



Fecomércio RS

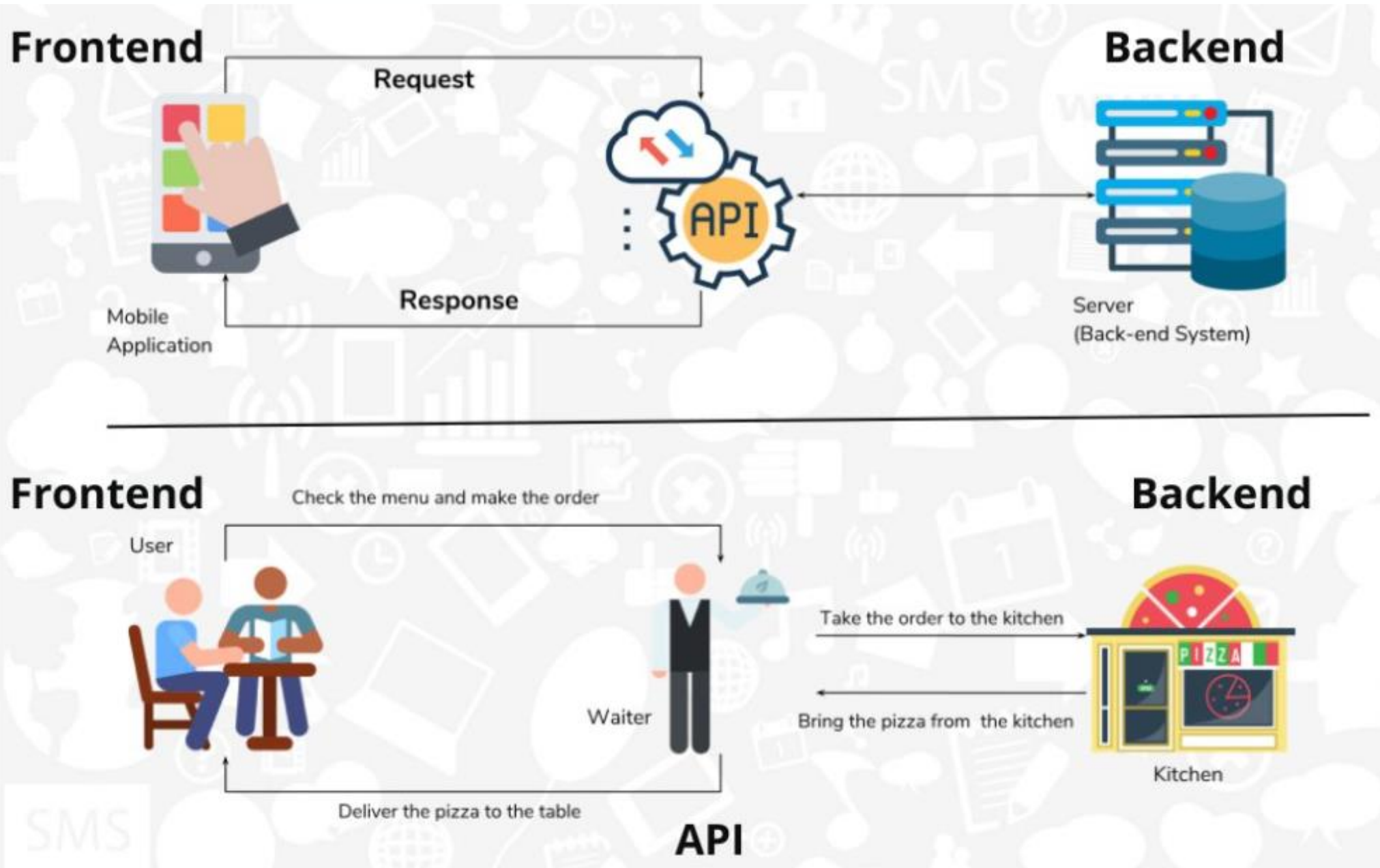


Desenvolvimento de Serviços e APIs

Faculdade Senac Pelotas

Escola de Tecnologia da Informação

Prof. Edécio Fernando Iepsen



Passo a Passo: Criação de uma API

1) Via linha de comandos, criar o projeto e instalar pacotes

```
C:\apis\manha\aula2>npm init -y  
Wrote to C:\apis\manha\aula2\package.json:
```

```
{  
  "name": "aula2",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

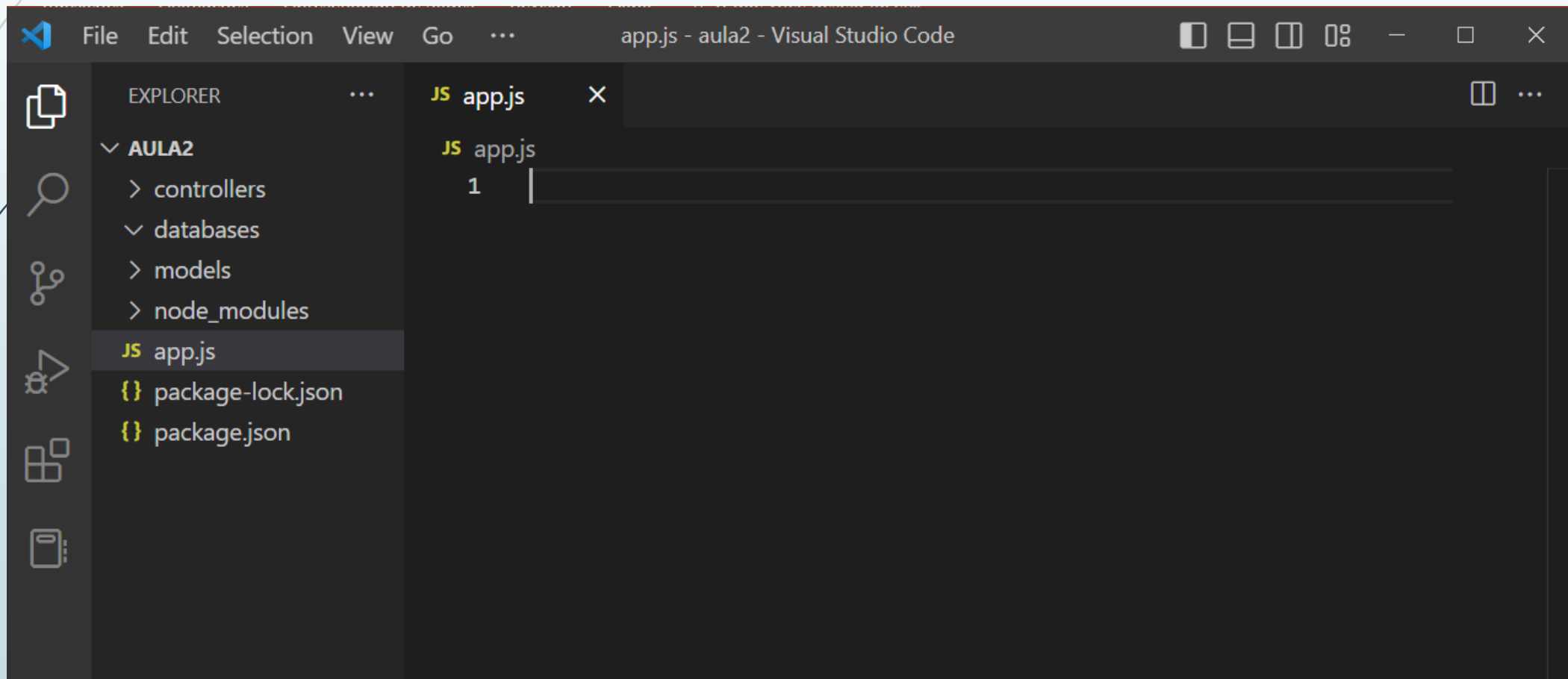
```
C:\apis\manha\aula2>npm i express sequelize sqlite3 cors
```

```
[ ] - idealTree:sequelize: timing idealTree:node_modules/sequelize Completed in 1936ms
```

2) Via VSCode criar as pastas

- databases
- models
- controllers

e o arquivo app.js



3) Código base do arquivo app.js

```
JS app.js > ...
1  import express from 'express'
2  import { sequelize } from './databases/conecta.js'
3  import cors from 'cors'
4  import routes from './routes.js'
5
6  const app = express()
7  const port = 3000
8
9  app.use(express.json())
10 app.use(cors())
11 app.use(routes)
12
13 async function conecta_db() {
14   try {
15     await sequelize.authenticate();
16     console.log('Conexão com banco de dados realizada com sucesso');
17     await sequelize.sync(); // cria as tabelas do sistema (a partir dos modelos - se não existirem)
18   } catch (error) {
19     console.error('Erro na conexão com o banco: ', error);
20   }
21 }
22 conecta_db()
23
24 app.get('/', (req, res) => {
25   res.send('Aula 1: Desenvolvimento de Serviços e APIs')
26 })
27
28 app.listen(port, () => {
29   console.log(`Servidor Rodando na Porta: ${port}`)
30 })
```


4) Ajustar package.json ("type": "module",)

{} package.json X

{} package.json > ...

```
1  {
2    "name": "aula2",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module",
7    "scripts": {
8      "test": "echo \"Error: no test specified\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "cors": "^2.8.5",
15     "express": "^4.18.2",
16     "sequelize": "^6.29.3",
17     "sqlite3": "^5.1.5"
18   }
19 }
20
```

5) Na pasta databases, criar o arquivo conecta.js

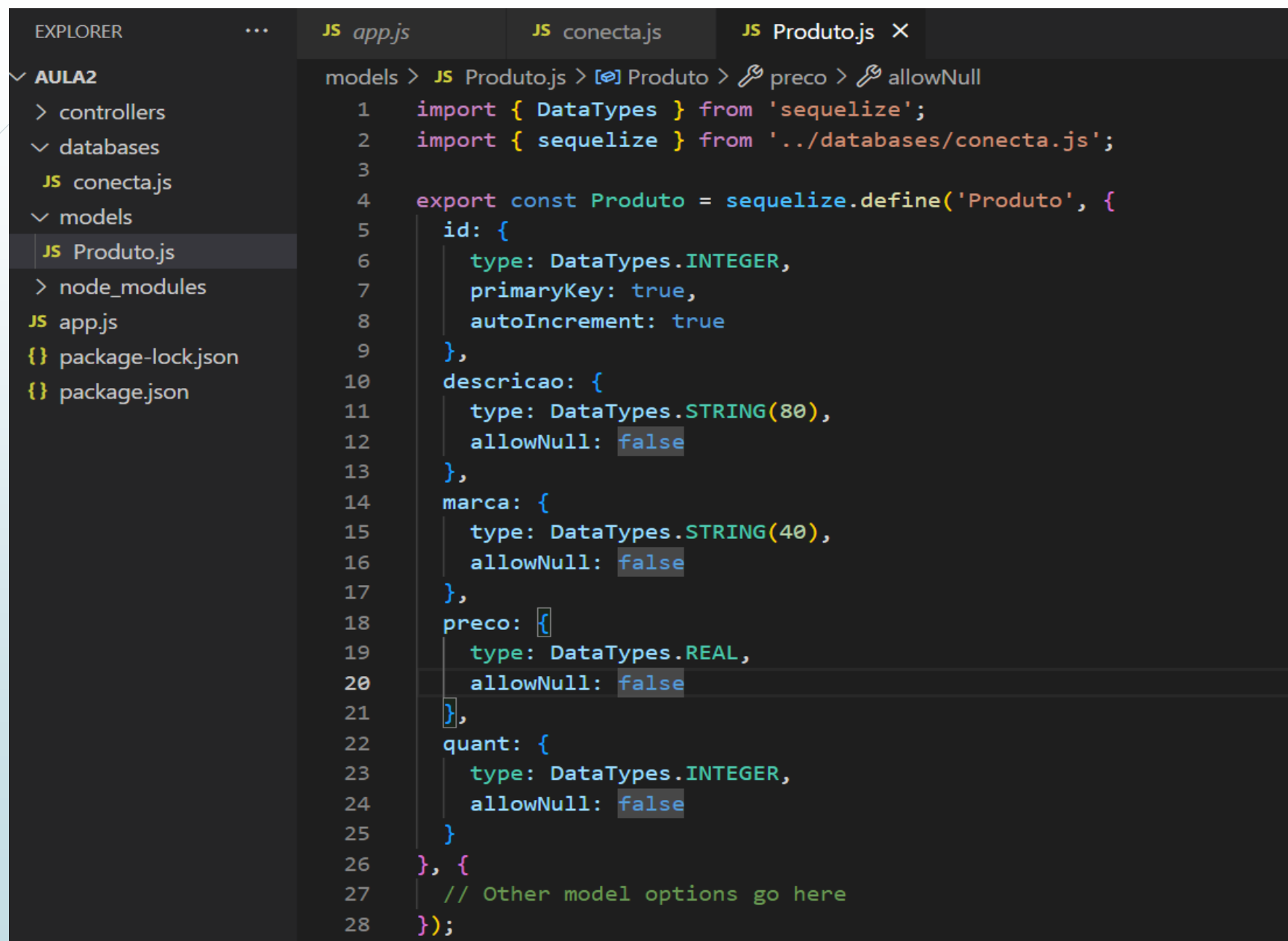


```
EXPLORER  ...  JS app.js  JS conecta.js X

✓ AULA2
  > controllers
  ✓ databases
    JS conecta.js
  > models
  > node_modules
  JS app.js
  {} package-lock.json
  {} package.json

databases > JS conecta.js > ...
1  import { Sequelize } from 'sequelize';
2
3  export const sequelize = new Sequelize({
4    dialect: 'sqlite',
5    storage: './databases/estoque.db3'
6  });
```

6) Na pasta models, criar o(s) modelo(s) - conforme tabela(s) do sistema



```
EXPLORER  ...  JS app.js  JS conecta.js  JS Produto.js X

AULA2
├── controllers
├── databases
│   └── JS conecta.js
├── models
│   └── JS Produto.js
├── node_modules
├── JS app.js
├── {} package-lock.json
└── {} package.json

models > JS Produto.js > [🔍] Produto > 🔑 preco > 🔑 allowNull
1  import { DataTypes } from 'sequelize';
2  import { sequelize } from '../databases/conecta.js';
3
4  export const Produto = sequelize.define('Produto', {
5      id: {
6          type: DataTypes.INTEGER,
7          primaryKey: true,
8          autoIncrement: true
9      },
10     descricao: {
11         type: DataTypes.STRING(80),
12         allowNull: false
13     },
14     marca: {
15         type: DataTypes.STRING(40),
16         allowNull: false
17     },
18     preco: {
19         type: DataTypes.REAL,
20         allowNull: false
21     },
22     quant: {
23         type: DataTypes.INTEGER,
24         allowNull: false
25     }
26 }, {
27     // Other model options go here
28 });
```


7) Na pasta controllers, criar os métodos da API




The image shows a screenshot of the Visual Studio Code editor. On the left, the Explorer sidebar is open, showing a project structure for 'AULA2'. The 'controllers' folder is expanded, and 'produtoController.js' is selected. The main editor area shows the code for 'produtoController.js'. The code imports 'sequelize' from '../databases/conecta.js' and 'Produto' from '../models/Produto.js'. It then exports an asynchronous function 'produtoIndex' that takes 'req' and 'res' as arguments. Inside this function, a 'try' block attempts to find all products using 'Produto.findAll()'. If successful, it returns a 200 status with the product data. If an error occurs, it catches the error and returns a 400 status with the error message.

```
EXPLORER  ...  JS app.js  JS conecta.js  JS Produto.js  JS produtoController.js X

✓ AULA2
  ✓ controllers
    JS produtoController.js
  ✓ databases
    JS conecta.js
  ✓ models
    JS Produto.js
  > node_modules
JS app.js
{} package-lock.json
{} package.json

controllers > JS produtoController.js > [🔍] produtoCreate > [🔍] produto
1  import { sequelize } from '../databases/conecta.js';
2  import { Produto } from '../models/Produto.js'
3
4  export const produtoIndex = async (req, res) => {
5    try {
6      const produtos = await Produto.findAll();
7      res.status(200).json(produtos)
8    } catch (error) {
9      res.status(400).send(error)
10   }
11 }
12
```

```
13 export const produtoCreate = async (req, res) => {
14   const { descricao, marca, quant, preco } = req.body
15
16   // se não informou estes atributos
17   if (!descricao || !marca || !quant || !preco) {
18     res.status(400).json({ id: 0, msg: "Erro... Informe nome, marca, quant e preco do produto." })
19     return
20   }
21
22   try {
23     const produto = await Produto.create({
24       descricao, marca, quant, preco
25     });
26     res.status(201).json(produto)
27   } catch (error) {
28     res.status(400).send(error)
29   }
30 }
31
```



```
32 export const produtoUpdate = async (req, res) => {
33   const { id } = req.params
34   const { descricao, marca, quant, preco } = req.body
35
36   if (!descricao || !marca || !quant || !preco) {
37     res.status(400).json({ id: 0, msg: "Erro... Informe nome, marca, quant e preco do produto." })
38     return
39   }
40
41   try {
42     const produto = await Produto.update({
43       descricao, marca, quant, preco
44     }, {
45       where: { id }
46     });
47     res.status(200).json(produto)
48   } catch (error) {
49     res.status(400).send(error)
50   }
51 }
```

```
52
53 export const produtoDestroy = async (req, res) => {
54   const { id } = req.params
55   try {
56     const produto = await Produto.destroy({
57       where: { id }
58     });
59     res.status(200).json(produto)
60   } catch (error) {
61     res.status(400).send(error)
62   }
63 }
64
65 export const produtoSearch = async (req, res) => {
66   const { id } = req.params
67   try {
68     const produto = await Produto.findByPk(id);
69     if (produto) {
70       res.status(200).json(produto)
71     } else {
72       res.status(200).json({ id: 0, msg: "Erro... Produto não encontrado." })
73     }
74   } catch (error) {
75     res.status(400).send(error)
76   }
77 }
```

```
78
79 export const produtoTotal = async (req, res) => {
80   try {
81     const num = await Produto.count();
82     const total = await Produto.sum('preco')
83
84     // para calcular o total em estoque (soma da quantidade * preco de cada produto)
85     const total2 = await Produto.findOne({
86       attributes: [[sequelize.fn("SUM", sequelize.literal("quant*preco")), 'subtotal']],
87       raw: true
88     });
89
90     if (num && total && total2) {
91       res.status(200).json({ num, total, total2: total2.subtotal })
92     } else {
93       res.status(200).json({ id: 0, msg: "Erro... Falha no cálculo dos totais ou estoque vazio" })
94     }
95   } catch (error) {
96     res.status(400).send(error)
97   }
98 }
```

8) Criar o arquivo routes.js

```
JS app.js    JS conecta.js    JS Produto.js    JS produtoController.js    JS routes.js X

JS routes.js > [🔍] default
1  import { Router } from "express"
2  import { produtoCreate, produtoDestroy, produtoIndex,
3      produtoSearch, produtoTotal, produtoUpdate } from "../controllers/produtoController.js"
4
5  const router = Router()
6
7  router.get('/produtos', produtoIndex)
8      .post('/produtos', produtoCreate)
9      .put('/produtos/:id', produtoUpdate)
10     .delete('/produtos/:id', produtoDestroy)
11     .get('/produtos/total', produtoTotal)
12     .get('/produtos/:id', produtoSearch)
13
14  export default router
```

9) (instalar nodemon) e rodar o projeto

```
C:\apis\manha\aula2>npm i --save-dev nodemon
```

```
added 27 packages, and audited 213 packages in 4s
```

```
15 packages are looking for funding  
  run `npm fund` for details
```

```
found 0 vulnerabilities
```

```
C:\apis\manha\aula2>npx nodemon app
```

```
[nodemon] 2.0.21
```

```
[nodemon] to restart at any time, enter `rs`
```

```
[nodemon] watching path(s): *.*
```

```
[nodemon] watching extensions: js,mjs,json
```

```
[nodemon] starting `node app.js`
```

```
Servidor Rodando na Porta: 3000
```

```
Executing (default): SELECT 1+1 AS result
```

```
Conexão com banco de dados realizada com sucesso
```

```
Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='Produtos';
```

```
Executing (default): CREATE TABLE IF NOT EXISTS `Produtos` (`id` INTEGER PRIMARY KEY AUTOINCREMENT,  
  `descricao` VARCHAR(80) NOT NULL, `marca` VARCHAR(40) NOT NULL, `preco` REAL NOT NULL, `quant` INT  
  EGER NOT NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL);
```

```
Executing (default): PRAGMA INDEX_LIST(`Produtos`)
```

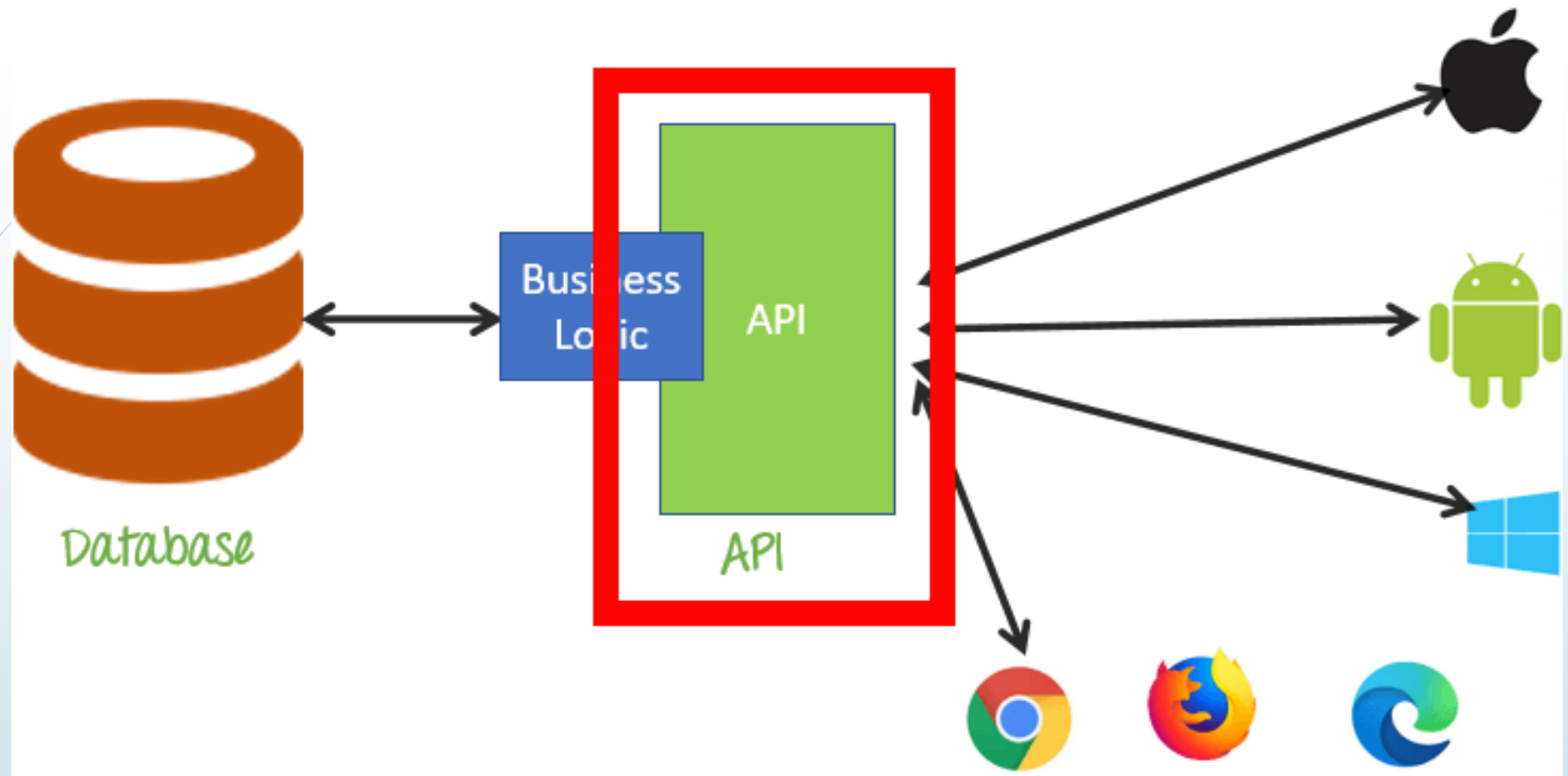
10) Testar as rotas via Insomnia

The screenshot displays the Insomnia application window. The top bar includes the Insomnia logo, a 'Star' button with '27.061' stars, the text 'Insomnia / Estoque23', and 'Login' and 'Sign Up' buttons. The main interface is divided into several sections:

- Left Sidebar:** Contains a home icon, a search filter, and a list of endpoints under the 'Estoque' folder. The endpoints are: 'Delete' (DEL), 'Total' (GET), 'Alteração' (PUT), 'Inclusão' (POST, highlighted), and 'Listagem' (GET).
- Request Section:** Shows a POST request to 'http://localhost:3000/produtos'. The 'Send' button is visible. Below the URL bar are tabs for 'JSON', 'Auth', 'Query', 'Headers' (with a count of 1), and 'Docs'.
- Response Section:** Displays the response status '201 Created', response time '544 ms', and response size '150 B'. It also shows the response body in JSON format, which includes an 'id' and timestamps for 'updatedAt' and 'createdAt'.
- Bottom Bar:** Includes a 'Beautify JSON' button and a JSON path '\$.store.books[*].author'.

```
1 {
2   "descricao": "Nescafé",
3   "marca": "Nestlé",
4   "quant": 12,
5   "preco": 8.90
6 }
```

```
1 {
2   "id": 1,
3   "descricao": "Nescafé",
4   "marca": "Nestlé",
5   "quant": 12,
6   "preco": 8.9,
7   "updatedAt": "2023-03-14T03:16:09.583Z",
8   "createdAt": "2023-03-14T03:16:09.583Z"
9 }
```

Exemplo: Consumir a API a partir da Web usando React

Exercícios

Acesse a guia da RealPython (<https://realpython.com/api-integration-in-python/>) sobre Python e REST APIs e crie uma aplicação em Python para consumir a API criada. Exibir um menu com as opções:

1. Incluir Produto
2. Listar Produtos
3. Alterar Produto
4. Excluir Produto
5. Pesquisar por Código
6. Exibir totais
7. Finalizar