Desenvolvimento de um banco de dados para automação residencial utilizando *MySQL* e *Sequelize* em *javascript*

Leonardo Reginato

Resumo: A automação residencial já não é mais algo do futuro. Cada vez mais populares, as tecnologias utilizadas dentro de casa estão transformando em realidade o que antes era apenas ficção. Automação residencial diz respeito a um conjunto de tecnologias com a capacidade de programar eventos em uma casa e de tornar automático o funcionamento de diversos equipamentos. Por isso, este trabalho descreve o desenvolvimento de um banco de dados para automação residencial utilizando as tecnologias *MySQL* e *Sequelize* na linguagem de programação *javascript*.

Introdução: O projeto tem como objetivo, utilizando o banco de dados modelado e implementado na parte inicial da disciplina, de realizar a implementação de um sistema de automação utilizando a linguagem de programação orientada a objetos *javascript* e um ORM (*Object-Relational Mapping*) chamado *Sequelize*. A interface com o sistema é realizada através do terminal do software *Visual Studio Code* (*VS Code*). Para o código, foi utilizado também a tecnologia *node.js* pois esta possui bibliotecas prontas que facilitam na hora do desenvolvimento.

Sequelize: Como a linguagem de programação escolhida para este trabalho foi o javascript, a primeira etapa foi encontrar um ORM que se adequasse a ela. Desta forma foi possível encontrar e utilizar A primeira parte do código é realizar a importação da imagem objeto de teste no programa e que é utilizada durante todo o processamento. Essa imagem recebe então um tratamento de cor em escala de cinza e também um *blur* gaussiano. Assim os próximos passos do processamento podem ser executados.Desta forma foi possível encontrar e utilizar o *Sequelize*.
Sequelize é um *ORM* para *node.js* baseado em *promise*, utilizado para diversos bancos de dados e também *MySQL*. Como *ORM*, o *sequelize* faz mapeamento de dados relacionais (tabelas, colunas e linhas) para objetos *JavaScript*. Permite criar, buscar, atualizar e remover dados do banco utilizando métodos JS, permite modificar a estrutura de tabelas, facilitando na criação, população e migração do banco de dados.

SGBD MySQL: Como na primeira parte do projeto (Grau A) da disciplina o banco de dados *MySQL* foi escolhido para ser utilizado, foi possível também utiliza-lo pois o *ORM sequelize* tem ligação com o banco/ferramenta. O *MySQL* é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem *SQL* como interface.

Diagrama ER do banco criado: Para que seja possível iniciar a criação do código *ORM* utilizando o *sequelize* em *node.js*, foi necessário criar novamente o diagrama ER do banco. Primeiramente foram definidas as entidades do projeto. Neste caso são 3: ***equipamentos***, ***cenas*** e ***propriedades***. Ao mesmo tempo foi definido o relacionamento ***cenaEquipamento***.
O diagrama mostra ainda as relações entre as entidades sendo de *muitos-para-muitos* e de *um-para-muitos*. Cada tabela contém suas chaves primárias e seus atributos. Uma tabela adicional foi criada a partir do relacionamento entre as entidades equipamentos e cenas.

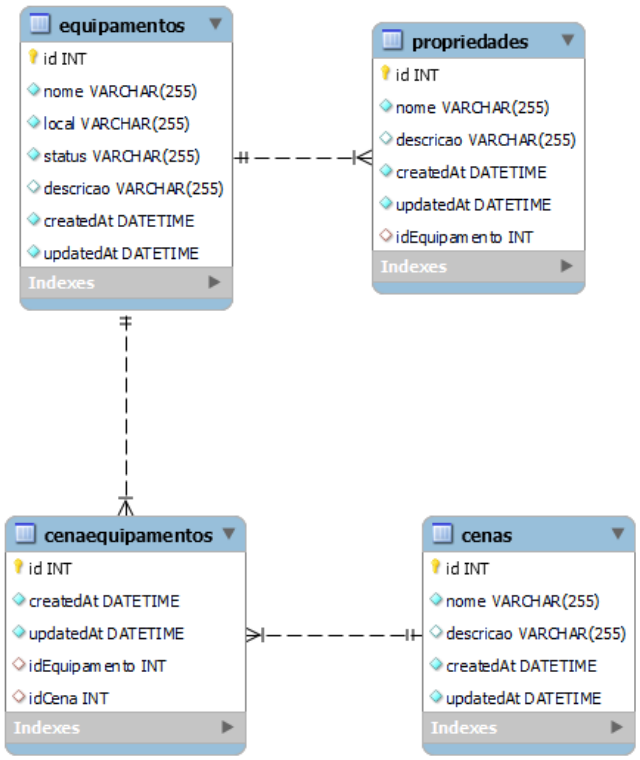

Fig. 1 *Diagrama ER criado para o desenvolvimento do código e relacionamento com o banco.*

Desenvolvimento do código: Como já dito anteriormente, para o desenvolvimento do código foi utilizada a ferramenta *VS Code* e a linguagem de programação *javascript* com *node.js*. Para ligação do banco de dados *MySQL* e da aplicação desenvolvida, foi utilizado o *ORM Sequelize*.
Primeiro devemos carregar a dependência do *Sequelize*, inicializar um novo objeto usando o construtor dele que espera o nome do banco de dados, do usuário, da senha e de um objeto com opções para a conexão.

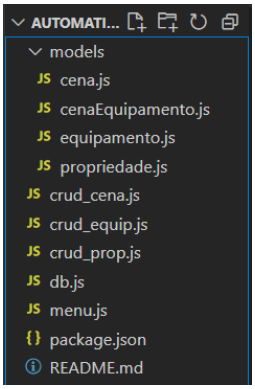

Fig. 2 *Estrutura dos arquivos do programa desenvolvido.*

A figura 2 mostra a estrutura e os arquivos *javascript* criados para a aplicação. As três entidades do banco de dados foram criadas utilizando o conceito de '*models*', assim como para o relacionamento/tabela. Esses arquivos contém as informações do mapeamento objeto-relacional, ou seja, o código *JavaScript* que representa uma tabela do banco de dados.

Desenvolvimento do CRUD: O próximo passo foi utilizar os módulos criados na aplicação. Como mencionado, o *Sequelize* faz a gestão não apenas das conexões com o banco, mas também a criação das tabelas necessárias, se elas ainda não existirem.
Para cada entidade foi criado um arquivo *CRUD* (*create*, *read*, *update* e *delete*). Desta forma o programa principal consegue acessar cada função presente nestes respectivos arquivos.

Desenvolvimento da aplicação: O programa principal é um menu criado via terminal do VS Code. Esse menu permite ao usuário selecionar qual opção ele deseja executar. Estas opções são os CRUDS de cada entidade. Outra opção disponível é a de Executar a Cena selecionada, que seria a ação principal do projeto.

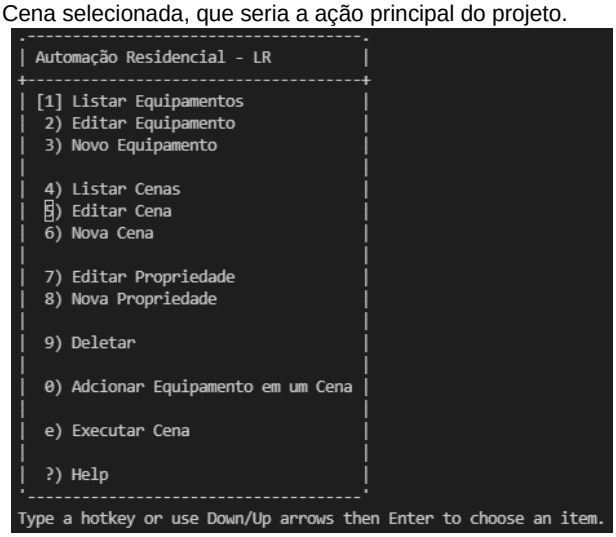

Fig. 3 *Menu criado via terminal para seleção das opções do programa.*

Resultados e Conclusões: Foi possível desenvolver um código capaz de realizar o relacionamento entre uma interface via terminal com o banco de dados criado. Todas as *queries* foram possíveis de serem executadas vias funções *CRUD* criadas para cada entidade do banco. Para fins de estudo este trabalho apresenta aplicações definidas e bem detalhadas. Caso uma futura melhoria fosse realizada, o código poderia ser melhorado para que as funções sejam executadas de forma mais efetiva levando em conta tempos de execução e custos.

Referências

1.**Sequelize documentation**. Disponível em: https://sequelize.org/. Acesso em: 10 nov. 2021.

2. **Node JS documentation**. Disponível em: https://nodejs.org/en/ Acesso em: 8 nov. 2021.