

Fundamentos de Banco de Dados

Aula 3

Msc. Maiara Coelho

Instituto de Computação

UFAM

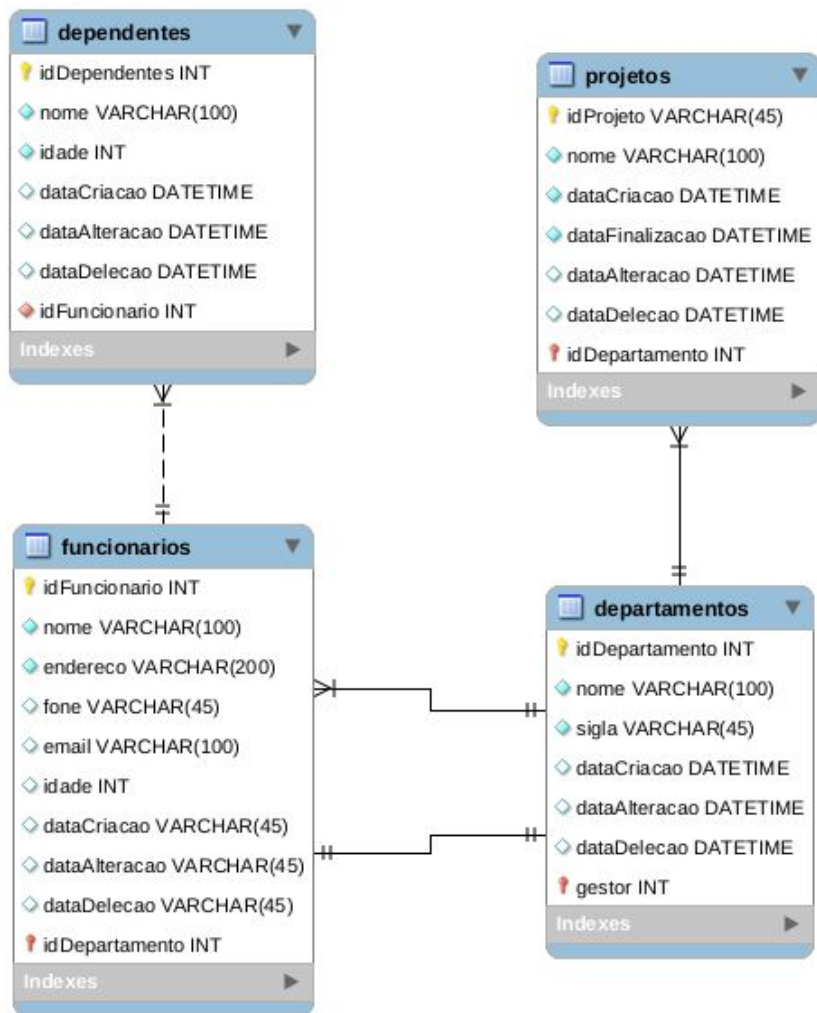
Completando nossa API

Exercício 5

1. Crie o restante das tabelas do modelo e faça as associações necessárias.

Crie CRUDs para o restante dos models

Obs: Esse exercício é para ser entregue ainda hoje (15/06/2023) via colabweb, ele é parte da nota 3 e vale 2 pontos. Compactar o código e enviar.



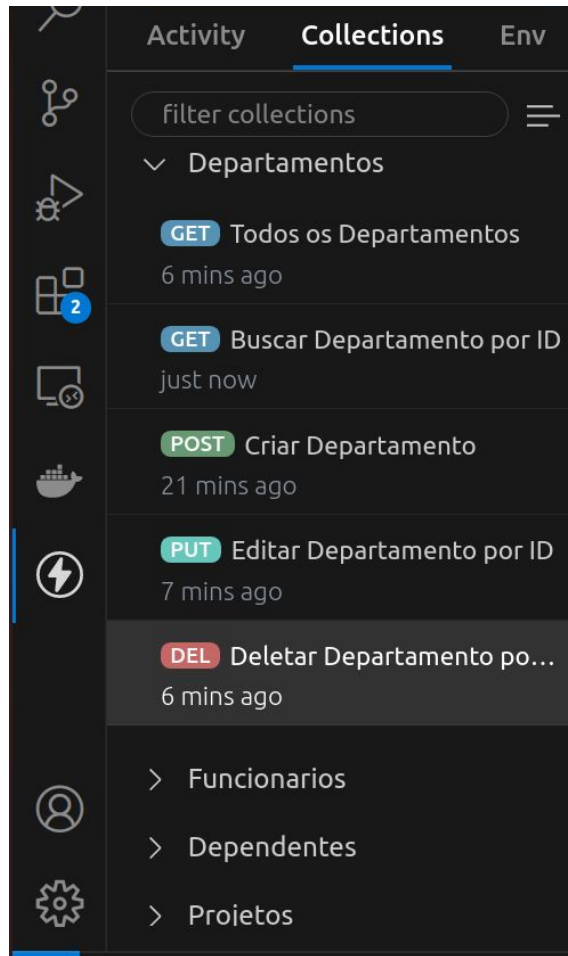
Testando as Rotas com Thunder Client



- Cliente para testar APIs
- Extensão no Visual Studio Code
- Documentação
 - <https://github.com/rangav/thunder-client-support>
- Tutorial de utilização
 - <https://rangav.medium.com/thunder-client-cli-a-new-way-to-test-apis-inside-vscode-d91eb5c71d8e>

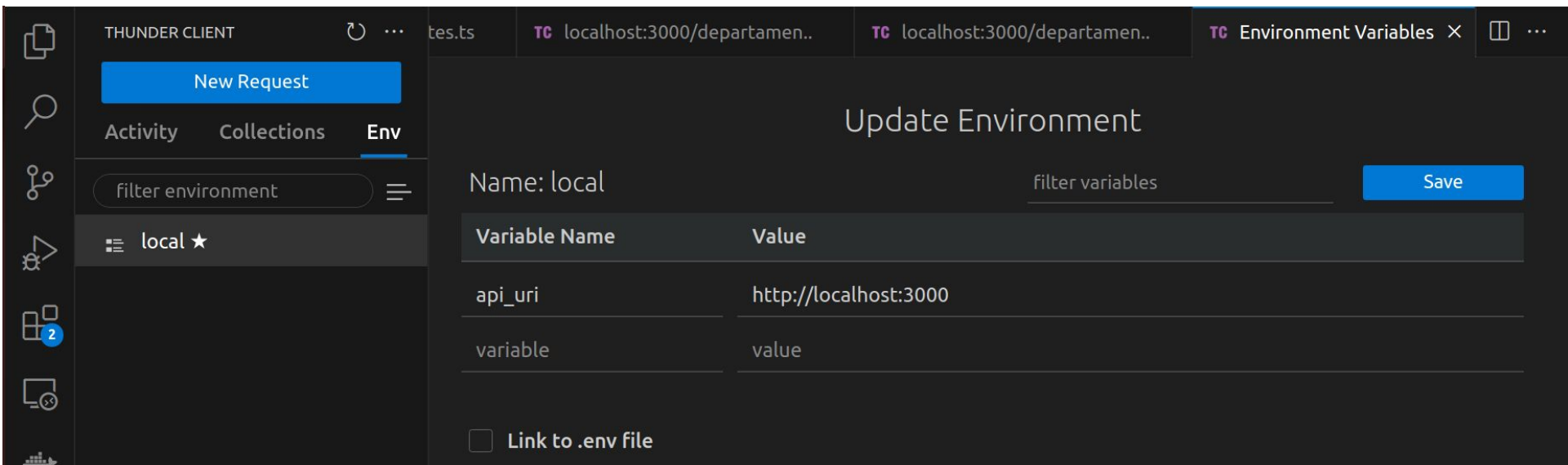
Criando collections

- Collections são uma forma de organizar melhor as rotas da API
- Cada model pode ter a sua collection e uma collection tem seu conjunto de rotas (CRUD).



Criando .env

- As variáveis de ambiente, `{{variavel}}`, ajudam a referenciar valores que geralmente se repetem.
- Por exemplo, a url da API na figura abaixo, `{{api_uri}}`.



Testando GET

The screenshot displays the Thunder Client application interface. On the left sidebar, the 'Collections' tab is active, showing a list of requests under the 'Departamentos' collection. The first request, 'GET Todos os Departamentos', is selected and shows a timestamp of '6 mins ago'. Below it are 'GET Buscar Departamento por ID' (just now), 'POST Criar Departamento' (21 mins ago), 'PUT Editar Departamento por ID' (7 mins ago), and 'DEL Deletar Departamento po...' (6 mins ago). At the bottom of the sidebar are expandable sections for 'Funcionarios', 'Dependentes', and 'Projetos'.

The main panel shows the details of the selected 'GET Todos os Departamentos' request. The URL bar contains 'GET' and '{{api_uri}}/departamentos', with a 'Send' button on the right. Below the URL bar are tabs for 'Query', 'Headers 2', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Query' tab is active, showing a 'Query Parameters' section with a table for defining parameters.

parameter	value

Below the query parameters, the response status is shown as 'Status: 200 OK', 'Size: 339 Bytes', and 'Time: 6 ms'. A 'Response' dropdown is on the right. The response body is a JSON array with one object, displayed with line numbers 1 through 11.

```
1  [  
2    {  
3      "id": "9d349960-0ac1-11ee-94b4-33f9f726fc1f",  
4      "name": "name 4",  
5      "sigla": "GLO",  
6      "gestorId": null,  
7      "createdAt": "2023-06-14T14:41:57.000Z",  
8      "updatedAt": "2023-06-14T14:41:57.000Z"  
9    },  
10   {  
11     "id": "7d4400-005f-11ee-94b4-33f9f726fc1f"
```

Testando GET

The screenshot displays the Thunder Client application interface. The top bar shows the application name 'THUNDER CLIENT' and several active tabs: 'TC Todos os Funcionarios', 'TC Buscar Funcionario por ID', and 'TC Buscar Departamento por ID'. The left sidebar contains a 'Collections' view with a tree structure showing 'Departamentos' and 'Funcionarios'. The main panel is divided into tabs for 'Query', 'Headers', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Headers' tab is selected, showing a list of HTTP headers. Below the headers, the status bar indicates a successful request with 'Status: 200 OK', 'Size: 167 Bytes', and 'Time: 3 ms'. The response panel at the bottom shows a JSON object representing a department.

THUNDER CLIENT

TC Todos os Funcionarios TC Buscar Funcionario por ID TC Buscar Departamento por ID

New Request

Activity Collections Env

filter collections

Departamentos

GET Todos os Departamentos 29 mins ago

GET Buscar Departamento por ID just now

POST Criar Departamento 21 mins ago

PUT Editar Departamento por ID 7 mins ago

DEL Deletar Departamento po... 6 mins ago

Funcionarios

GET {{api_uri}}/departamentos/a7dc4d80-095f-11ee-9eb5-13299ce668e5

Send

Query Headers 2 Auth Body Tests Pre Run

HTTP Headers

Accept */*

User-Agent Thunder Client (https://www.thunderclient.com)

header value

Status: 200 OK Size: 167 Bytes Time: 3 ms

Response

```
1 {
2   "id": "a7dc4d80-095f-11ee-9eb5-13299ce668e5",
3   "name": "name",
4   "sigla": "GLO",
5   "gestorId": null,
6   "createdAt": "2023-06-12T20:28:13.000Z",
7   "updatedAt": "2023-06-12T20:28:13.000Z"
8 }
```

Testando POST

The screenshot displays the Thunder Client interface. On the left sidebar, the 'Collections' tab is active, showing a list of API collections. The 'Departamentos' collection is expanded, revealing several requests. The 'POST Criar Departamento' request, made 21 minutes ago, is selected. The main panel shows the details of this request. The method is 'POST' and the URL is '{{api_uri}}/departamentos'. The 'Body' tab is selected, showing a JSON payload. Below the body, the status is '201 Created', the size is '153 Bytes', and the time taken is '29 ms'. The response body is also shown in JSON format.

THUNDER CLIENT

New Request

Activity Collections Env

filter collections

▼ Departamentos

- GET Todos os Departamentos
29 mins ago
- GET Buscar Departamento por ID
just now
- POST Criar Departamento**
21 mins ago
- PUT Editar Departamento por ID
7 mins ago
- DEL Deletar Departamento po...
6 mins ago

> Funcionarios

> Dependentes

TC Criar Departamento X TC Environment Variables TC Editar Departamento por I.. TC Deletar Depart ...

POST {{api_uri}}/departamentos Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

```
1 {
2   "name": "name 4",
3   "sigla": "GLO"
4 }
```

Status: 201 Created Size: 153 Bytes Time: 29 ms Response ▼

```
1 {
2   "id": "9d349960-0ac1-11ee-94b4-33f9f726fc1f",
3   "name": "name 4",
4   "sigla": "GLO",
5   "updatedAt": "2023-06-14T14:41:57.750Z",
6   "createdAt": "2023-06-14T14:41:57.750Z"
7 }
```

Copy

Testando PUT

The screenshot displays the Thunder Client interface with a PUT request configured to update a department. The left sidebar shows a collection named 'Departamentos' with several requests, including the active 'PUT Editar Departamento por ID' request made 7 minutes ago. The main panel shows the request details: the method is PUT, the URL is `{{api_uri}}/departamentos/9d349960-0ac1-11ee-94b4-33f9f726fc1f`, and the body is a JSON object with `"name": "name Alterado"`. The request has been successfully executed, returning a 200 OK status with a response body containing detailed information about the updated department, including its ID, name, sigla, gestorId, and timestamps for createdAt and updatedAt.

THUNDER CLIENT

New Request

Activity Collections Env

filter collections

Departamentos

- GET Todos os Departamentos
29 mins ago
- GET Buscar Departamento por ID
just now
- POST Criar Departamento
21 mins ago
- PUT Editar Departamento por ID**
7 mins ago
- DEL Deletar Departamento po...
6 mins ago

> Funcionarios

> Dependentes

TC Criar Departamento TC Environment Variables **TC Editar Departamento por I..** X TC Deletar Depart

PUT `{{api_uri}}/departamentos/9d349960-0ac1-11ee-94b4-33f9f726fc1f` Send

Query Headers 2 Auth **Body 1** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

```
1 {  
2   "name": "name Alterado"  
3 }
```

Status: 200 OK Size: 176 Bytes Time: 19 ms Response

```
1 {  
2   "id": "9d349960-0ac1-11ee-94b4-33f9f726fc1f",  
3   "name": "name Alterado",  
4   "sigla": "GLO",  
5   "gestorId": null,  
6   "createdAt": "2023-06-14T14:41:57.000Z",  
7   "updatedAt": "2023-06-14T14:56:13.000Z"  
8 }
```

Testando DELETE

The screenshot shows the Thunder Client interface with a list of API collections on the left and a detailed view of a DELETE request on the right.

Left Panel (Collections):

- THUNDER CLIENT
- New Request
- Activity
- Collections
- Env
- filter collections
- Departamentos
 - GET Todos os Departamentos (29 mins ago)
 - GET Buscar Departamento por ID (just now)
 - POST Criar Departamento (21 mins ago)
 - PUT Editar Departamento por ID (7 mins ago)
 - DEL Deletar Departamento po... (6 mins ago)
- Funcionarios
- Dependentes

Right Panel (Request Details):

- Method: DELETE
- URL: `{{api_uri}}/departamentos/9d349960-0ac1-11ee-94b4-33f9f726fc1f`
- Send button
- Tabs: Query, Headers 2, Auth, Body, Tests, Pre Run
- Query Parameters: parameter, value
- Status: 200 OK, Size: 176 Bytes, Time: 16 ms
- Response:

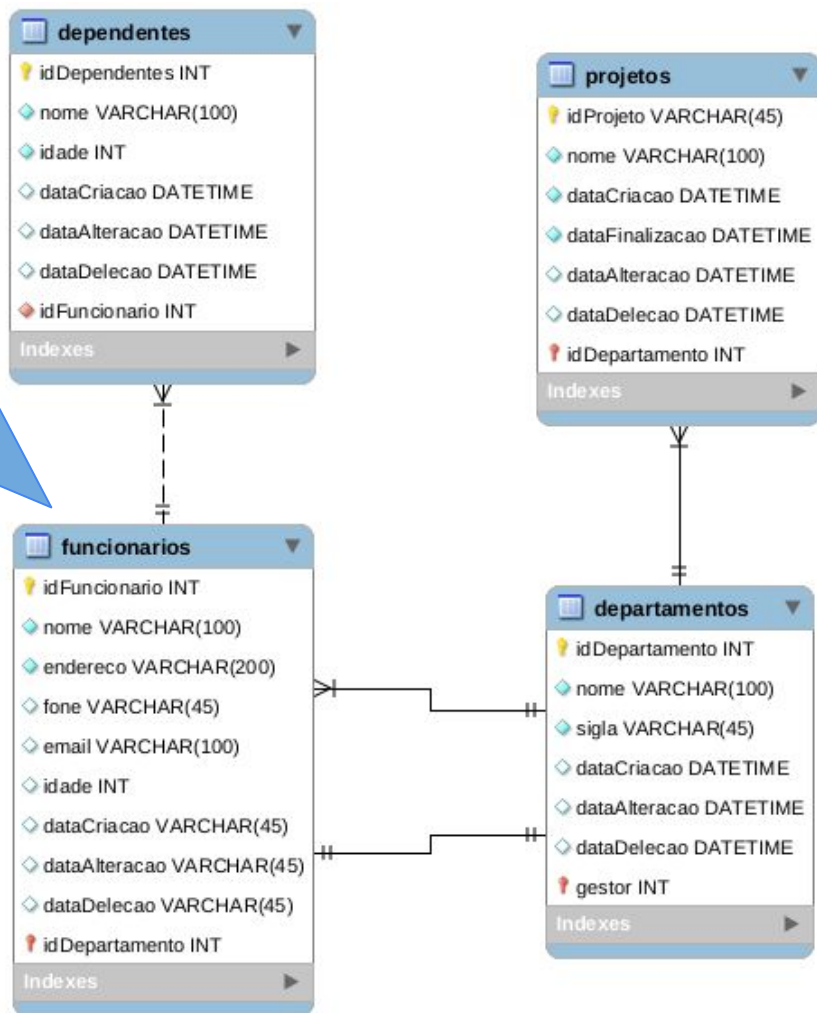
```
1 {
2   "id": "9d349960-0ac1-11ee-94b4-33f9f726fc1f",
3   "name": "name Alterado",
4   "sigla": "GLO",
5   "gestorId": null,
6   "createdAt": "2023-06-14T14:41:57.000Z",
7   "updatedAt": "2023-06-14T14:56:13.000Z"
8 }
```

Refinando nossas consultas

Exercício 5

2. Faça includes de funcionário em Departamento
3. Faça includes de dependentes e departamento em funcionários.
4. Faça com que na listagem de funcionários venha a quantidade de registros.

Obs: Esse exercício é para ser entregue ainda hoje (15/06/2023) via colabweb, ele é parte da nota 3 e vale 2 pontos. Compactar o código e enviar.

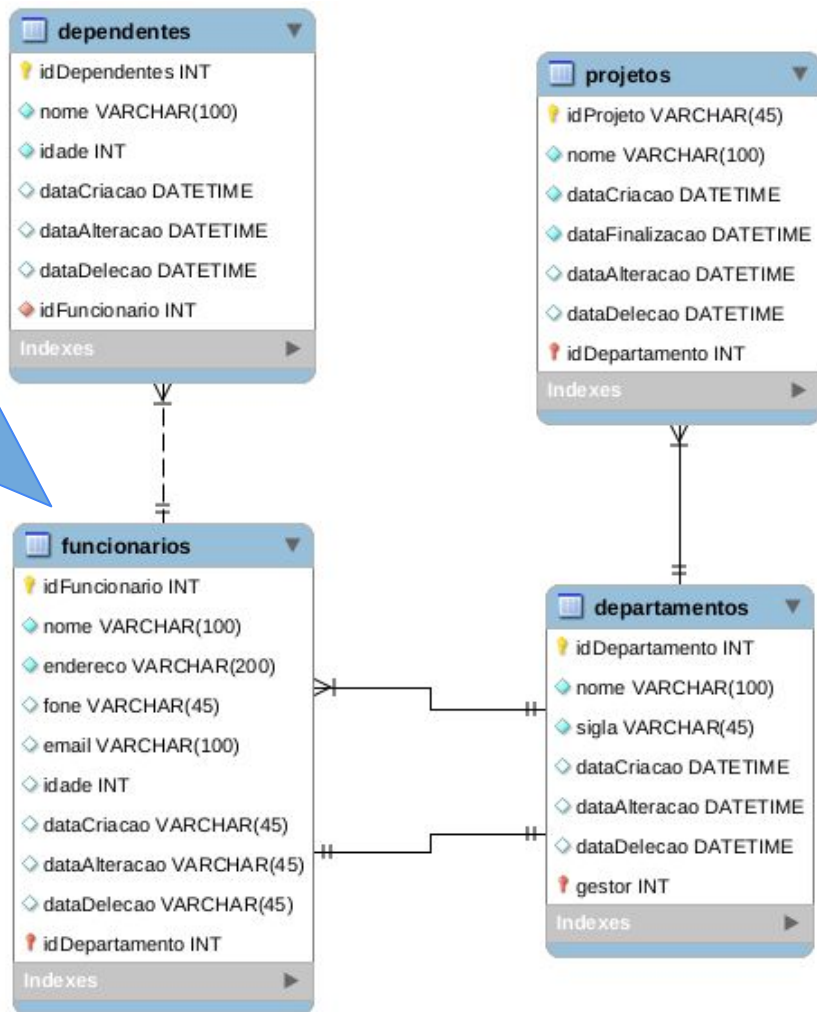


Refinando nossas consultas

Exercício 5

5. Faça uma rota que recupere todos os dependentes de um único funcionário. A rota deve receber por parâmetro o id do funcionário.

Obs: Esse exercício é para ser entregue ainda hoje (15/06/2023) via colabweb, ele é parte da nota 3 e vale 2 pontos. Compactar o código e enviar.



Migrações

- Criar uma forma de registrar a versão atual do banco
 - `api.info.ts`
 - Tabela/model de versão: `models/VersionDB.ts`
- Criar arquivo que receberá as consultas SQL: `db/migracoes.ts`
- Modificações em `db/config`
- Modificações em `server.ts`
- Modificações em `main.ts`

Migrações: api.info.ts

```
// api.info.ts
export const api = {
  name: 'API-EMPRESA',
  defaultPort: 3000,
  db: {
    id: '043577f0-0b22-11ee-be56-0242ac120002',
    dbVersion: 0,
  },
};
```

Migrações: models/VersaoDB.ts

```
import { Table, Column, Model, PrimaryKey,
DataType, AllowNull, IsUUID } from
'sequelize-typescript';
@Table({
  paranoid: true,
  timestamps: true,
})
export class VersaoDB extends Model {
  @IsUUID('all')
  @PrimaryKey
  @Column({
    type: DataType.UUID,
    defaultValue: DataType.UUIDV1,
  })
  id!: string;
```

```
    @AllowNull(false)
    @Column({
      type: DataType.INTEGER,
      defaultValue: 0,
    })
    numeroVersao!: number;
  }
```

oe56-0242ac120002',

Migrações: db/migracoes.ts

```
export interface MigracaoDB {
  consultas?: Array<{ model: string, query: string }>
}

const migracoes: Map<number, MigracaoDB> = new Map<number, MigracaoDB>();

// migracoes.set(1, {
//   consultas: [
//     {
//       model: 'Dependentes',
//       query: `ALTER TABLE "Dependentes" ALTER COLUMN name SET DATA TYPE
text;`,
//     },
//   ],
// });

export { migracoes };
```


Migrações: modificações em db/config.ts

- Retirada dos models

```
import { Sequelize } from "sequelize-typescript";

const connection = new Sequelize({
  dialect: "mysql",
  host: "localhost",
  username: "root",
  password: "mySQL@2023",
  database: "mydb",
  logging: false,
});

export default connection;
```

Migrações: modificações em server.ts

- Adicionar routes e models para ficarem centralizados
- Função de inicialização do serviço
- A migração pode ocorrer no server.ts assim que o servidor é iniciado
- Função de inicialização de Models: sincronizar os modelos
- Função para fazer a migração
 - Comparação entre as versões para saber se a versão do banco está atualizada ou não.
 - Se não estiver atualizado, fazer a migração

Migrações: modificações em server.ts

- Adicionar routes e models para ficarem centralizados
- Função de inicialização do serviço
- A migração pode ocorrer no server.ts assim que o servidor é iniciado
- Função de inicialização de Models: sincronizar os modelos
- Função para fazer a migração
 - Comparação entre as versões para saber se a versão do banco está atualizada ou não.
 - Se não estiver atualizado, fazer a migração

Migrações: adicionar Routes e Models em server.ts

```
...  
const routes = [  
  departamentosRouter,  
  projetosRouter,  
  funcionariosRouter,  
  //dependentesRouter,  
]  
  
const models = [  
  VersaoDB,  
  Funcionarios,  
  Departamentos,  
  Projetos,  
  // Dependentes,  
];  
...
```

Migrações: função de Inicialização do Serviço

```
async bootstrap():  
Promise<Api> {  
  try {  
    await this.middleware();  
    await this.router();  
    await this.initModels();  
    await this.migrations();  
  } catch (err) {  
    console.error(err);  
  }  
  
  return this;  
}
```

```
private async middleware() {  
  this.server.use(express.json());  
}  
  
private async router() {  
  this.server.use(routes);  
  try {  
    await  
this.server.listen(api.defaultPort);  
  } catch (err) {  
    console.error(err);  
    throw error;  
  }  
}
```

Migrações: função de inicialização de Models

```
private async initModels() {  
  await connection.authenticate()  
    .then(async () => {  
    console.info('MySQL DB Conectado!');  
    await connection.addModels(models);  
    await connection.sync();  
  })  
    .then(() => {  
    console.info('DB sync!');  
  })  
    .catch((err) => {  
    console.error(err);  
    throw error;  
  });  
}
```

Migrações: função de Migração

```
private async migrations() {

    let versaoAtualBanco = await VersaoDB.findByPk(api.db.id);
    let numeroVersaoAtualBanco = versaoAtualBanco == null ? 0 :
versaoAtualBanco.numeroVersao;

    console.info('VERSAO DO BANCO API-EMPRESA: ' + numeroVersaoAtualBanco);
    if (numeroVersaoAtualBanco < api.db.dbVersion) {
        console.info(migracoes);
        let models: string[] = [];

        for (let i = numeroVersaoAtualBanco; i < api.db.dbVersion; i++) {
            const migracao: MigracaoDB | undefined = migracoes.get(i + 1);
```

Migrações: função de Migração

```
if (migracao && migracao.consultas) {  
  if (migracao.consultas !== null) {  
    for (const consulta of migracao.consultas) {  
      console.info('executando: ' + consulta.query);  
      if (models.indexOf(consulta.model) < 0) {  
        await connection.query(consulta.query);  
        console.info('  executed!');  
      } else {  
        console.info('  not executed: new model.');      }  
    }  
  }  
}
```


Migrações: função de Migração

```
if (versaoAtualBanco == null) {  
  await VersaoDB.create({ id: api.db.id, numeroVersao: api.db.dbVersion });  
} else {  
  versaoAtualBanco.numeroVersao = api.db.dbVersion;  
  await versaoAtualBanco.save();  
}  
  
}  
  
await connection.sync()  
  .then(() => {  
    console.info('Models sync!');  
  })  
  .catch((error) => {  
    console.error(error);  
  });  
  
}  
  
}
```

Migrações: modificações em main.ts

```
const server = new Api();  
try {  
  server.bootstrap()  
    .then((server) => {  
      console.info(`API Empresa rodando na porta ${api.defaultPort}`);  
    });  
} catch (error) {  
  console.error('Server failed to start.');
```

```
  console.error(error);  
  process.exit(1);  
}
```

Migrações: adicionando nova tabela

- Basta que os novos models sejam adicionados no Array de models em `server.ts`
- A função do sequelize **addModels** permite que o model seja criado na inicialização do serviço.
- Ex: Dependentes
 - Delete a tabela de Dependentes e as referências a ele
 - Execute o serviço: `npm start`
 - Verifique que a tabela de Dependentes terá sido criada novamente.
- Para fazer qualquer alteração em tabelas existentes basta modificar o model, adicionar o comando SQL em `migracoes.ts` e mudar a versão da base em `api.info.ts`

Migrações: adicionando coluna nova em tabela

- Modificar o Model

```
@AllowNull (true)
@Column ({
    type: DataType.STRING,
})
atributo_adicionado!: string;
```

Migrações: adicionando coluna nova em tabela

- Adicionar consultas SQL em migracoes e mudar a versão em api.info.ts

```
// migracoes
migracoes.set(1, {
  consultas: [
    {
      model: 'Dependentes',
      query: `ALTER TABLE Dependentes ADD atributo_adicionado
VARCHAR(45);`,
    },
  ],
});
```

Migrações: adicionando coluna nova em tabela

```
MySQL DB Conectado!  
DB sync!  
VERSAO DO BANCO API-EMPRESA: 0  
Map(1) { 1 => { consultas: [ [Object] ] } }  
executando: ALTER TABLE Dependentes ADD atributo_adicionado VARCHAR(45);  
executed!  
Models sync!  
API Empresa rodando na porta 3000
```

- Cada consulta que está na versão chave do map e que corresponde à versão informada em api.info.ts será executada!

Migrações: adicionando coluna nova em tabela

- Adicione alguns departamentos, funcionários e dependentes.

```
DB sync!  
VERSAO DO BANCO API-EMPRESA: 1  
Map(2) {  
  1 => { consultas: [ [Object] ] },  
  2 => { consultas: [ [Object] ] }  
}  
executando: ALTER TABLE Dependentes ADD atributo_adicionado_2 VARCHAR(45) NOT NULL DEFAULT "";  
executed!  
Models sync!  
API Empresa rodando na porta 3000
```

- Se o atributo for NOT NULL e já existirem registros na tabela, precisa dar um valor padrão para que não dê erro na execução do SQL.

Migrações: alterando coluna em tabela

- Modificou o nome do atributo de “atributo_adicionado” para “endereco”.

```
migracoes.set(3, {  
  consultas: [  
    {  
      model: 'Dependentes',  
      query: `ALTER TABLE Dependentes CHANGE atributo_adicionado endereco  
VARCHAR(45);`,  
    },  
  ],  
});
```


Migrações: retirando coluna de tabela

- Retirando as colunas “endereco” e “atributo_adicionado_2”

```
migracoes.set(4, {
  consultas: [
    {
      model: 'Dependentes',
      query: `ALTER TABLE Dependentes DROP COLUMN endereco;`,
    },
    {
      model: 'Dependentes',
      query: `ALTER TABLE Dependentes DROP COLUMN atributo_adicionado_2;`,
    },
  ],
});
```

Migrações: deletando uma tabela

- Deletando a Tabela de Dependentes, essa operação deve ser feita com muito cuidado, pois refletirá em registros em outras tabelas nas quais essa estiver associada.
- Retirar Models e Routers que referenciam tal Tabela

```
migracoes.set(5, {  
  consultas: [  
    {  
      model: 'Dependentes',  
      query: `DROP TABLE Dependentes`,  
    },  
  ],  
});
```

Exceções do Sequelize

`SequelizeUniqueConstraintError` - unique violation

`SequelizeValidationError` - notNull Violation

`SequelizeDatabaseError` - Incorrect date value: 'Invalid date'

`SequelizeForeignKeyConstraintError`

- O tratamento de erros impede a API de sair do ar quando ocorrerem erros de entrada de usuário.

```
try{  
  // funções de comunicação com o banco  
  // retorno ao usuário  
}catch (error){  
  // retorno ao usuário  
}
```

Exercício 6

1 - Crie o esquema de migração manual para a sua Loja Virtual, inclua novos atributos e inclua novas versões do banco de dados e adicione tratamento de erros. Crie as collections no Thunder Cliente para consumo das rotas.

Faça um pequeno readme com todas os recursos de sua API, para guiar o avaliador.

Obs: Esse exercício é para ser entregue até o dia 20/06/2023, via colabweb, ele é parte da nota 3 e vale 8 pontos. Compactar o código, exportar todas as collections criadas e enviar os dois arquivos compactados.

Exercício 6: Modelo da Loja

