72       token
73     })
74   }
75 }
```

The status bar at the bottom indicates the current file is `main*`, the editor is in `Launchpad` mode, and the language is `JavaScript`. The system tray shows the date and time as `POR 17:31 03/12/2024`.

Extratos do Código

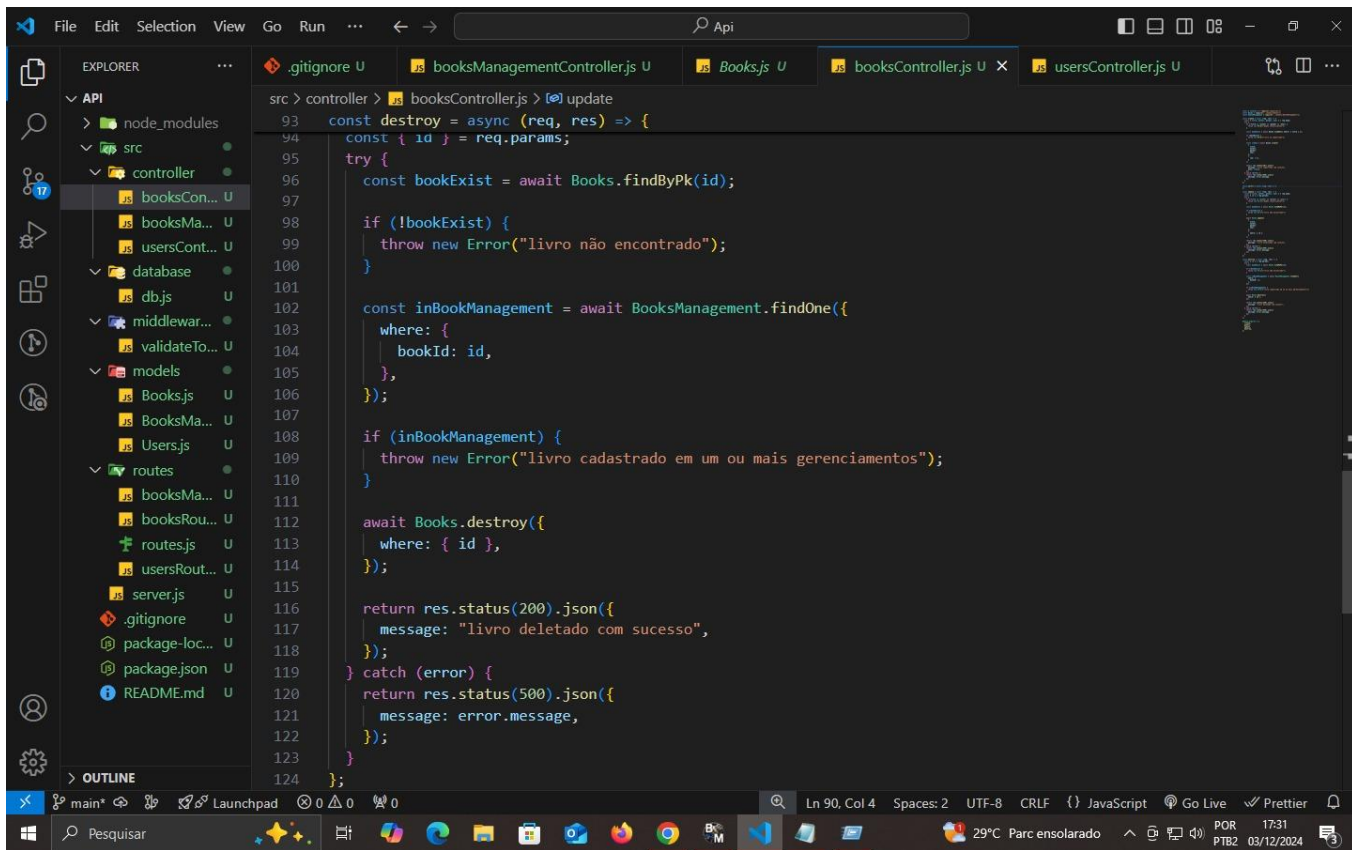


The image shows a screenshot of a Visual Studio Code editor interface. The Explorer panel on the left displays a project structure with folders like 'API', 'src', 'controller', 'database', 'middleware', 'models', and 'routes'. The 'booksManagementController.js' file is selected in the 'controller' folder. The main editor area shows the following JavaScript code snippet:

```
126 const update = async (req, res) => {  
127   const {  
128     type,  
129     management_date,  
130     return_date,  
131     is_returned,  
132     bookId,  
133     userId,  
134     lendingid,  
135   } = req.body;  
136   const { id } = req.params;  
137   try {  
138     const bookManagementExist = await BooksManagement.findOne(  
139       where: { bookId, userId },  
140     );  
141     if (!bookManagementExist) {  
142       throw new Error("gerenciamento não existe");  
143     }  
144     await BooksManagement.update(  
145       {  
146         type,  
147         management_date,  
148         return_date,  
149         is_returned,  
150         bookId,  
151         userId,  
152         lendingid,  
153       },  
154       {  
155         where: { id },  
156       },  
157     );  
158   } catch (error) {  
159     res.status(500).json({ message: error.message });  
160   }  
161   res.status(200).json({ message: "Atualizado com sucesso" });  
162 }
```

The status bar at the bottom indicates the file is 'main.js', the cursor is at line 282, column 1, and the encoding is UTF-8. The system tray shows the date as 03/12/2024.

Extratos do Código



```
File Edit Selection View Go Run ... < -> Api
EXPLORER
  API
    node_modules
    src
      controller
        booksCon... U
        booksMa... U
        usersCont... U
      database
        db.js U
      middlewar...
        validateTo... U
      models
        Books.js U
        BooksMa... U
        Users.js U
      routes
        booksMa... U
        booksRou... U
        routes.js U
        usersRout... U
        server.js U
      .gitignore U
      package-loc... U
      package.json U
      README.md U
  OUTLINE
src > controller > booksController.js > update
93 const destroy = async (req, res) => {
94   const { id } = req.params;
95   try {
96     const bookExist = await Books.findById(id);
97
98     if (!bookExist) {
99       throw new Error("livro não encontrado");
100     }
101
102     const inBookManagement = await BooksManagement.findOne({
103       where: {
104         bookId: id,
105       },
106     });
107
108     if (inBookManagement) {
109       throw new Error("livro cadastrado em um ou mais gerenciamentos");
110     }
111
112     await Books.destroy({
113       where: { id },
114     });
115
116     return res.status(200).json({
117       message: "livro deletado com sucesso",
118     });
119   } catch (error) {
120     return res.status(500).json({
121       message: error.message,
122     });
123   }
124 }
```

Ln 90, Col 4 Spaces: 2 UTF-8 CRLF {} JavaScript Go Live Prettier

Pesquisar 29°C Parc ensolarado 17:31 03/12/2024

Funcionamento do Código

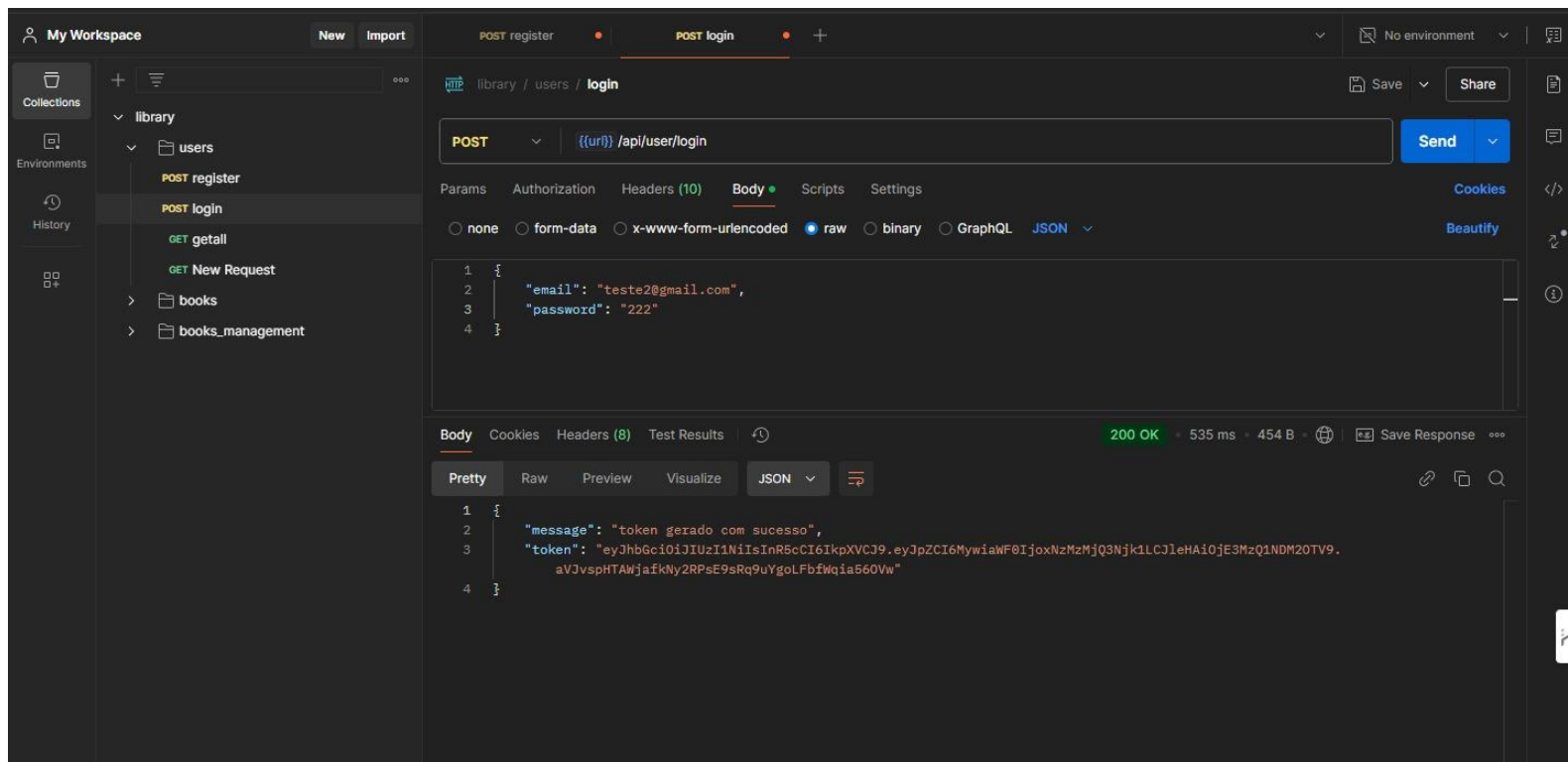
The screenshot displays the Postman interface with a workspace named "My Workspace". On the left sidebar, the "Collections" section shows a folder named "library" containing a sub-folder "users". Inside "users", there is a "POST register" request highlighted. The main panel shows the details of this request, including the URL `{{url}}/api/user/register` and the method "POST". The "Body" tab is selected, showing a JSON payload:

```
1 {
2   "name": "Aluno1",
3   "address": "recife, PE",
4   "email": "teste2@gmail.com",
5   "phone_number": "81 999999908",
6   "password": "222",
7   "confirmPassword": "222"
8 }
```

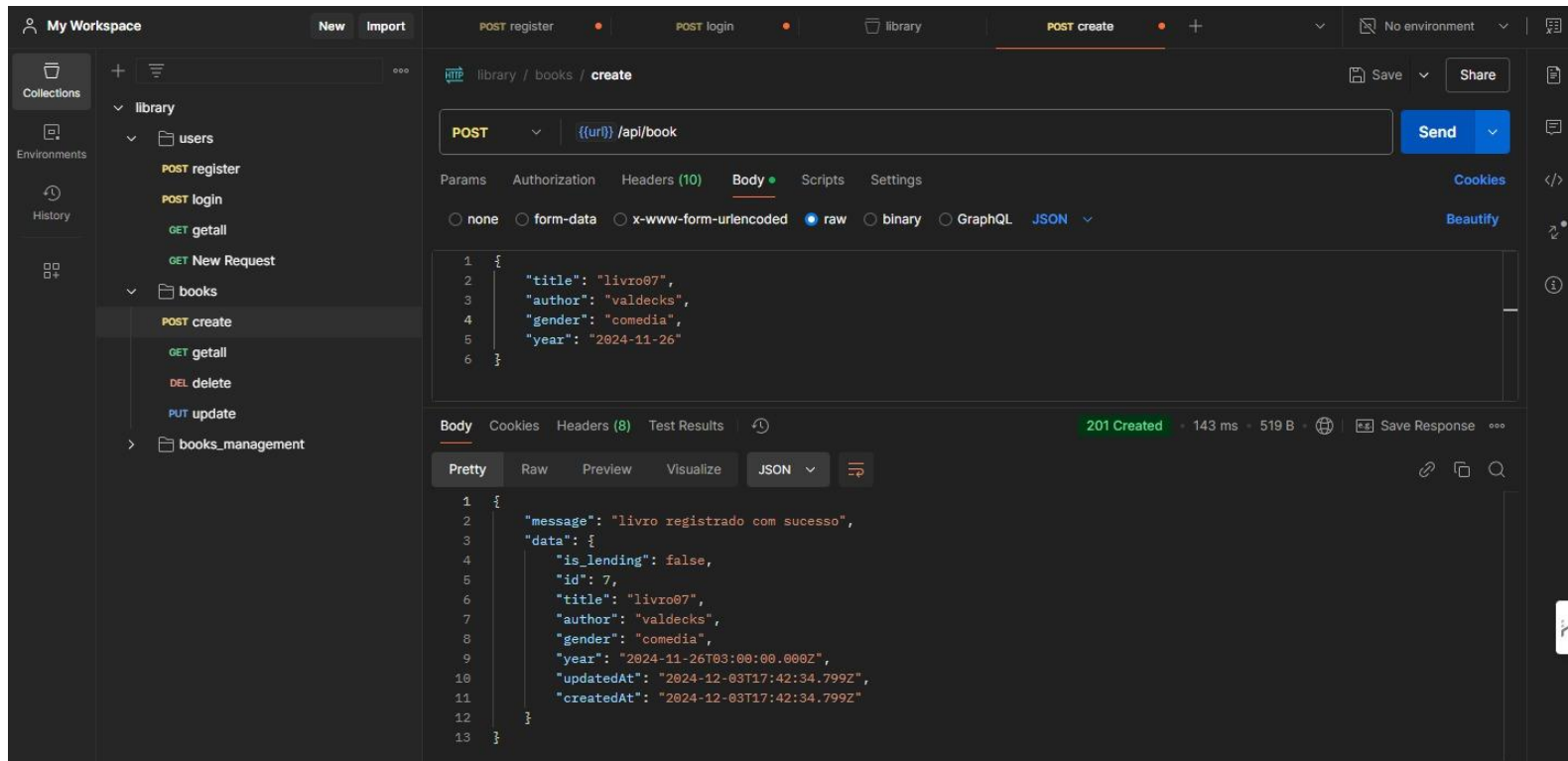
Below the request details, the "Test Results" section shows a successful response with a status of "201 Created", a response time of "503 ms", and a size of "525 B". The response body is displayed in the "Body" tab, showing a JSON object:

```
1 {
2   "message": "usuario registrado com sucesso",
3   "data": {
4     "lending_count": 0,
5     "id": 3,
6     "name": "Aluno1",
7     "address": "recife, PE",
8     "email": "teste2@gmail.com",
9     "phone_number": "81 999999908",
10    "updatedAt": "2024-12-03T17:40:21.775Z",
11    "createdAt": "2024-12-03T17:40:21.775Z"
12  }
13 }
```

Funcionamento do Código



Funcionamento do Código



Funcionamento do Código

The screenshot displays the Postman application interface in dark mode. The left sidebar shows the 'My Workspace' with a collection named 'library' containing sub-collections 'users' and 'books'. The 'books' collection is expanded, showing a 'POST create' request selected. The main panel shows the details of this request, which is a POST to 'library / books_management / create'. The request body is raw JSON, containing a book creation record with fields like 'type', 'management_date', 'return_date', 'is_returned', 'bookId', 'userId', and 'lendingid'. The response is also shown in the bottom panel, indicating a '201 Created' status with a response time of 237 ms and a body size of 614 B. The response body is raw JSON, containing a success message and a detailed object with book and user information, including dates and timestamps.

Request Details:

- Method: POST
- URL: `{{url}} /api/bookmanagement`
- Body Type: raw
- Body Content:

```
1 {
2   "type": "emprestimo",
3   "management_date": "2024-11-28T03:00:00.000Z",
4   "return_date": "2024-12-03",
5   "is_returned": true,
6   "bookId": 7,
7   "userId": 3,
8   "lendingid": 15,
```

Response Details:

- Status: 201 Created
- Time: 237 ms
- Size: 614 B
- Body Type: JSON
- Body Content:

```
1 {
2   "message": "registrado com sucesso",
3   "data": {
4     "id": 19,
5     "type": "emprestimo",
6     "management_date": "2024-11-28T03:00:00.000Z",
7     "return_date": "2024-12-03T00:00:00.000Z",
8     "is_returned": true,
9     "bookId": 7,
10    "userId": 3,
11    "lendingid": "15",
12    "devolution_date": "2024-11-09T03:00:00.000Z",
13    "updatedAt": "2024-12-03T17:44:56.612Z",
14    "createdAt": "2024-12-03T17:44:56.612Z"
```

Funcionamento do Código

The screenshot displays the Postman application interface. On the left sidebar, the 'My Workspace' section is active, showing a collection of requests under the 'books_management' folder. The 'GET bookReport' request is selected. The main panel shows the details of this request, including the URL `{{url}}/api/booksmanagement/bookreport` and the method `GET`. The response is a JSON object with a status of `200 OK`, a response time of `120 ms`, and a size of `622 B`. The response body is displayed in the 'Body' tab, showing a JSON array of book reports.

```
1 {
2   "data": [
3     {
4       "amount": "4",
5       "book": {
6         "title": "livro06",
7         "id": 6
8       }
9     },
10    {
11      "amount": "2",
12      "book": {
13        "title": "livro02",
14        "id": 2
15      }
16    },
17    {
18      "amount": "2",
19      "book": {
20        "title": "livro01",
21        "id": 1
22      }
23    }
24  ]
25 }
```


Fim