Aula 5 - Banco de dados

Tópicos especiais em Sistemas

Prof. Juliana Costa Silva - juliana.silva@up.edu.br

O que veremos hoje

1. Revendo...
2. Infraestrutura
3. Conexão
4. Criando tabelas

Revendo...

O que já aprendemos?

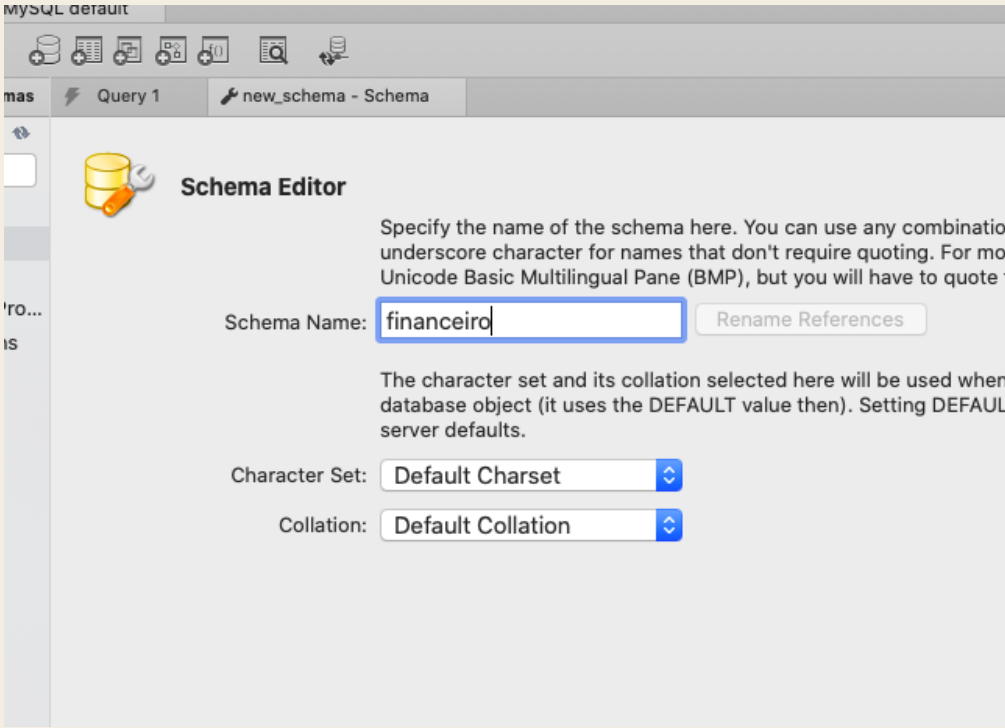
- Criamos um projeto node;
- organizamos os arquivos de configuração na pasta config;
- Organizamos os controladores na pasta controllers;
- Configuramos ações de GET e POST para a rota movimento.
- Enviamos dados em formato JSON

O projeto da disciplina

- Faremos um sistema de controle financeiro pessoal;
- Este sistema deve ter:
  - Registro de gastos;
  - Login de usuários;
  - Registro de renda (salários - comissões - negócios);
  - Registro de cartões de créditos;
  - Registro de contas bancárias;

Utilizaremos MySQL

- Para utilizarmos o MySQL instalaremos o package;
- No terminal digite npm install mysql
- No MySQL crie um banco de dados chamado **financeiro**;



Infraestrutura

Pasta de infraestrutura

- Essa pasta será responsável por arquivos de infraestrutura;
- Tudo o que é necessário para a aplicação funcionar além das regras de negócio;
- Crie uma pasta infra e dentro dela o arquivo **conncetion.js**.

Conexão

Para realizar a conexãoimportaremos o pacote **mysql**

- No arquivo **connection.js**;
- Vamos acrescentar o código com os dados da nossa conexão;

infra > JS connection.js > ...

```
1   const mysql = require('mysql')
2
3   const conexao = mysql.createConnection({
4       host: 'localhost',
5       port: 3306,
6       user: 'root',
7       password: 'sql12345',
8       database: 'financeiro'
9   })
10
11  module.exports = conexao
```

Fonte: O autor

Problemas

No Terminal, notaremos que que rodamos o servidor antes mesmo da mensagem do sucesso da conexão, então executou independentemente.

- Se a conexão não der certo a aplicação não deveria funcionar;
- Devemos iniciar a aplicação com a conexão ao banco de dados e após ela iniciar o app.
- Para isso acrescentaremos o else na inicialização;

Else

No else colocaremos a inicialização da aplicação, que só ocorrerá se erro for falso;

JS index.js > ...

```
1   const custemExpress = require('./config/customExpress')
2   const conexao = require('./infra/connection')
3
4   conexao.connect( erro => {
5       if(erro){
6           console.log(erro)
7       }else{
8           console.log('Conectado ao banco com sucesso!')
9           const app = custemExpress()
10          app.listen(3000, () => console.log('Servidor rodando na porta 3000'))
11      }
12  })
```

Fonte: O autor

tabelas.js

Criaremos um arquivo **tabelas.js**, na pasta infra.

Editando tabelas.js

- Nesse arquivo criaremos a classe Tabelas;
- Esta classe será responsável por criar tabelas no banco de dados;
- Dentro da classe Tabelas neste arquivo, aplicaremos o método init()
- O método init() iniciará nosso trabalho;
- Receberemos a conexao do banco de dados por parâmetro, que o deixará desacoplado sem que saiba de onde vem exatamente.
- Nesta conexão, escreveremos no console.log() para identificar se realmente a class está sendo chamada.

tabelas.js

infra > JS tabelas.js > ...

```
1   class Tabelas {
2       init(conexao) {
3           console.log('Tabelas foram chamadas')
4       }
5   }
6
7   module.exports = new Tabelas
```

Código no arquivo **tabelas.js**. **Fonte:** O autor

Teste Tabelas

index.js

- Já dentro de **index.js**, quando conectarmos com o banco, queremos que este conecte e já crie as tabelas que precisamos.
- Neste arquivo, importaremos Tabelas sendo igual a require() recebendo o caminho do **tabela.js**.
- Executaremos a importação e, antes de criarmos app igual a customExpress(), teremos Tabelas com .init() recebendo a conexao que estamos usando, importada da infraestrutura.

index.js

JS index.js > ...

```
1   const custemExpress = require('./config/customExpress')
2   const conexao = require('./infra/connection')
3   const Tabelas = require('./infra/tabelas')
4
5   conexao.connect( erro => {
6       if(erro){
7           console.log(erro)
8       }else{
9           console.log('Conectado ao banco com sucesso!')
10
11          Tabelas.init(conexao)
12          const app = custemExpress()
13
14          app.listen(3000, () => console.log('Servidor rodando na porta 3000'))
15      }
16  })
```

Código no arquivo **index.js**. **Fonte:** O autor

Método criarMovimento

- Agora poderemos criar as tabelas;
- No arquivo **tabelas.js**, geraremos um novo método chamado criarMovimentos() e usaremos esta conexao;
- Para conectarmos, precisaremos passar para nosso escopo com this no lugar de console.log();
- Então dentro do método, digitaremos this com .conexao chamando uma .query() que espera uma query SQL para podermos executar, e logo em seguida espera também uma função a ser executada na sequência.

tabelas.js

infra > JS tabelas.js > ...

```
1   class Tabelas {
2       init(conexao) {
3           console.log('Tabelas foram chamadas')
4       }
5
6       criarMovimento() {
7           const sql = 'CREATE TABLE movimento '+
8           '(id int NOT NULL AUTO_INCREMENT,'+
9           ' descricao varchar(50) NOT NULL,'+
10          ' valor double NOT NULL, '+
11          ' tipo varchar(1) NOT NULL,'+
12          ' observacoes text PRIMARY KEY(id))'
13
14          this.conexao.query()
15      }
16  }
17
18  module.exports = new Tabelas
```

Código no arquivo **tabelas.js**. **Fonte:** O autor

Ainda sobre o Método criarMovimento

Ainda não estamos executando a query()

- Após const sql, passaremos a função que será executada quando a tabela for criada.
- O primeiro parâmetro que receberá será o erro.
- Logo, se a query() não for executada, o sistema indicará algum erro.
- Portanto, if() recebendo o erro imprimirá a falha. Caso contrário, imprimirá a Tabela Movimentos criada com sucesso.
- O init() deverá chamar a tabela, e este método por sua vez está sendo chamado dentro do **index.js**.
- Logo, digitaremos this com .criarMovimentos().

tabelas.js

infra > JS tabelas.js > Tabelas > criarMovimento

```
5
6       criarMovimento() {
7           const sql = 'CREATE TABLE movimento '+
8           '(id int NOT NULL AUTO_INCREMENT,'+
9           ' descricao varchar(50) NOT NULL,'+
10          ' valor double NOT NULL, '+
11          ' tipo varchar(1) NOT NULL,'+
12          ' observacoes text PRIMARY KEY(id))'
13
14          this.conexao.query(sql, erro => {
15              if(erro){
16                  console.log(erro)
17              }else{
18                  console.log('Tabela movimento criada com sucesso!')
19              }
20          })
21      }
```

Código no arquivo **tabelas.js**. **Fonte:** O autor

tabelas.js chamando o método no init()

infra > JS tabelas.js > Tabelas > criarMovimento > [sql]

```
1   class Tabelas {
2       init(conexao) {
3           this.conexao = conexao
4           this.criarMovimento()
5       }
6
7       criarMovimento() {
8           const sql = 'CREATE TABLE movimento '+
9           '(id int NOT NULL AUTO_INCREMENT,'+
10          ' descricao varchar(50) NOT NULL,'+
11          ' valor double NOT NULL, '+
12          ' tipo varchar(10) NOT NULL,'+
13          ' observacoes text, PRIMARY KEY(id))'
14
15          this.conexao.query(sql, erro => {
16              if(erro){
17                  console.log(erro)
18              }else{
19                  console.log('Tabela movimento criada com sucesso!')
20              }
21          })
22      }
23  }
```

Código no arquivo **tabelas.js**. **Fonte:** O autor