NodeJS

Dominando o JavaScript no Backend

DEV SAMURAI

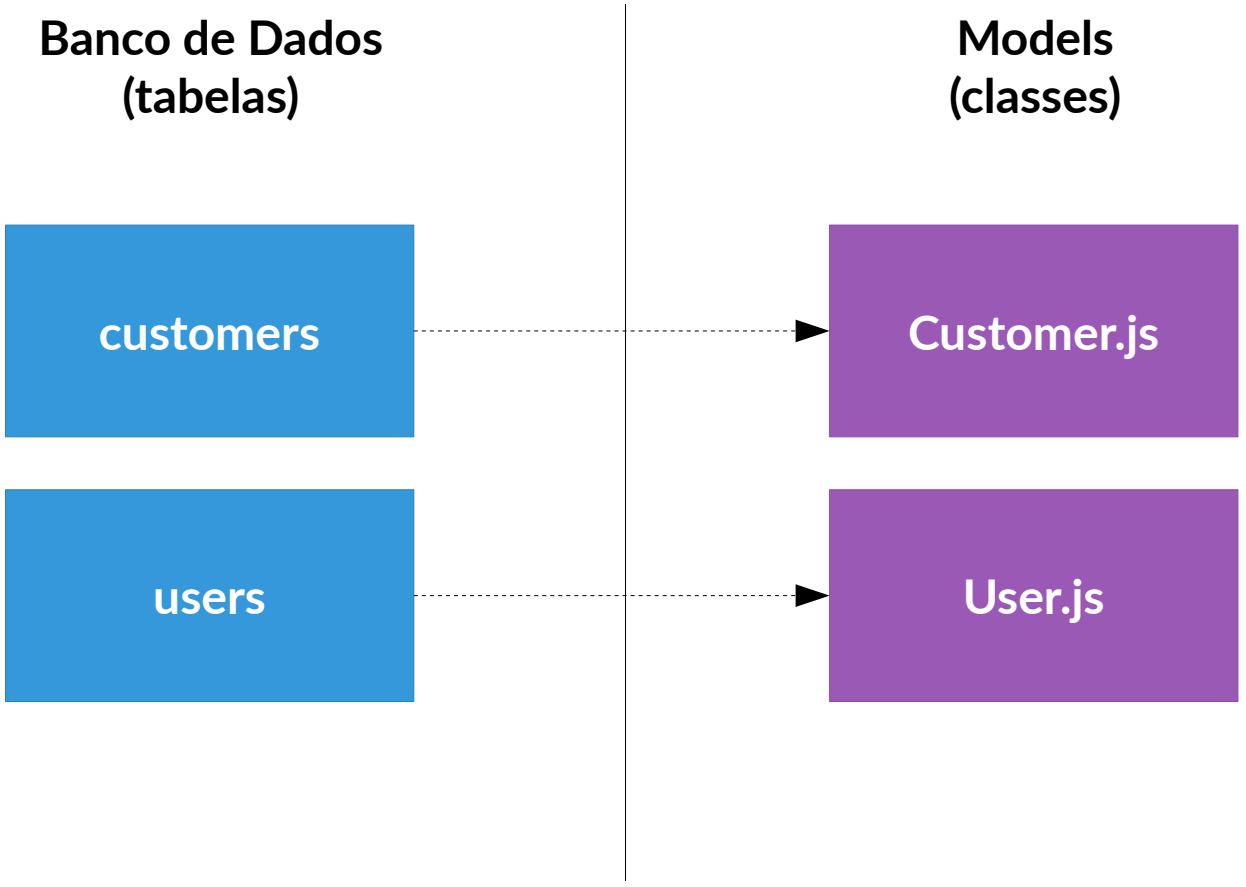
ORM

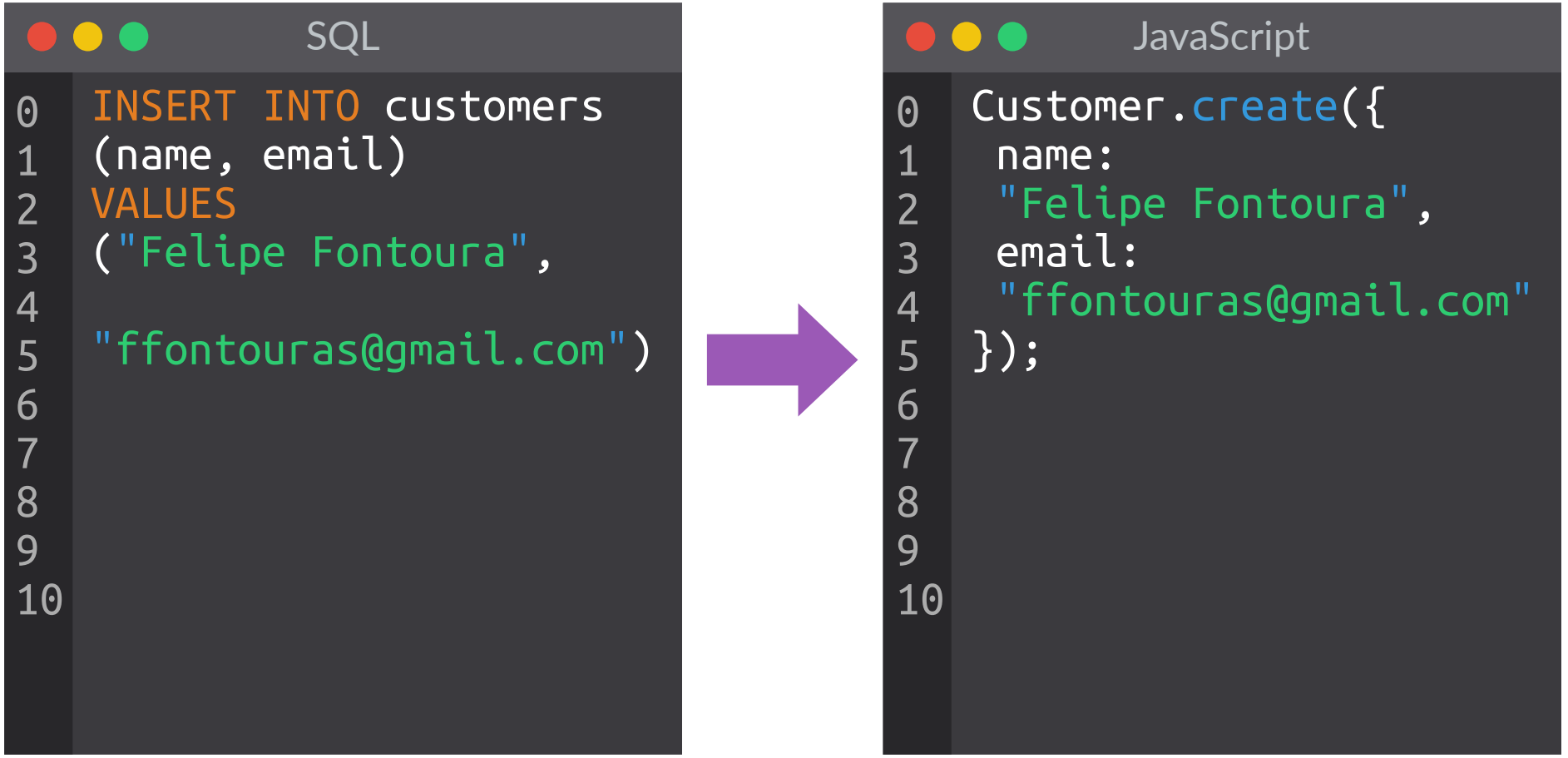
ORM

(*Object Relational Mapper*)

Abstração do Banco de Dados e Tabelas tornam-se **Classes**.

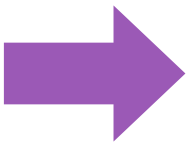
Sequelize


Banco de Dados
(tabelas)
Models
(classes)
customers
users
Customer.js
User.js


SQL
INSERT INTO customers
(name, email)
VALUES
("Felipe Fontoura",
"ffontouras@gmail.com")
JavaScript
Customer.create({
name:
"Felipe Fontoura",
email:
"ffontouras@gmail.com"
});

SQL

```sql
SELECT *
FROM customers
WHERE email =
"ffontouras@gmail.com"
LIMIT 1
```

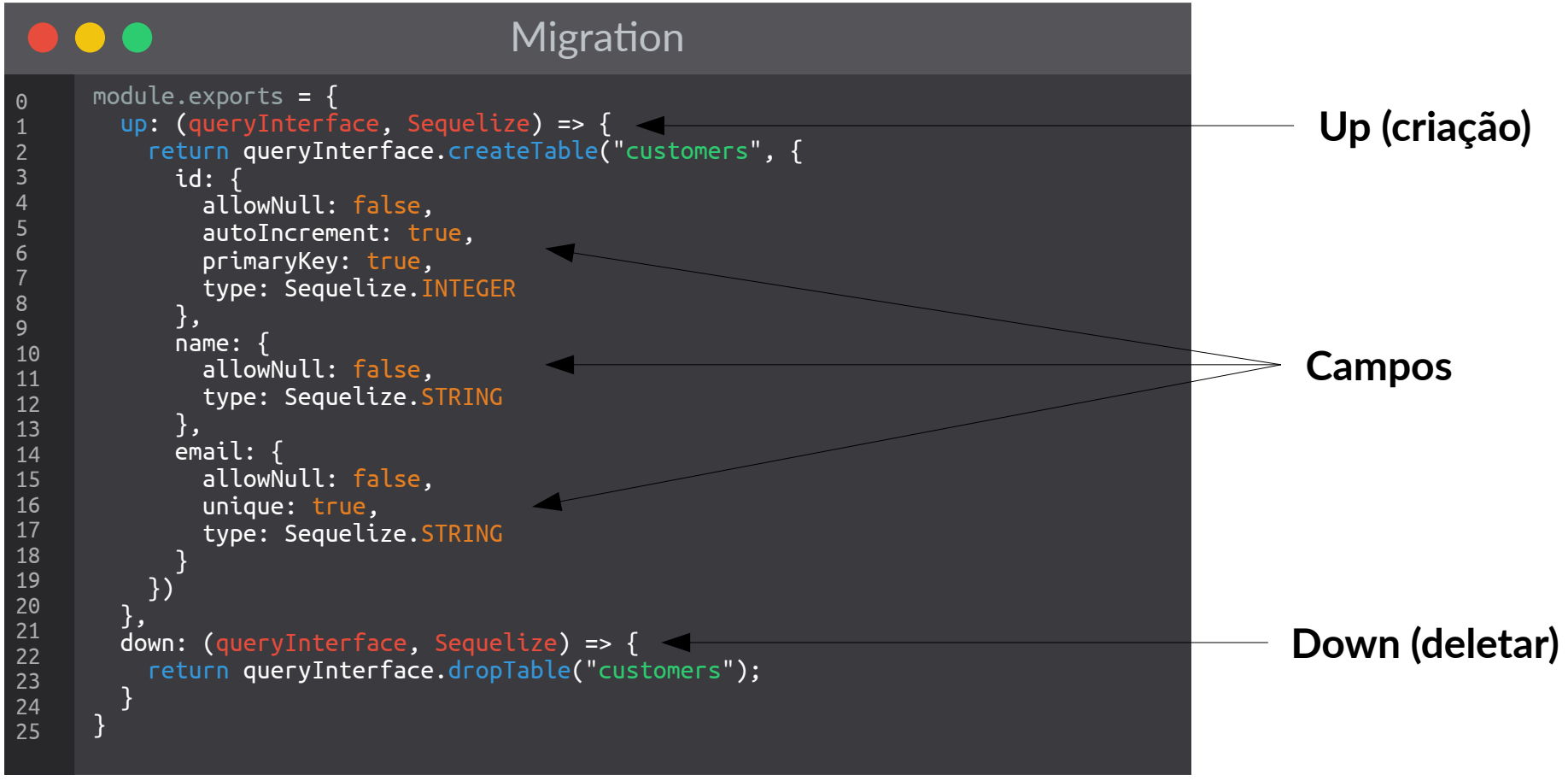


JavaScript

```javascript
Customer.findOne({
 where: {
 email:
 "ffontouras@gmail.com"
 }
});
```

Migrations

- Controlas as versões das tabelas/banco de dados.
- Cada migration contem as instruções de criação, alteração e remoção de tabelas e colunas.
- Permite manter a base sempre atualizada (desenvolvimento e produção).
- Uma migration por arquivo e ordenada por data/hora.


Migration
module.exports = {
up: (queryInterface, Sequelize) => {
return queryInterface.createTable("customers", {
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("customers");
}
}
Up (criação)
Campos
Down (deletar)

- Toda migration pode ser desfeita.
- Depois de uma migration é enviada (git ou produção) nunca mais pode ser desfeita.
- Cada migration manipula **uma tabela**.