06

Persistência e redirecionamento

Transcrição

Nessa etapa queremos finalizar a persistência da nossa lista de livros e criar um redirecionamento após um livro ser adicionado ao banco de dados.

Primeiramente, precisaremos do método `adiciona()` no nosso `livro-dao.js`, recebendo `req.body` (que representa o livro) e devolvendo uma `Promise` que será tratada na nossa rota `/livros`.

```
adiciona(livro) {
    return new Promise((resolve, reject) => {


    });
}
```

Agora temos que implementar o corpo dessa `Promise`, dizendo para a aplicação como ela deve fazer para pegar o `livro` que o método está recebendo e efetivamente inserindo-o no banco de dados.

Com `this`, pegaremos a instância do nosso banco (`_db`) e executaremos o `run()`, um método so SQLite utilizado para executar, no banco de dados, instruções que não retornem nenhum resultado, como as instruções de inserção, deleção e atualização.

Esse método receberá três parâmetros. O primeiro deles é a *string* representando a instrução que queremos executar no banco de dados - nesse caso, a instrução de inserção:

```
adiciona(livro) {
    return new Promise((resolve, reject) => {
        this._db.run(`
        INSERT INTO LIVROS (
                titulo,
                preco,
                descricao
            ) values (?, ?, ?)
        `,)

    });
}
```

Aqui, estamos inserindo nos campos `titulo`, `preco` e `descricao`, com os valores `?, ?, ?`. Cada uma dessas interrogações representa uma informação do nosso `livro` - ou seja, título, preço e descrição, respectivamente. Se tivéssemos mais informações, bastaria adicionarmos mais interrogações.

Como segundo parâmetro, passaremos um array com as informações que substituirão essas interrogações:

```
adiciona(livro) {
    return new Promise((resolve, reject) => {
        this._db.run(`
```

```
            INSERT INTO LIVROS (
                    titulo,
                    preco,
                    descricao
                ) values (?, ?, ?)
            `,
            [
                livro.titulo,
                livro.preco,
                livro.descricao
            ],
        )
        });
    }
```

Repare que, no array, é necessário manter exatamente a mesma ordem que foi colocada na instrução de inserção, de modo que as informações sejam inseridas nas colunas correspondentes.

O último parâmetro que passaremos é uma função *callback* que será executada ao final dessa inserção. Essa função receberá, como parâmetro, somente um erro (`err`) - lembrando que o método `run()` não retorna nenhum resultado, somente um erro (caso ocorra um).

Em caso de erro, ele será impresso no console, além de executarmos a função `reject()` dizendo "Não foi possível adicionar o livro!". Caso contrário, simplesmente resolveremos a `Promise` .

```
    adiciona(livro) {
        return new Promise((resolve, reject) => {
            this._db.run(`
                INSERT INTO LIVROS (
                        titulo,
                        preco,
                        descricao
                    ) values (?, ?, ?)
                `,
                [
                    livro.titulo,
                    livro.preco,
                    livro.descricao
                ],
                function (err) {
                    if (err) {
                        console.log(err);
                        return reject('Não foi possível adicionar o livro!');
                    }

                    resolve();
                }
            )
        });
    }
```

Feito isso, no `rotas.js` , podemos definir o que será feito quando o livro for adicionado ao banco de dados. Ao final da inserção, queremos retornar para a página de listagem de livros. Como já definimos essa página anteriormente, iremos simplesmente redirecionar o usuário para uma rota já existente.

```
app.post('/livros', function(req, resp) {
    console.log(req.body);
    const livroDao = new LivroDao(db);
    livroDao.adiciona(req.body)
            .then()
            .catch(erro => console.log(erro));
});
```

Para isso, no método `then()`, utilizaremos a resposta (`resp`) para invocar o método `redirect()`, que recebe como parâmetro uma *string* identificando a rota que queremso acionar - nesse caso, `/livros`.

```
app.post('/livros', function(req, resp) {
    console.log(req.body);
    const livroDao = new LivroDao(db);
    livroDao.adiciona(req.body)
            .then(resp.redirect('/livros'))
            .catch(erro => console.log(erro));
});
```

Feito isso, poderemos acessar a URL http://localhost:3000/livros/form (http://localhost:3000/livros/form) para testarmos essa nova funcionalidade. Adicionando quaisquer valores para os campos "Título", "Preço" e "Descrição", e em seguida clicando em "Salvar", seremos redirecionados para a página http://localhost:3000/livros (http://localhost:3000/livros). Nela, o novo livro terá sido cadastrado.

Nosso próximo passo será criar as duas últimas funcionalidades da nossa aplicação: edição e deleção dos dados.