CRUD SEQUELIZE

CREATE: <https://sequelize.org/docs/v6/core-concepts/model-querying-basics/>

const jane = await User.create({

firstName: "Jane",

lastName: "Doe"

});

READ: <https://sequelize.org/docs/v6/core-concepts/model-querying-basics/>

https://sequelize.org/docs/v6/core-concepts/model-querying-finders/

findAll():

const users = await User.findAll();

Indicar atributos/columnas a devolver:

Model.findAll({  
 attributes: ['foo', 'bar']  
});

BUSCAR POR LLAVE PRIMARIA

const project = await Project.findByPk(123);  
if (project === null) {  
 console.log('Not found!');  
} else {  
 console.log(project instanceof Project); *// true*  
 *// Its primary key is 123*  
}

ESCONTRAR UNO:

const project = await Project.findOne({ where: { title: 'My Title' } });  
if (project === null) {  
 console.log('Not found!');  
} else {  
 console.log(project instanceof Project); *// true*  
 console.log(project.title); *// 'My Title'*  
}

FIND OR CREATE:

const [user, created] = await User.findOrCreate({  
 where: { username: 'sdepold' },  
 defaults: {  
 job: 'Technical Lead JavaScript'  
 }  
});  
console.log(user.username); *// 'sdepold'*  
console.log(user.job); *// This may or may not be 'Technical Lead JavaScript'*  
console.log(created); *// The boolean indicating whether this instance was just created*  
if (created) {  
 console.log(user.job); *// This will certainly be 'Technical Lead JavaScript'*  
}

USO DE AGREGACIONES

Model.findAll({  
 attributes: [  
 'foo',  
 [sequelize.fn('COUNT', sequelize.col('hats')), 'n\_hats'],  
 'bar'  
 ]  
});

SELECT foo, COUNT(hats) AS n\_hats, bar FROM ...

//EXCLUIR ATRIBUTOS:

Model.findAll({  
 attributes: { exclude: ['baz'] }  
});

WHERE:

Post.findAll({  
 where: {  
 authorId: 2  
 }  
});  
*// SELECT \* FROM post WHERE authorId = 2;*

WHERE CON OP:

const { Op } = require("sequelize");  
Post.findAll({  
 where: {  
 authorId: {  
 [Op.eq]: 2  
 }  
 }  
});  
*// SELECT \* FROM post WHERE authorId = 2;*

WHERE CON OR:

const { Op } = require("sequelize");  
Post.findAll({  
 where: {  
 [Op.or]: [  
 { authorId: 12 },  
 { authorId: 13 }  
 ]  
 }  
});

VER TODOS LOS OPERADORES:

<https://sequelize.org/docs/v6/core-concepts/model-querying-basics/#operators>

UPDATE: https://sequelize.org/docs/v6/core-concepts/model-querying-basics/#simple-update-queries

await User.update({ lastName: "Doe" }, {  
 where: {  
 lastName: null  
 }  
});

DELETE / DESTROY:

<https://sequelize.org/docs/v6/core-concepts/model-querying-basics/#simple-delete-queries>

await User.destroy({  
 where: {  
 firstName: "Jane"  
 }  
});

ORDER Y GROUP BY

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/#ordering-and-grouping

COUNT:

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/#ordering-and-grouping

console.log(`There are ${await Project.count()} projects`);  
  
const amount = await Project.count({  
 where: {  
 id: {  
 [Op.gt]: 25  
 }  
 }  
});  
console.log(`There are ${amount} projects with an id greater than 25`);

MAX, MIN, SUM:

<https://sequelize.org/docs/v6/core-concepts/model-querying-basics/#max-min-and-sum>

await User.max('age'); *// 40*  
await User.max('age', { where: { age: { [Op.lt]: 20 } } }); *// 10*  
await User.min('age'); *// 5*  
await User.min('age', { where: { age: { [Op.gt]: 5 } } }); *// 10*  
await User.sum('age'); *// 55*  
await User.sum('age', { where: { age: { [Op.gt]: 5 } } }); *// 50*

INCREMENT / DECREMENT: https://sequelize.org/docs/v6/core-concepts/model-querying-basics/#increment-decrement

await User.max('age'); *// 40*  
await User.max('age', { where: { age: { [Op.lt]: 20 } } }); *// 10*  
await User.min('age'); *// 5*  
await User.min('age', { where: { age: { [Op.gt]: 5 } } }); *// 10*  
await User.sum('age'); *// 55*  
await User.sum('age', { where: { age: { [Op.gt]: 5 } } }); *// 50*