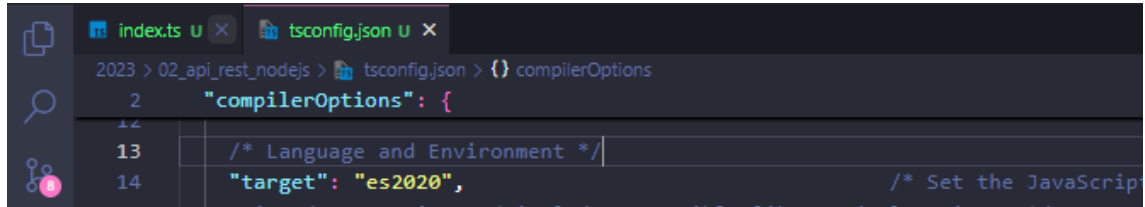


API REST NODE JS

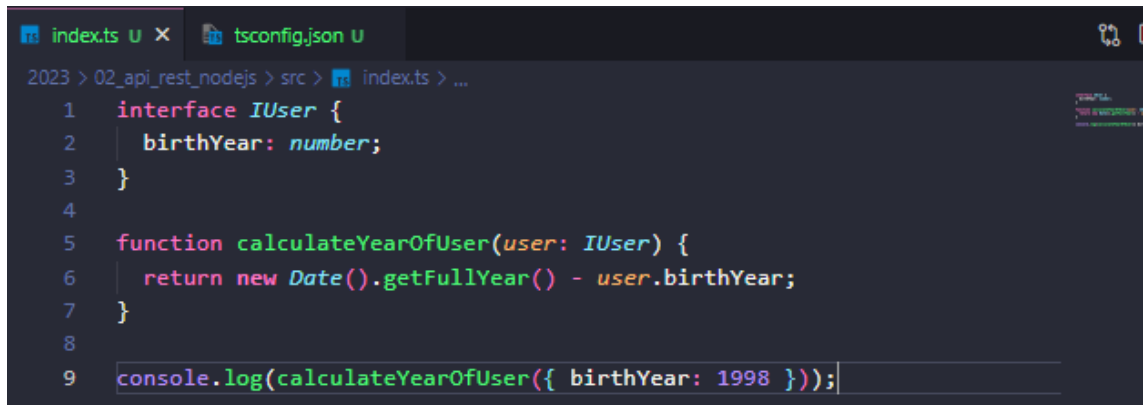
01 – Entendendo o typescript

Instalar `npm i -D typescript` e depois executar o comando `npx tsc -init` para criar o arquivo `tsconfig.json` e colocar o target para 2020



```
2023 > 02_api_rest_nodejs > tsconfig.json > {} compilerOptions
2  "compilerOptions": {
13  /* Language and Environment */
14  "target": "es2020", /* Set the JavaScript
```

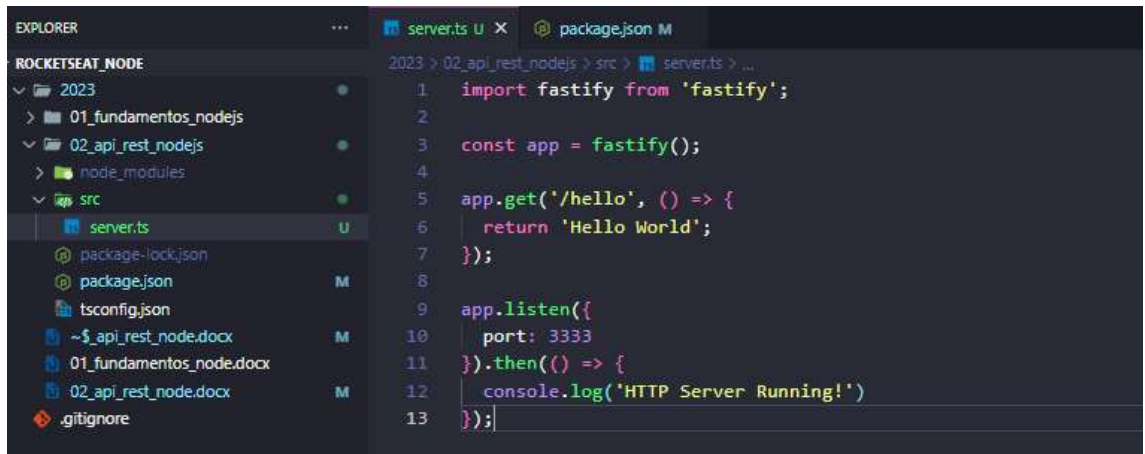
E para converter o código em js `npx tsc src/index.ts`



```
1 interface IUser {
2   birthYear: number;
3 }
4
5 function calculateYearOfUser(user: IUser) {
6   return new Date().getFullYear() - user.birthYear;
7 }
8
9 console.log(calculateYearOfUser({ birthYear: 1998 }));
```

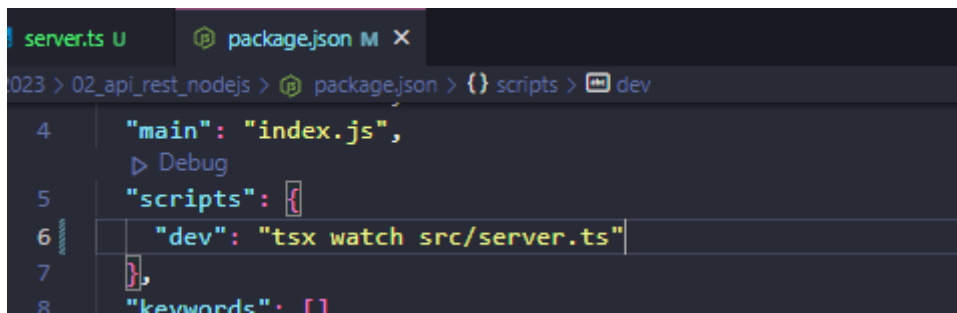
02 – Criando a aplicação

Instalar `npm i fastify` e também instalar `npm install -D @types/node` e `npm install tsx -D`



```
EXPLORER
ROCKETSEAT_NODE
2023
  > 01_fundamentos_nodejs
  > 02_api_rest_nodejs
    > node_modules
    > src
      server.ts
      package-lock.json
      package.json
      tsconfig.json
      ~$ _api_rest_node.docx
      01_fundamentos_node.docx
      02_api_rest_node.docx
      .gitignore

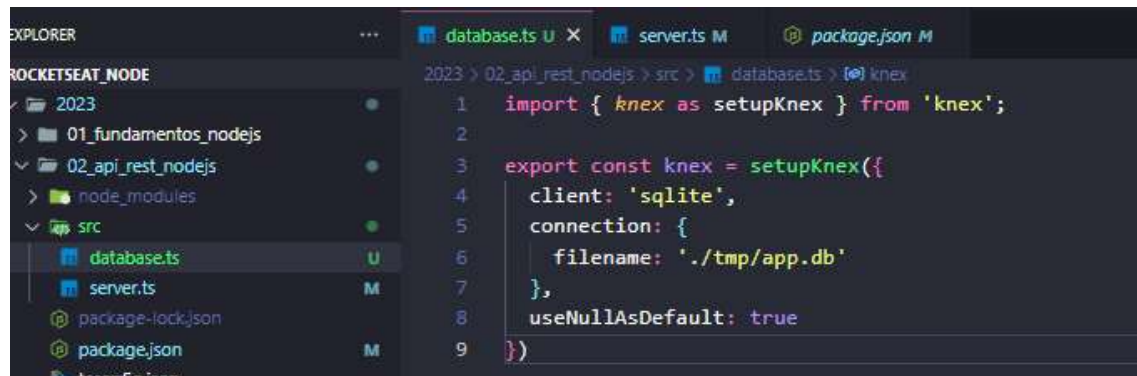
2023 > 02_api_rest_nodejs > src > server.ts > ...
1 import fastify from 'fastify';
2
3 const app = fastify();
4
5 app.get('/hello', () => {
6   return 'Hello World';
7 });
8
9 app.listen({
10   port: 3333
11 }).then(() => {
12   console.log('HTTP Server Running!')
13 });
```



```
server.ts U package.json M X
2023 > 02_api_rest_nodejs > package.json > {} scripts > dev
4 "main": "index.js",
  > Debug
5 "scripts": {
6   "dev": "tsx watch src/server.ts"
7 },
8 "keywords": []
```

03 – Configurando o knex

Instalando o knex `npm install knex sqlite3`

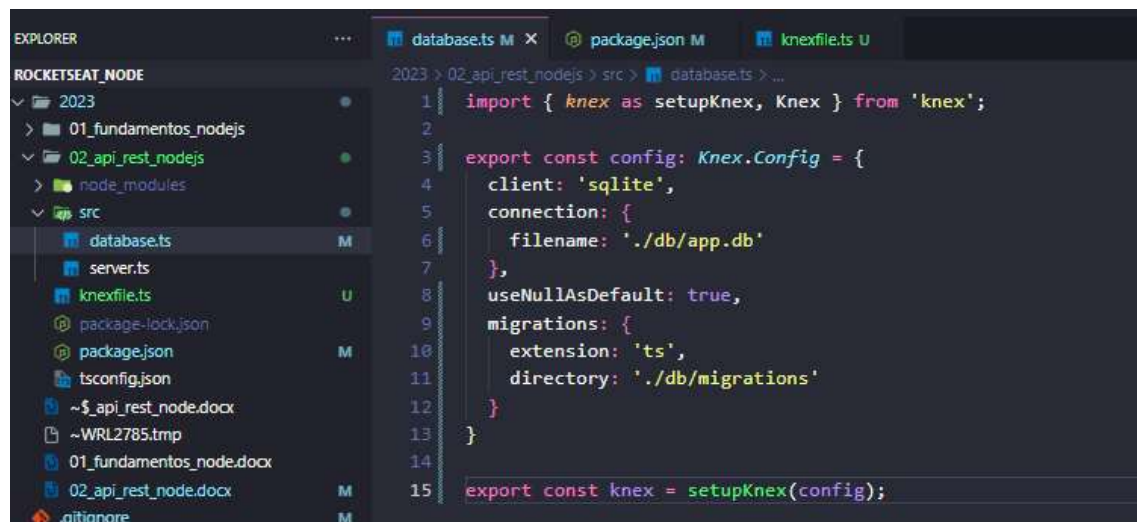


The screenshot shows the VS Code interface with the Explorer on the left and the Source Control view on the right. The Explorer shows the project structure with folders for 2023, 01_fundamentos_nodejs, 02_api_rest_nodejs, node_modules, and src. The src folder contains database.ts, server.ts, package-lock.json, package.json, and tsconfig.json. The Source Control view shows the changes in database.ts. The code in database.ts is as follows:

```
1 import { knex as setupKnex } from 'knex';
2
3 export const knex = setupKnex({
4   client: 'sqlite',
5   connection: {
6     filename: './tmp/app.db'
7   },
8   useNullAsDefault: true
9 });
```

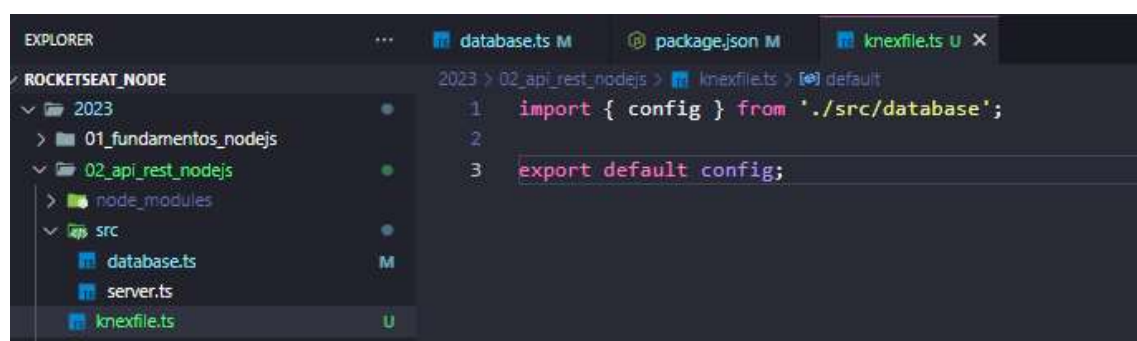
04 – Criando primeira migration

Instalar `npm install -g knex`



The screenshot shows the VS Code interface with the Explorer on the left and the Source Control view on the right. The Explorer shows the project structure with folders for 2023, 01_fundamentos_nodejs, 02_api_rest_nodejs, node_modules, and src. The src folder contains database.ts, server.ts, knexfile.ts, package-lock.json, package.json, tsconfig.json, and .gitignore. The Source Control view shows the changes in database.ts and knexfile.ts. The code in database.ts is as follows:

```
1 import { knex as setupKnex, Knex } from 'knex';
2
3 export const config: Knex.Config = {
4   client: 'sqlite',
5   connection: {
6     filename: './db/app.db'
7   },
8   useNullAsDefault: true,
9   migrations: {
10     extension: 'ts',
11     directory: './db/migrations'
12   }
13 }
14
15 export const knex = setupKnex(config);
```



The screenshot shows the VS Code interface with the Explorer on the left and the Source Control view on the right. The Explorer shows the project structure with folders for 2023, 01_fundamentos_nodejs, 02_api_rest_nodejs, node_modules, and src. The src folder contains database.ts, server.ts, knexfile.ts, package-lock.json, package.json, tsconfig.json, and .gitignore. The Source Control view shows the changes in database.ts and knexfile.ts. The code in knexfile.ts is as follows:

```
1 import { config } from './src/database';
2
3 export default config;
```

```
database.ts M package.json M X knexfile.ts U
023 > 02_api_rest_nodejs > package.json > {} devDependencies
4   "main": "index.js",
5   "scripts": {
6     "dev": "tsx watch src/server.ts",
7     "knex": "tsx ./node_modules/knex/bin/cli.js"
8   },
```

Comando para iniciar a migration `npm run knex migrate:make create-documents`

05 - Criando tabela de transações

```
20250106155409_create-transactions.ts • server.ts
db > migrations > 20250106155409_create-transactions.ts > ...
1  import type { Knex } from "knex";
2
3  export async function up(knex: Knex): Promise<void> {
4    await knex.schema.createTable('transactions', (table) => {
5      table.uuid('id').primary()
6      table.text('title').notNullable()
7      table.decimal('amount', 10,2).notNullable()
8      table.timestamp('created_at').defaultTo(knex.fn.now()).notNullable()
9    })
10 }
11
12 export async function down(knex: Knex): Promise<void> {
13   await knex.schema.dropTable('transactions')
14 }
```

Depois rodar o comando `npm run knex -- migrate:latest` desfazer a migration `npm run knex -- migrate:rollback`

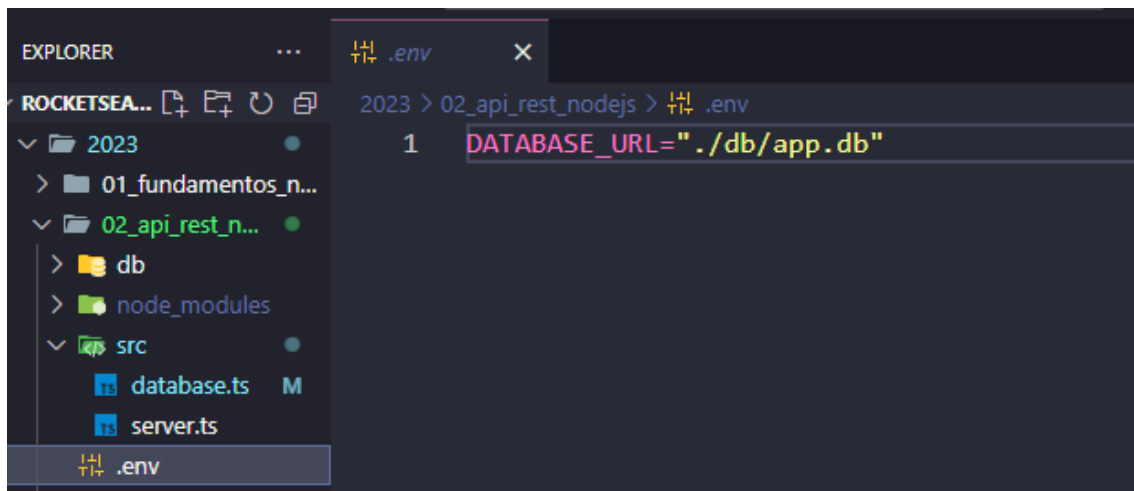
```
EXPLORER 20250106155409_create-transactions.ts 20250106185936_add-session-id-to-transactions.ts X
db > migrations > 20250106185936_add-session-id-to-transactions.ts > ...
1  import type { Knex } from "knex";
2
3  export async function up(knex: Knex): Promise<void> {
4    await knex.schema.alterTable('transactions', (table) => {
5      table.uuid('session_id').after('id').index()
6    })
7  }
8
9  export async function down(knex: Knex): Promise<void> {
10   await knex.schema.alterTable('transactions', (table) => {
11     table.dropColumn('session_id')
12   })
13 }
14
```

06 – Realizando queries com o knex

```
server.ts x
src > server.ts > app.get('/hello') callback
4
5  const app = fastify();
6
7  app.get('/hello', async () => {
8    // const transation = await knex('transactions').insert({
9    //   id: crypto.randomUUID(),
10   //   title: 'Transação de teste',
11   //   amount: 1000
12   // }).returning('*');
13
14   //const transation = await knex('transactions').select('*');
15   const transation = await knex('transactions').where('amount', 1000).select('*');
16
17   return transation;
18 });
19
20 app.listen({
21   port: 3333
22 }).then(() => {
23   console.log('HTTP Server Running!')
24 });
25
```

07 – Variáveis de ambiente

Instalar npm i dotenv



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a file tree for a project named 'ROCKETSEA...'. The tree includes a folder '2023' which contains subfolders '01_fundamentos_n...' and '02_api_rest_n...'. The '02_api_rest_n...' folder is expanded, showing subfolders 'db' and 'node_modules', and a 'src' folder containing 'database.ts' and 'server.ts'. The '.env' file is selected and highlighted. On the right, the editor window shows the content of the '.env' file, which contains a single line: 'DATABASE_URL="./db/app.db'.

```
EXPLORER
ROCKETSEA...
  2023
    01_fundamentos_n...
    02_api_rest_n...
      db
      node_modules
      src
        database.ts
        server.ts
      .env

2023 > 02_api_rest_nodejs > .env
1 DATABASE_URL="./db/app.db"
```

```
database.ts M X
2023 > 02_api_rest_nodejs > src > database.ts > config > migrations
1 import 'dotenv/config';
2 import { knex as setupKnex, Knex } from 'knex';
3
4 if (!process.env.DATABASE_URL) {
5   throw new Error('DATABASE_URL env not found.')
6 }
7
8 export const config: Knex.Config = {
9   client: 'sqlite',
10  connection: {
11    filename: process.env.DATABASE_URL
12  },
13  useNullAsDefault: true
```

08 - Tratando env com zod

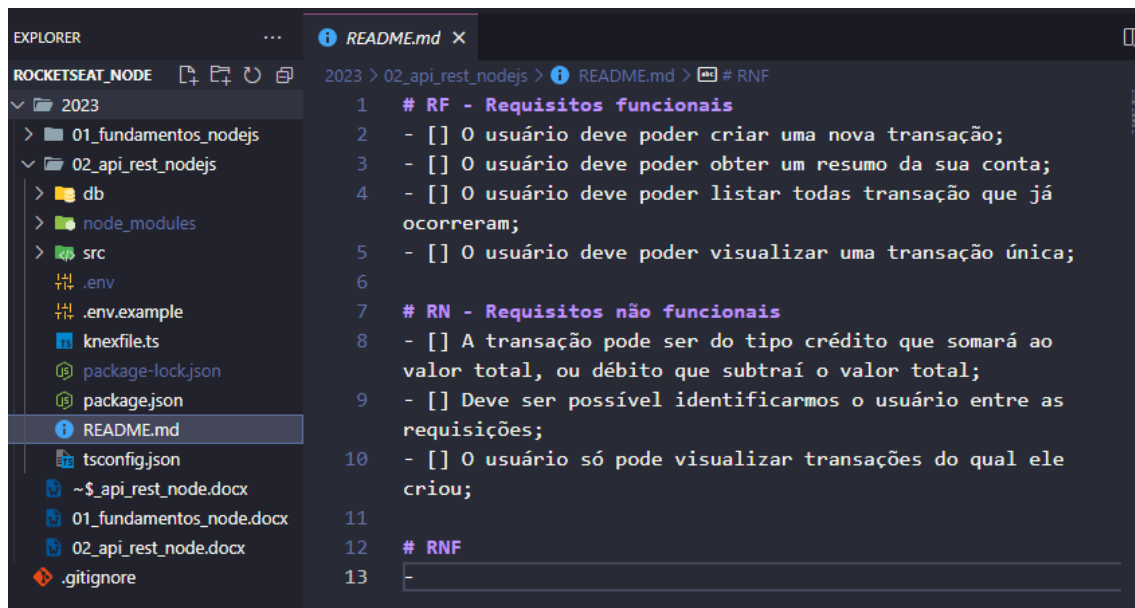
Instalar npm i zod

```
index.ts U X .env database.ts M X
2023 > 02_api_rest_nodejs > src > database.ts > config > connection > filename
1 import 'dotenv/config';
2 import { z } from 'zod';
3
4 const envSchema = z.object({
5   NODE_ENV: z.enum(['development', 'test', 'production']).default('production'),
6   DATABASE_URL: z.string(),
7   PORT: z.number().default(3333)
8 });
9
10 const _env = envSchema.safeParse(process.env);
11
12 if (_env.success === false) {
13   console.error('Invalid environment variables!', _env.error.format());
14   throw new Error('Invalid environment variables!');
15 }
16
17 export const env = _env.data;
```

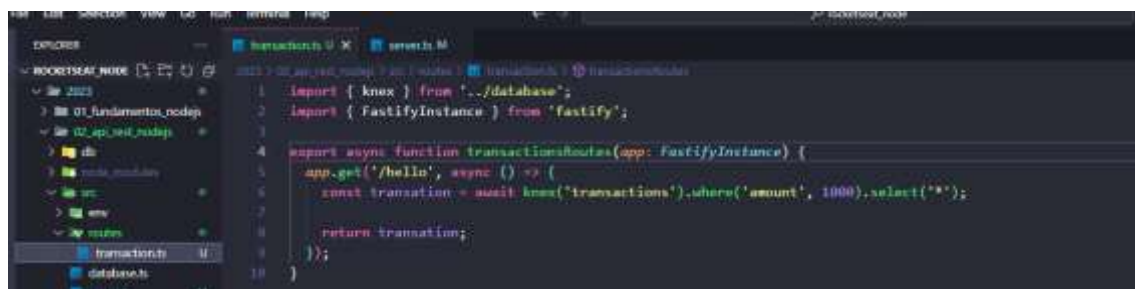
```
index.ts U .env database.ts M X
2023 > 02_api_rest_nodejs > src > database.ts > config > connection > filename
1 import { knex as setupKnex, Knex } from 'knex';
2 import { env } from './env';
3
4 export const config: Knex.Config = {
5   client: 'sqlite',
6   connection: {
7     filename: env.DATABASE_URL
8   },
9 }
```

```
index.ts U .env server.ts M database.ts M
2023 > 02_api_rest_nodejs > .env
1 NODE_ENV=development
2 DATABASE_URL="./db/app.db"
```

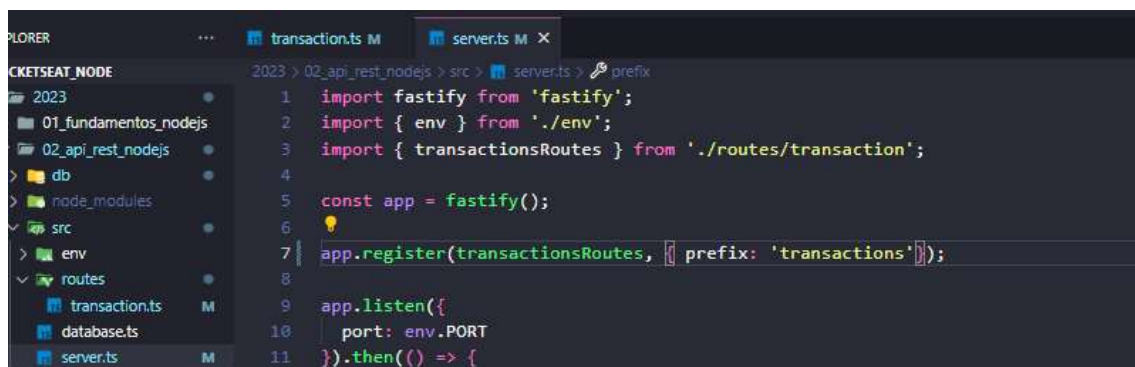
09 – Requisitos da aplicação

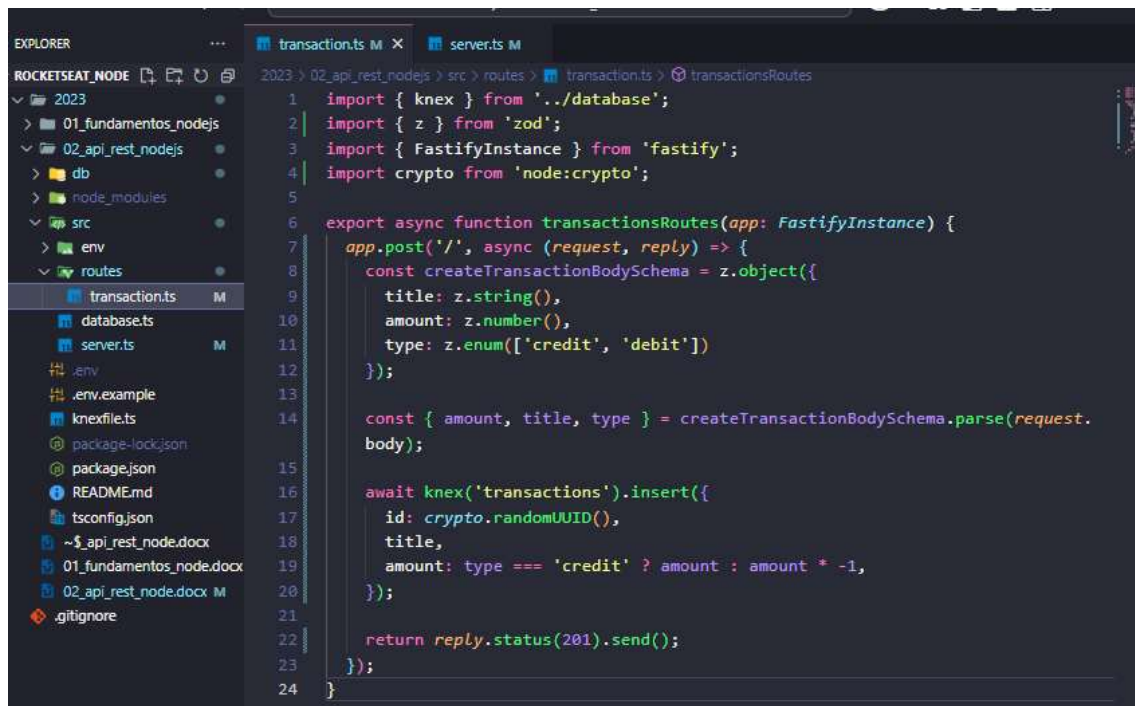


10 – Plugin do fastify



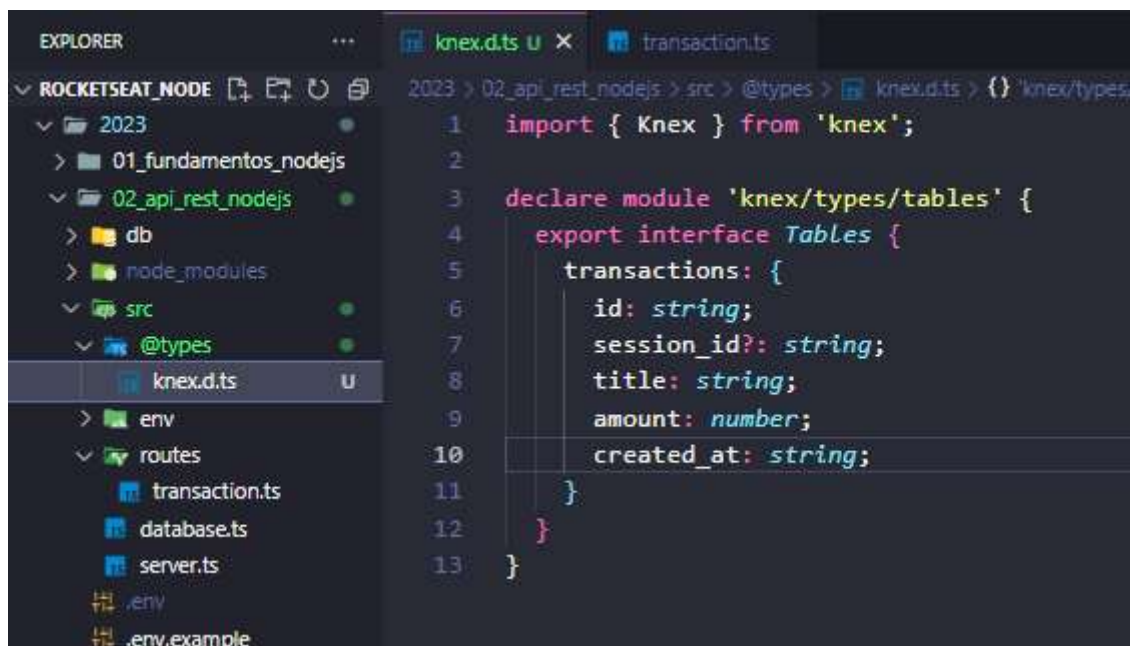
11 – Criação de transação





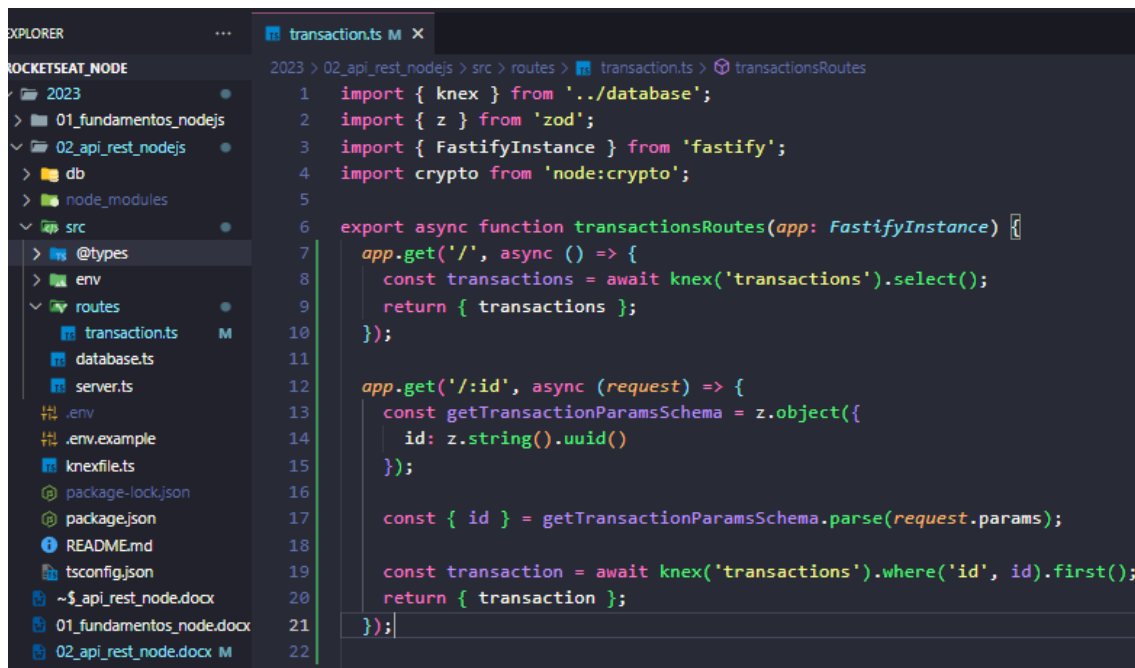
```
1 import { knex } from '../database';
2 import { z } from 'zod';
3 import { FastifyInstance } from 'fastify';
4 import crypto from 'node:crypto';
5
6 export async function transactionsRoutes(app: FastifyInstance) {
7   app.post('/', async (request, reply) => {
8     const createTransactionBodySchema = z.object({
9       title: z.string(),
10      amount: z.number(),
11      type: z.enum(['credit', 'debit'])
12    });
13
14    const { amount, title, type } = createTransactionBodySchema.parse(request.body);
15
16    await knex('transactions').insert({
17      id: crypto.randomUUID(),
18      title,
19      amount: type === 'credit' ? amount : amount * -1,
20    });
21
22    return reply.status(201).send();
23  });
24 }
```

12 - Tipagem do knex



```
1 import { Knex } from 'knex';
2
3 declare module 'knex/types/tables' {
4   export interface Tables {
5     transactions: {
6       id: string;
7       session_id?: string;
8       title: string;
9       amount: number;
10      created_at: string;
11    }
12  }
13 }
```

13 – Listagem das transações



```
EXPLORER
ROCKETSEAT_NODE
2023
01_fundamentos_nodejs
02_api_rest_nodejs
db
node_modules
src
  @types
  env
  routes
    transaction.ts
    database.ts
    server.ts
  .env
  .env.example
  knexfile.ts
  package.json
  package-lock.json
  README.md
  tsconfig.json
  ~$api_rest_node.docx
  01_fundamentos_node.docx
  02_api_rest_node.docx M

transaction.ts M X
2023 > 02_api_rest_nodejs > src > routes > transaction.ts > transactionsRoutes
1 import { knex } from '../database';
2 import { z } from 'zod';
3 import { FastifyInstance } from 'fastify';
4 import crypto from 'node:crypto';
5
6 export async function transactionsRoutes(app: FastifyInstance) {
7   app.get('/', async () => {
8     const transactions = await knex('transactions').select();
9     return { transactions };
10  });
11
12   app.get('/:id', async (request) => {
13     const getTransactionParamsSchema = z.object({
14       id: z.string().uuid()
15     });
16
17     const { id } = getTransactionParamsSchema.parse(request.params);
18
19     const transaction = await knex('transactions').where('id', id).first();
20     return { transaction };
21  });
22 }
```

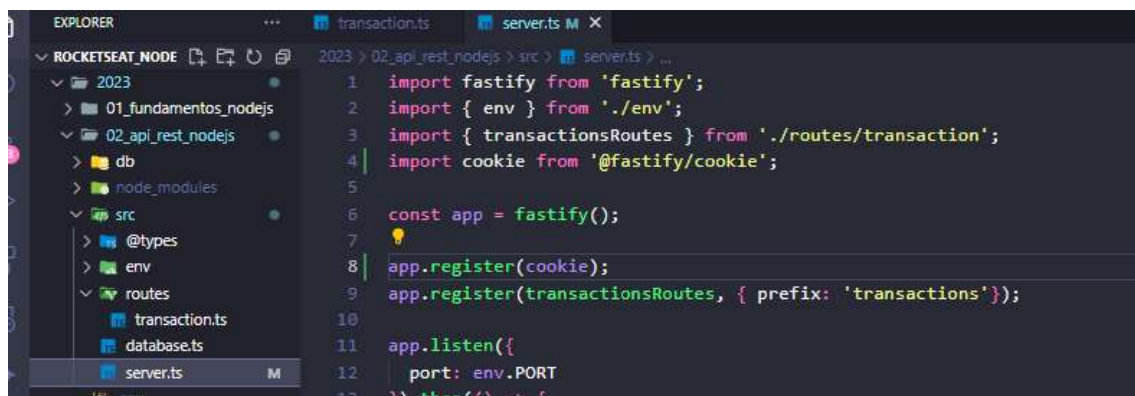
14 – Resumo de transações



```
transaction.ts M X
2023 > 02_api_rest_nodejs > src > routes > transaction.ts > transactionsRoutes
1 export async function transactionsRoutes(app: FastifyInstance) {
2   app.get('/', async () => {
3     const transactions = await knex('transactions').select();
4     return { transactions };
5   });
6
7   app.get('/summary', async () => {
8     const summary = (await knex('transactions').sum('amount', { as: 'amount' }).first());
9     return { summary };
10  });
11 }
```

15 – Utilizando cookies no fastify

Instalar `npm i @fastify/cookie`



```
EXPLORER
ROCKETSEAT_NODE
2023
01_fundamentos_nodejs
02_api_rest_nodejs
db
node_modules
src
  @types
  env
  routes
    transaction.ts
    database.ts
    server.ts
  .env
  .env.example
  knexfile.ts
  package.json
  package-lock.json
  README.md
  tsconfig.json
  ~$api_rest_node.docx
  01_fundamentos_node.docx
  02_api_rest_node.docx M

server.ts M X
2023 > 02_api_rest_nodejs > src > server.ts > ...
1 import fastify from 'fastify';
2 import { env } from './env';
3 import { transactionsRoutes } from './routes/transaction';
4 import cookie from '@fastify/cookie';
5
6 const app = fastify();
7
8 app.register(cookie);
9 app.register(transactionsRoutes, { prefix: 'transactions' });
10
11 app.listen({
12   port: env.PORT
13 }, then(() => {
```



```
2023 > 02_api_rest_nodejs > src > routes > transaction.ts > transactionsRoutes > app.post('/') callback
6 export async function transactionsRoutes(app: FastifyInstance) {
27
28   app.post('/', async (request, reply) => {
29     const createTransactionBodySchema = z.object({
30       title: z.string(),
31       amount: z.number(),
32       type: z.enum(['credit', 'debit'])
33     });
34
35     const { amount, title, type } = createTransactionBodySchema.parse(request
body);
36
37     let sessionId = request.cookies.sessionId;
38
39     if (!sessionId) {
40       sessionId = randomUUID();
41
42       reply.cookie('sessionId', sessionId, {
43         path: '/',
44         maxAge: 100 * 60 * 60 * 24 * 7
45       });
46
47       await knex('transactions').insert({
48         id: randomUUID(),
49         title,
50         amount: type === 'credit' ? amount : amount * -1,
51         session_id: sessionId
52       });
53
54       return reply.status(201).send();
55     }
56   });
57 }
```

16 – Validando existência do cookie

```
2023 > 02_api_rest_nodejs > src > middleware > check-session-id-exists > checkSessionIdExists
1 import { FastifyReply, FastifyRequest } from "fastify";
2
3 export async function checkSessionIdExists(request: FastifyRequest, reply: FastifyReply) {
4   const sessionId = request.cookies.sessionId;
5
6   if (!sessionId) {
7     return reply.status(401).send({ error: "Unauthorized." });
8   }
9 }
```

```
1 import { knex } from '../database';
2 import { z } from 'zod';
3 import { FastifyInstance } from 'fastify';
4 import { randomUUID } from 'node:crypto';
5 import { checkSessionIdExists } from '../middleware/check-session-id-exists';
6
7 export async function transactionsRoutes(app: FastifyInstance) {
8   app.get('/', {
9     preHandler: [checkSessionIdExists]
10   }, async (request, reply) => {
11     const { sessionId } = request.cookies;
12
13     const transactions = await knex('transactions')
14       .where('session_id', sessionId)
15       .select();
16     return { transactions };
17   });
18
19   app.get('/:id', {
20     preHandler: [checkSessionIdExists]
21   }, async (request) => {
22     const getTransactionParamsSchema = z.object({
23       id: z.string().uuid()
24     });
25
26     const { id } = getTransactionParamsSchema.parse(request.params);
27     const { sessionId } = request.cookies;
28
29     const transaction = await knex('transactions')
30       .where({ id, session_id: sessionId })
31       .first();
32     return { transaction };
33   });
34 }
```

17 – Configurando um hook global

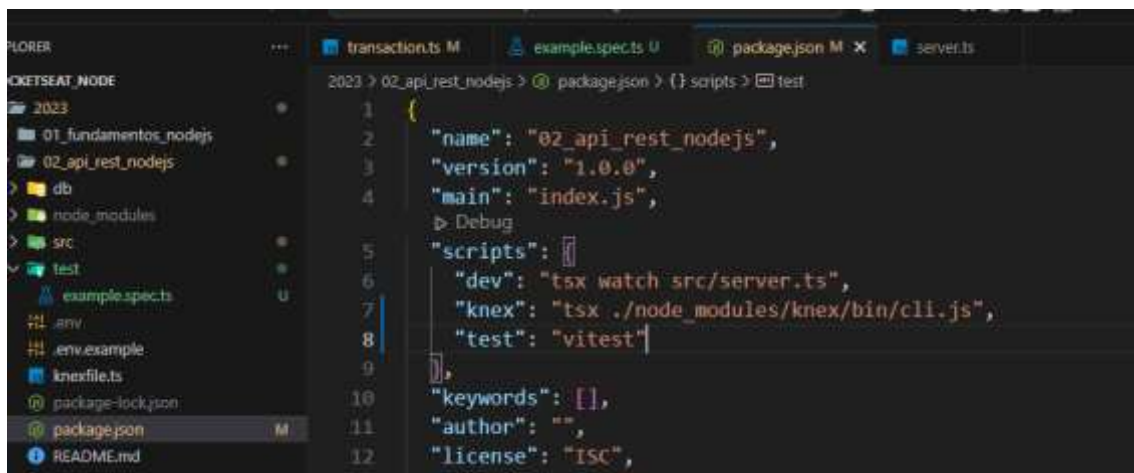
```
1 import { knex } from '../database';
2 import { z } from 'zod';
3 import { FastifyInstance } from 'fastify';
4 import { randomUUID } from 'node:crypto';
5 import { checkSessionIdExists } from '../middleware/check-session-id-exists';
6
7 export async function transactionsRoutes(app: FastifyInstance) {
8   app.addHook('preHandler', async (request, reply) => {
9     await checkSessionIdExists(request, reply);
10   });
11
12   app.get('/', {
13     preHandler: [checkSessionIdExists]
14   }, async (request, reply) => {
15     const { sessionId } = request.cookies;
```

18 – Criando primeiro teste

Instalar o npm i vitest -D



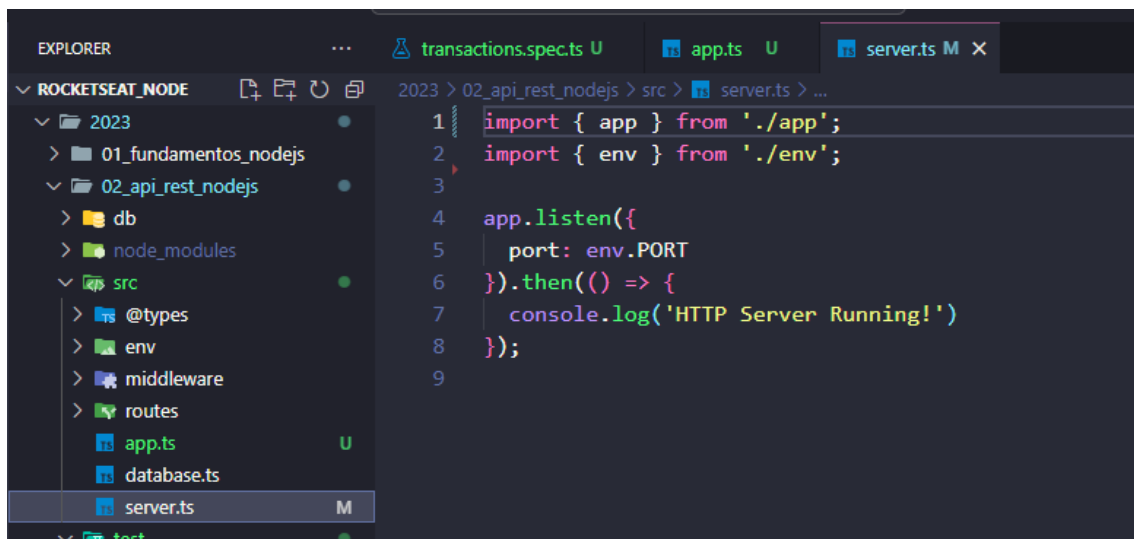
```
2023 > 02_api_rest_nodejs > test > example.spec.ts > ...
1 import { expect, test } from 'vitest';
2
3 test('0 usuário consegue criar uma nova transação', () => {
4   const responseStatusCode = 201;
5   expect(responseStatusCode).toEqual(201);
6 });
```



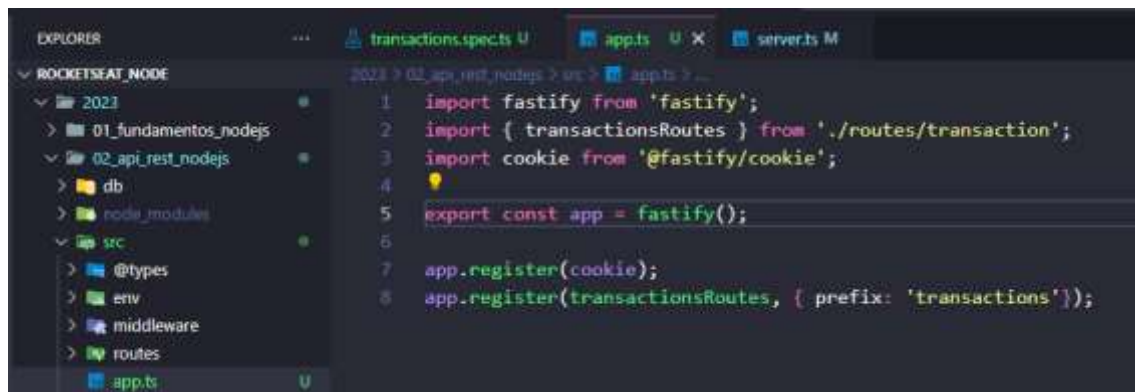
```
2023 > 02_api_rest_nodejs > package.json > {} scripts > test
1 {
2   "name": "02_api_rest_nodejs",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "dev": "tsx watch src/server.ts",
7     "knex": "tsx ./node_modules/knex/bin/cli.js",
8     "test": "vitest"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
```

19 - Testando criação de transação

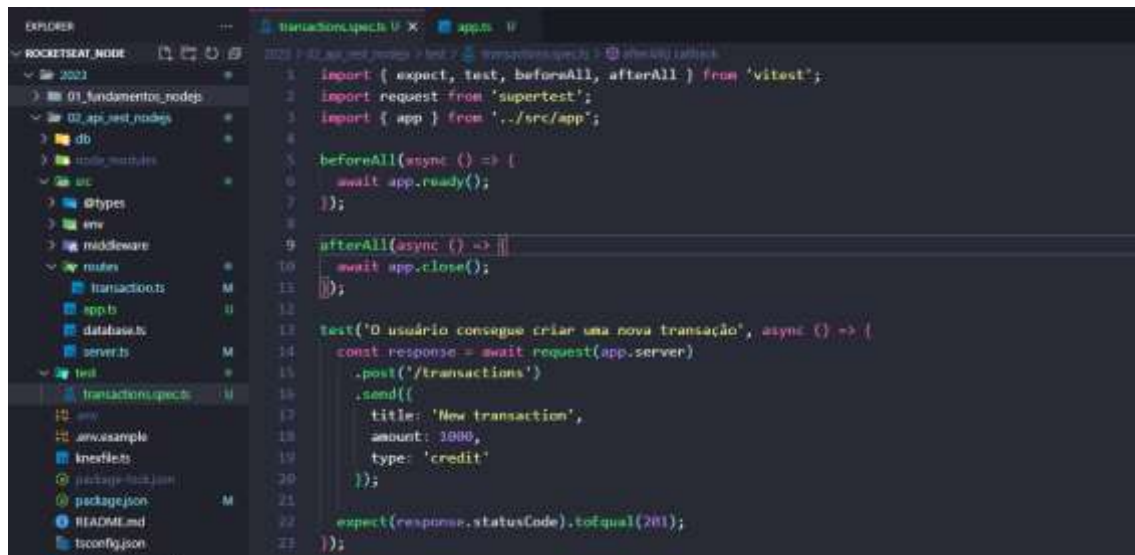
Instalar o `npm i supertest -D` e depois `npm i @types/supertest -D`



```
2023 > 02_api_rest_nodejs > src > server.ts > ...
1 import { app } from './app';
2 import { env } from './env';
3
4 app.listen({
5   port: env.PORT
6 }).then(() => {
7   console.log('HTTP Server Running!')
8 });
9
```



```
1 import fastify from 'fastify';
2 import { transactionsRoutes } from './routes/transaction';
3 import cookie from '@fastify/cookie';
4
5 export const app = fastify();
6
7 app.register(cookie);
8 app.register(transactionsRoutes, { prefix: 'transactions' });
```



```
1 import { expect, test, beforeAll, afterAll } from 'vitest';
2 import request from 'supertest';
3 import { app } from '../src/app';
4
5 beforeAll(async () => {
6   await app.ready();
7 });
8
9 afterAll(async () => {
10   await app.close();
11 });
12
13 test('O usuário consegue criar uma nova transação', async () => {
14   const response = await request(app.server)
15     .post('/transactions')
16     .send({
17       title: 'New transaction',
18       amount: 1000,
19       type: 'credit'
20     });
21   expect(response.statusCode).toEqual(201);
22 });
```

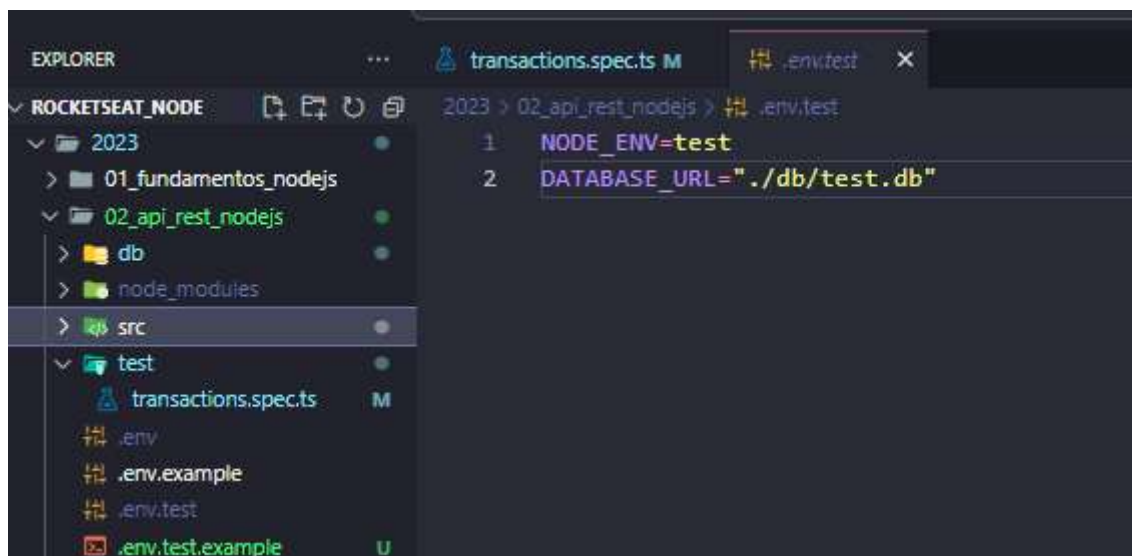
20 - Categorizando os testes

```
transactions.spec.ts M X app.ts
023 > 02_api_rest_nodejs > test > transactions.spec.ts > ...
1  import { expect, it, beforeAll, afterAll, describe } from 'vitest';
2  import request from 'supertest';
3  import { app } from '../src/app';
4
5  describe('Transactions routes', () => {
6    beforeAll(async () => {
7      await app.ready();
8    });
9
10   afterAll(async () => {
11     await app.close();
12   });
13
14   it('should be able to create a new transaction', async () => {
15     const response = await request(app.server)
16       .post('/transactions')
17       .send({
18         title: 'New transaction',
19         amount: 1000,
20         type: 'credit'
21       });
22
23     expect(response.statusCode).toEqual(201);
24   });
25 });
```

21 – Testando listagem de transações

```
transactions.spec.ts M X transaction.ts app.ts
test > transactions.spec.ts > describe('Transactions routes') callback > it('should be able to list all transactions') callback >
5 describe('Transactions routes', () => {
25
26   it('should be able to list all transactions', async () => {
27     const createTransactionResponse = await request(app.server)
28       .post('/transactions')
29       .send({
30         title: 'New transaction',
31         amount: 1000,
32         type: 'credit'
33       });
34
35     const cookies = createTransactionResponse.get('Set-Cookie');
36
37     const listTransactionResponse = await request(app.server)
38       .get('/transactions')
39       .set('Cookie', cookies!)
40       .expect(200);
41
42     expect(listTransactionResponse.body.transactions).toEqual([
43       expect.objectContaining({
44         title: 'New transaction',
45         amount: 1000,
46       })
47     ]);
48   });
49 });
```

22 – Configurando banco de testes



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure under the 'ROCKETSEAT_NODE' workspace. The '2023' folder is expanded, showing subfolders '01_fundamentos_nodejs' and '02_api_rest_nodejs'. The '02_api_rest_nodejs' folder is further expanded, revealing 'db', 'node_modules', 'src', and 'test' folders. The 'test' folder is selected, showing files: 'transactions.spec.ts', '.env', '.env.example', '.env.test', and '.env.test.example'. The main editor area shows the content of the '.env.test' file, which contains two lines of configuration:

```
1 NODE_ENV=test
2 DATABASE_URL="./db/test.db"
```



```
transactions.spec.ts M X .env.test
2023 > 02_api_rest_nodejs > test > transactions.spec.ts > describe('Transactions routes') callback
1 import { expect, it, beforeAll, afterAll, describe, beforeEach } from 'vitest';
2 import request from 'supertest';
3 import { app } from '../src/app';
4 import { execSync } from 'node:child_process';
5
6 describe('Transactions routes', () => {
7   beforeAll(async () => {
8     await app.ready();
9   });
10
11   afterAll(async () => {
12     await app.close();
13   });
14
15   beforeEach(() => {
16     execSync('npm run knex migrate:rollback --all')
17     execSync('npm run knex migrate:latest')
18   });
19
20   it('should be able to create a new transaction', async () => {
21     const response = await request(app.server)
22       .post('/transactions')
23       .send({
24         title: 'New transaction',
25         amount: 1000,
26         type: 'credit'
27       });
28
29     expect(response.statusCode).toEqual(201);
30   });
31
```

23 - Finalizando testes da aplicação

```
transactions.spec.ts M • .env.test X
2023 > 02_api_rest_nodejs > test > transactions.spec.ts > describe("Transactions routes") callback
6 describe('Transactions routes', () => {
55
56   it('should be able to get a specific transaction', async () => {
57     const createTransactionResponse = await request(app.server)
58       .post('/transactions')
59       .send({
60         title: 'New transaction',
61         amount: 1000,
62         type: 'credit'
63       });
64
65     const cookies = createTransactionResponse.get('Set-Cookie');
66
67     const listTransactionResponse = await request(app.server)
68       .get('/transactions')
69       .set('Cookie', cookies!)
70       .expect(200);
71
72     const transactionId = listTransactionResponse.body.transactions[0].id;
73
74     const getTransactionResponse = await request(app.server)
75       .get(`/transactions/${transactionId}`)
76       .set('Cookie', cookies!)
77       .expect(200);
78
79     expect(getTransactionResponse.body.transaction).toEqual(
80       expect.objectContaining({
81         title: 'New transaction',
82         amount: 1000,
83       })
84     );
85   });
86 }
```

```
transactions.spec.ts M x .env.test
2023 > 02_api_rest_nodejs > test > transactions.spec.ts > describe("Transactions routes") callback > it("should be able to get the su
6 describe('Transactions routes', () => {
87   it('should be able to get the summary', async () => {
88     const createTransactionResponse = await request(app.server)
89       .post('/transactions')
90       .send({
91         title: 'New transaction',
92         amount: 1000,
93         type: 'credit'
94       });
95
96     const cookies = createTransactionResponse.get('Set-Cookie');
97
98     await request(app.server)
99       .post('/transactions')
100       .set('Cookie', cookies!)
101       .send({
102         title: 'Debit transaction',
103         amount: 100,
104         type: 'debit'
105       });
106
107     const summaryResponse = await request(app.server)
108       .get('/transactions/summary')
109       .set('Cookie', cookies!)
110       .expect(200);
111
112     expect(summaryResponse.body.summary).toEqual({
113       amount: 900
114     });
115   });
116 });
```

24 – Preparando para deploy

Antes de subir o projeto no ar, converta ele para js, baixe a lib [npm](#) i [tsup](#) -D

```
transactions.spec.ts x package.json M x .gitignore
2023 > 02_api_rest_nodejs > package.json > {} devDependencies
1 {
2   "name": "02_api_rest_nodejs",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "dev": "tsx watch src/server.ts",
7     "knex": "tsx ./node_modules/knex/bin/cli.js",
8     "test": "vitest",
9     "build": "tsup src --out-dir build"
10  }
```

25 – Deploy do app no render