Instituto Federal Catarinense
Campus Sombrio
Curso Técnico em Informática para Internet

TRABALHO DE DESENVOLVIMENTO WEB

DESVENDANDO O MUNDO DOS ORMS COM SEQUELIZE

Luísa Pioner França

Turma: 3ª série B

Sombrio
Julho, 2025

Sumário

1. Introdução

Este trabalho busca explorar os conceitos e práticas do Mapeamento Objeto-Relacional (ORM), com ênfase no Sequelize, uma biblioteca amplamente utilizada no ecossistema Node.js. O objetivo principal é compreender o papel dos ORMs, identificar suas vantagens e limitações e aplicar esse conhecimento em cenários reais de desenvolvimento.

2. Fundamentos dos ORMs

2.1. O que é um ORM?

ORM (Object-Relational Mapping) é uma técnica que permite que linguagens orientadas a objetos, como JavaScript, interajam com bancos de dados relacionais. Em vez de escrever comandos SQL diretamente, o programador utiliza objetos e métodos da linguagem para manipular os dados.

Um dos principais problemas que os ORMs resolvem é o chamado Impedância Objeto-Relacional — a diferença entre a estrutura dos objetos em código e a forma como os dados são armazenados em tabelas relacionais. ORMs como o Sequelize atuam como uma ponte entre esses dois mundos.

2.2. Como um ORM funciona em alto nível?

O funcionamento básico de um ORM pode ser resumido assim:

1. O desenvolvedor escreve Usuario.findAll() no código.
2. O ORM interpreta esse comando.
3. Ele gera a SQL equivalente (SELECT * FROM Usuarios).
4. A consulta é enviada ao banco.
5. O banco retorna os dados.
6. O ORM converte os resultados em objetos.
7. O sistema os utiliza normalmente em JavaScript.

A camada de abstração criada pelo ORM oculta os detalhes do SQL, permitindo foco na lógica da aplicação.

3. Sequelize em Detalhes

3.1. O que é o Sequelize?

Sequelize é um ORM baseado em Promises, desenvolvido para uso com Node.js. Ele suporta múltiplos bancos de dados, como MySQL, PostgreSQL, SQLite, MariaDB e SQL Server. Sua popularidade deve-se à sua sintaxe simples, documentação acessível e capacidade de modelar relacionamentos complexos entre tabelas.

3.2. Models e associações

No Sequelize, um *model* representa uma tabela do banco. Exemplo de model:

```
const Produto = sequelize.define('Produto', {
  id: {
    type: Sequelize.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  nome: {
    type: Sequelize.STRING,
    allowNull: false
  },
  preco: {
    type: Sequelize.DECIMAL(10, 2),
    allowNull: false
  }
});
```

As principais associações no Sequelize são:

- hasOne: Um registro tem um relacionado (um-para-um).
- hasMany: Um registro tem muitos (um-para-muitos).
- belongsTo: Um registro pertence a outro (inverso do hasMany).
- belongsToMany: Muitos para muitos, com tabela intermediária.

Exemplo de belongsToMany:

```
Aluno.belongsToMany(Disciplina, {
  through: 'AlunoDisciplina',
  foreignKey: 'alunoID'
});
Disciplina.belongsToMany(Aluno, {
```

```
  through: 'AlunoDisciplina',
  foreignKey: 'disciplinaID'
});
```

3.3. Consultas e a Abstração do SQL

SQL	Sequelize
SELECT * FROM Usuarios;	Usuario.findAll();
SELECT * FROM Usuarios WHERE id = 1;	Usuario.findByPk(1);
SELECT * FROM Usuarios WHERE cidade = ‘Sombrio’;	Usuario.findAll({ where: { cidade: ‘Sombrio’ } });
JOIN com outra tabela	Usuarios.findAll({ include: Endereco });

4. Tópicos Avançados e Boas Práticas

4.1. Migrations

Migrations são scripts que versionam o banco de dados. Permitem aplicar mudanças estruturais como criação, alteração ou remoção de tabelas de forma controlada e segura.

Exemplo de fluxo com Sequelize CLI:

- Criar:
  npx sequelize-cli migration:generate --name criar-tabela-usuarios
- Aplicar:
  npx sequelize-cli db:migrate
- Reverter:
  npx sequelize-cli db:migrate:undo

O comando sequelize.sync({ force: true }) remove todas as tabelas e recria, sendo perigoso em produção por apagar dados.

4.2. Transações

Transações garantem que múltiplas operações sejam executadas como uma unidade. Se uma falhar, nenhuma será aplicada (conceito de atomicidade).

Exemplo:

```
const t = await sequelize.transaction();
try {
  await Usuario.update({ nome: 'João' }, { where: { id: 1 }, transaction: t });
  await Usuario.create({ nome: 'Maria' }, { transaction: t });
  await t.commit();
} catch (error) {
  await t.rollback();
}
```

5. Análise Crítica e Comparativa

5.1. Vantagens e Desvantagens de Usar um ORM

- Vantagens:

Produtividade: menos código e menos erros manuais.

Portabilidade: suporta múltiplos bancos sem reescrever SQL.

Organização: modelos centralizados facilitam a manutenção.

- Desvantagens:

Performance: pode ser menos eficiente em consultas complexas.

Curva de aprendizado: nem todos os conceitos são simples.

Abstração excessiva: o desenvolvedor pode perder controle do SQL.

5.2. Quando NÃO Usar um ORM

ORMs são inadequados em aplicações que exigem:

- Alto desempenho com otimizações específicas de SQL.
- Controle granular de consultas muito complexas.
- Tamanhos de banco muito grandes onde a abstração gera lentidão

5.3. Comparativo: Sequelize vs. Knex.js

Critério	Sequelize	Knex.js
Tipo	ORM	Query Builder
Linguagem	JavaScript	JavaScript
Definição de schema	Models (objetos)	Métodos encadeados
Facilidade de uso	Alta (abstração completa)	Média (exige mais SQL)

6. Conclusão

O Sequelize é uma poderosa ferramenta que simplifica o uso de bancos de dados relacionais com Node.js. Com ele, é possível aumentar a produtividade, manter o código organizado e evitar erros comuns ao manipular SQL diretamente. No entanto, para projetos com alta demanda de performance e consultas otimizadas, é importante avaliar se um ORM é realmente a melhor escolha.

7. Referências Bibliográficas

SEQUELIZE. Documentação Oficial. Disponível em: https://sequelize.org/docs/v6/. Acesso em: 15 jul. 2025.

WRDEV. Curso de Sequelize ORM com Node.js. YouTube, 2021. Disponível em: https://www.youtube.com/watch?v=g5ij7NIPR2s. Acesso em: 15 jul. 2025.

REDDIT. Does Sequelize have any downsides I should know? Reddit, 2019. Disponível em: https://www.reddit.com/r/node/comments/bfip13/does_sequelize_have_any_downsides_i_should_know/. Acesso em: 15 jul. 2025.

KNEX.JS. Documentação Oficial. Disponível em: https://knexjs.org/. Acesso em: 15 jul. 2025.