Sequelize

ORM

- Abstração do banco de dados;
- Tabelas viram models;

Manipulação dos dados

- Sem SQL (na maioria das vezes);
- Apenas código JavaScript;

```sql
INSERT INTO users (name, email)
     VALUES (
       "Diego Fernandes",
       "diego@rocketseat.com.br"
     )
```

```javascript
User.create({
  name: 'Diego Fernandes',
  email: 'diego@rocketseat.com.br',
})
```

```sql
SELECT *
  FROM users
 WHERE email = "diego@rocketseat.com.br"
 LIMIT 1
```

```javascript
User.findOne({
  where: {
    email: 'diego@rocketseat.com.br'
  }
})
```

Migrations

- Controle de versão para base de dados;
- Cada arquivo contém instruções para criação, alteração ou remoção de tabelas ou colunas;
- Mantém a base atualizada entre todos desenvolvedores do time e também no ambiente de produção;
- Cada arquivo é uma migration e sua ordenação ocorre por data;

```
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('users', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      name: {
        allowNull: false,
        type: Sequelize.STRING
      },
      email: {
        allowNull: false,
        unique: true,
        type: Sequelize.STRING
      }
    })
  },

  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('users')
  }
}
```

Instrução para criar uma nova tabela

Criação de 3 campos com suas propriedades.

O ID é a chave **primária** e auto **incremental**.

Instrução para deletar a tabela caso haja um rollback

- É possível desfazer uma migração se errarmos algo enquanto estivermos desenvolvendo a feature;
- Depois que a migration foi enviada para outros devs ou para ambiente de produção ela JAMAIS poderá ser alterada, uma nova deve ser criada;
- Cada migration deve realizar alterações em apenas uma tabela, você pode criar várias migrations para alterações maiores;

Seeds

- População da base de dados para desenvolvimento;
- Muito utilizado para popular dados para testes;
- Executável apenas por código;
- Jamais será utilizado em produção;
- Caso sejam dados que precisam ir para produção, a própria migration pode manipular dados das tabelas;

Arquitetura MVC

Model

O model armazena a abstração do banco, utilizado para manipular os dados contidos nas tabelas do banco. Não possuem responsabilidade sobre a regra de negócio da nossa aplicação.

Controller

O controller é o ponto de entrada das requisições da nossa aplicação, uma rota geralmente está associada diretamente com um método do controller. Podemos incluir a grande parte das regras de negócio da aplicação nos controllers (conforme a aplicação cresce podemos isolar as regras).

View

A view é o retorno ao cliente, em aplicações que não utilizando o modelo de API REST isso pode ser um HTML, mas no nosso caso a view é apenas nosso **JSON** que será retornado ao front-end e depois manipulado pelo **ReactJS** ou **React Native**.

A face de um controller

- Classes;
- Sempre retorna um JSON;
- Não chama outro controller/ método;
- Quando criar um novo controller:
  - Apenas 5 métodos;
  - Estou falando da mesma entidade?

```
class UserController {

  index() { } // Listagem de usuários

  show() { } // Exibir um único usuário

  store() { } // Cadastrar usuário

  update() { } // Alterar usuário

  delete() { } // Remover usuário

}
```

BORA CODAR!