

Models Sequelize relacions i consultes

Hugo Córdoba, Nil Díaz



INDEX:

Models:	3
Actuacio.js	3
Departament.js	4
Incidencia.js	5
Tecnic.js	6
Tipu.js	7
Relacions:	7
Departament-Incidència	8
Tècnic-Incidència	8
Incidència-Actuació	8
Tècnic-Actuació	9
Tipu-Incidència	9
Consultes	10
Llistat d'incidències pendents i resoltes	10
Llistat d'incidències per tècnic	11

Models:

Actuacio.js

```
// src/models/Actuacio.js

const { DataTypes } = require("sequelize");
const sequelize = require("../db");

const Actuacio = sequelize.define("Actuacio", {
  id: {
    type: DataTypes.INTEGER,
    unique: true,
    primaryKey: true,
    autoIncrement: true,
  },
  temps: {
    type: DataTypes.STRING,
    allowNull: true
  },
  id_incidencia: {
    type: DataTypes.INTEGER,
    allowNull: false,
  },
  id_tecnic: {
    type: DataTypes.INTEGER,
    allowNull: true,
  },
  descripcio_actuacio: {
    type: DataTypes.STRING,
    allowNull: false,
  },
});

module.exports = Actuacio;
```

Departament.js

```
// src/models/Departament.js
const { DataTypes } = require('sequelize');
const sequelize = require('../db');

const Departament = sequelize.define('Departament', {
  id: {
    type: DataTypes.INTEGER,
    unique: true,
    primaryKey: true,
    autoIncrement: true,
  },
  nom_dpt: {
    type: DataTypes.STRING,
    allowNull: false,
  },
});

module.exports = Departament;
```

Incidencia.js

```
// src/models/Incidencies.js
const { DataTypes } = require('sequelize');
const sequelize = require('../db');

const Incidencia = sequelize.define('Incidencia', {
  id: {
    type: DataTypes.INTEGER,
    unique: true,
    primaryKey: true,
    autoIncrement: true,
  },

  id_departament: {
    type: DataTypes.INTEGER,
    allowNull: true,
  },

  id_tecnic: {
    type: DataTypes.INTEGER,
    allowNull: true,
  },

  id_tipus: {
    type: DataTypes.INTEGER,
    allowNull: true,
  },

  descripcio: {
    type: DataTypes.STRING,
    allowNull: false,
    unique: false,
  },

  prioritat: {
    type: DataTypes.ENUM('Baixa', 'Mitjana', 'Alta'),
    allowNull: true,
  },

  estat: {
    type: DataTypes.ENUM('Resolt', 'No resolt'),
    allowNull: false,
    defaultValue: 'No resolt'
  }
});

module.exports = Incidencia;
```

Tecnic.js

```
// src/models/Tecnic.js
const { DataTypes } = require('sequelize');
const sequelize = require('../db');

const Tecnic = sequelize.define('Tecnic', {
  id: {
    type: DataTypes.INTEGER,
    unique: true,
    primaryKey: true,
    autoIncrement: true,
  },

  nom: {
    type: DataTypes.STRING,
    allowNull: false,
    unique: false,
  },
});

module.exports = Tecnic;
```

Tipu.js

```
// src/models/Tipus.js
const { DataTypes } = require('sequelize');
const sequelize = require('../db');

const Tipu = sequelize.define('Tipu', {
  id: {
    type: DataTypes.INTEGER,
    unique: true,
    primaryKey: true,
    autoIncrement: true,
  },
  nom_tipus: {
    type: DataTypes.STRING,
    allowNull: false,
  },
});

module.exports = Tipu;
```

Relacions:

Aquesta és la part del nostre codi en la que ens centrem en les relacions entre els models creats amb sequelize. És important tenir en compte que no totes les relacions són iguals perquè cada una està destinada a un objectiu.

```
// Relacions
Departament.hasMany(Incidencia, { foreignKey: 'id_departament', onDelete: 'SET NULL', onUpdate: 'CASCADE' });
Incidencia.belongsTo(Departament, { foreignKey: 'id_departament', onUpdate: 'CASCADE' });

Tecnic.hasMany(Incidencia, { foreignKey: 'id_tecnic', onDelete: 'SET NULL', onUpdate: 'CASCADE' });
Incidencia.belongsTo(Tecnic, { foreignKey: 'id_tecnic', onUpdate: 'CASCADE' });

Incidencia.hasMany(Actuacio, { foreignKey: 'id_incidencia', onDelete: 'CASCADE', onUpdate: 'CASCADE' });
Actuacio.belongsTo(Incidencia, { foreignKey: 'id_incidencia', onUpdate: 'CASCADE' });

Tecnic.hasMany(Actuacio, { foreignKey: 'id_tecnic', onDelete: 'SET NULL', onUpdate: 'CASCADE' });
Actuacio.belongsTo(Tecnic, { foreignKey: 'id_tecnic', onUpdate: 'CASCADE' });

Tipu.hasMany(Incidencia, { foreignKey: 'id_tipus', onDelete: 'SET NULL', onUpdate: 'CASCADE' });
Incidencia.belongsTo(Tipu, { foreignKey: 'id_tipus', onUpdate: 'CASCADE' });
```

Departament-Incidència

Primer de tot tenim la relació Departament-Incidència, tenim feta la relació per la id del departament que té una clau forana a Incidència que es diu `id_departament`.

En cas que esborri un departament, aquesta incidència que tingui assignada un departament que ha sigut esborrat, s'assignarà a "NULL" aquella id fins que no s'introdueixi un altre, en cas d'edició d'un departament d'una incidència, aquesta s'actualitzarà en cascada i el mateix en cas del departament.

Un departament pot estar assignat a moltes incidències però una incidència només pot tenir assignada un departament.

```
Departament.hasMany(Incidencia, { foreignKey: 'id_departament', onDelete: 'SET NULL', onUpdate: 'CASCADE' });
Incidencia.belongsTo(Departament, { foreignKey: 'id_departament', onUpdate: 'CASCADE' });
```

Tècnic-Incidència

En aquest cas la relació és bastant similar en el cas del Departament, la relació de clau forana està feta de la mateixa manera que la del departament, el model d'Incidència té una `id_tecnic` que és forana del model Tècnic.

La reacció del model en cas d'esborrar o actualitzar una incidència o un tècnic és la mateixa que la dels departaments, actua de la mateixa manera.

Un tècnic pot estar assignat a moltes incidències, però una incidència només pot tenir assignada un tècnic.

```
Tecnic.hasMany(Incidencia, { foreignKey: 'id_tecnic', onDelete: 'SET NULL', onUpdate: 'CASCADE' });
Incidencia.belongsTo(Tecnic, { foreignKey: 'id_tecnic', onUpdate: 'CASCADE' });
```

Incidència-Actuació

La relació d'aquesta segueix la mateixa metodologia que els anteriors, la relació amb la clau forana en aquest cas està localitzada al model Actuació el qual té un `id_incidencia` i després la incidència té el seu propi id.

El mètode d'esborrar o actualitzar alguna incidència o actuació actua de la mateixa manera, en format cascada, si una incidència és borra, esborren totes les actuacions que tinguin la clau forana de id aquella incidència.

Una incidència pot tenir moltes actuacions, però una actuació només pot estar relacionada amb una incidència.

```
Incidencia.hasMany(Actuacio, { foreignKey: 'id_incidencia', onDelete: 'CASCADE', onUpdate: 'CASCADE' });
Actuacio.belongsTo(Incidencia, { foreignKey: 'id_incidencia', onUpdate: 'CASCADE' });
```


Tècnic-Actuació

La relació dels tècnics i les actuacions podem ubicar la clau forana al model de l'actuació amb una `id_tecnic` i després tenim la `id` del tècnic al model corresponent.

El mètode d'esborrar i actualitzar és diferent que el model anterior, tenim el cas que si un tècnic esborra l'actuació que tenia registrada no és borra sinó que el valor que tenia de `id_tecnic` s'assigna com a "NULL", hem decidit que era una bona manera de tenir-ho perquè creiem que les actuacions d'una incidència són importants mantenir-les encara que un tècnic pugui ser esborrat de la nostra base de dades. En el cas de l'actualització es fa en forma de cascada.

I per últim un tècnic pot tenir moltes actuacions, però una actuació només pot tenir assignat un tècnic.

```
Tecnic.hasMany(Actuacio, { foreignKey: 'id_tecnic', onDelete: 'SET NULL', onUpdate: 'CASCADE' });
Actuacio.belongsTo(Tecnic, { foreignKey: 'id_tecnic', onUpdate: 'CASCADE' });
```

Tipu-Incidència

La relació entre el tipus i les incidències és la següent, tenim una clau forana ubicada al model d'incidències que es diu `id_tipus` que la utilitzem per trobar el nom del tipus sobre la `id` del model de tipus.

Segueix la mateixa tipologia que l'anterior, en cas que un tipus sigui esborrat, aquest es posa en "NULL", perquè no pot esborrar-se una incidència si un Tipus és modificat o esborrat, per això actua de la mateixa manera que si s'edita, s'actualitza amb normalitat.

I no obstant tenim la metodologia, un tipus pot estar assignat a moltes incidències, però només la incidència pot tenir un tipus.

```
Tipu.hasMany(Incidencia, { foreignKey: 'id_tipus', onDelete: 'SET NULL', onUpdate: 'CASCADE' });
Incidencia.belongsTo(Tipu, { foreignKey: 'id_tipus', onUpdate: 'CASCADE' });
```

Consultes

Llistat d'incidències pendents i resoltes

```
// Llistar incidències (GET)
router.get('/', async (req, res) => {
  try {
    const incidències = await Incidencia.findAll({ include: [Tipu, Departament, Tecnic] });
    res.render('incidències/list', { incidències });
  }
  catch (error) {
    console.error('Error al recuperar les incidències:', error);
    res.status(500).send('Error al recuperar les incidències');
  }
});
```

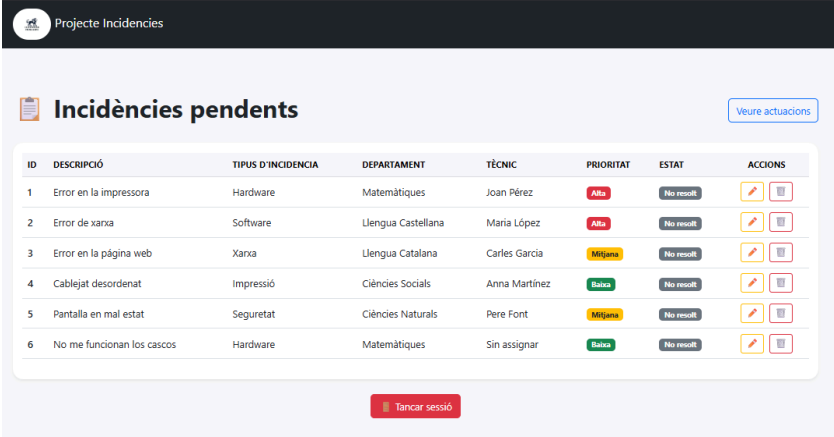
Aquest és el mètode que tenim per llistar les incidències, primer de tot declarem una constant la qual anirem guardant les incidències que ens vagi retornant, és important que incloem els Tipus i el Departament perquè és necessari per imprimir atributs d'aquest com són el departament i el tipus d'incidència.

El que fem és fer un render, que és una manera de dir-li al codi que li passem una constant on es poden guardar incidències i li donem al directori on les guardem que en aquest cas volem fer-ho en el list.ejs del directori incidències.













En aquest cas el filtratge per resoltes o no resoltes el tenim orientat al ejs, tenim un condicional posat en el ejs que comprova si l'estat és resolt, en cas de contrari, imprimeix la incidència amb els seus atributs.

```
<tbody>
  <% incidències.filter(incidència => incidència.estat === "No resolt").forEach(incidència => { %>
    <tr>
      <td class="fw-semibold"><%= incidència.id %></td>
      <td><%= incidència.descripció %></td>
      <td><%= incidència.Tipu.nom_tipus %></td>
      <td><%= incidència.Departament.nom_dpt %></td>
      <td><%= incidència.Tecnic && incidència.Tecnic.nom ? incidència.Tecnic.nom : 'Sin assignar' %></td>
    </tr>
  } %>
```

I aquí tenim l'exemple de com queda el llistat d'incidències des del nostre list.ejs:



The screenshot shows a web application interface for managing incidents. At the top, there's a header with a logo and the text "Projecte Incidències". Below this, there's a section titled "Incidències pendents" with a button "Veure actuacions". The main content is a table with 8 columns: ID, DESCRIPCIÓ, TIPUS D'INCIDÈNCIA, DEPARTAMENT, TÈCNIC, PRIORITAT, ESTAT, and ACCIONS. The table contains 6 rows of incident data. At the bottom, there's a red button labeled "Tancar sessió".

ID	DESCRIPCIÓ	TIPUS D'INCIDÈNCIA	DEPARTAMENT	TÈCNIC	PRIORITAT	ESTAT	ACCIONS
1	Error en la impressora	Hardware	Matemàtiques	Joan Pérez	Alta	No resol	 
2	Error de xarxa	Software	Llengua Castellana	Maria López	Alta	No resol	 
3	Error en la pàgina web	Xarxa	Llengua Catalana	Carles Garcia	Mitjana	No resol	 
4	Cablejat desordenat	Impressió	Ciències Socials	Anna Martínez	Baixa	No resol	 
5	Pantalla en mal estat	Seguretat	Ciències Naturals	Pere Font	Mitjana	No resol	 
6	No me funcionan los cascos	Hardware	Matemàtiques	Sin assignar	Baixa	No resol	 

Llistat d'incidències per tècnic

```
// GET /tecnicos
router.get('/', async (req, res) => {
  try {
    const tecnicos = await Tecnico.findAll({
      include: [
        {
          model: Incidencia,
          include: [Actuacio],
        }
      ]
    });
    res.render('tecnicos/list', { tecnicos });
  } catch (error) {
    console.error('Error al carregar tècnics:', error);
    res.status(500).send('Error al carregar tècnics');
  }
});
```

El llistat d'incidències per tècnic ho hem fet d'una manera peculiar, des de la ruta de tècnics hem fet un llistat de tots els tècnics passant també la incidència i les Actuacions, perquè el que hem realitzat és un llistat dels tècnics amb un desplegable cada un on es mostren les incidències que té assignat aquell tècnic i les actuacions.

En cas que aquell tècnic no tingui cap incidència assignada, li mostrarà un missatge a sota dient que no té cap assignada, però en cas que aquest sí que

tingui alguna assignada, però aquesta no té cap actuació, mostra un missatge confort dient que no té cap actuació assignada, que tot això es gestiona des del ejs.

```
<% if (incidencia.Actuacions && incidencia.Actuacions.length > 0) { %>
<h6 class="mt-3">Actuacions:</h6>
<ul>
  <% incidencia.Actuacions.forEach(actuacio => { %>
    <li>
      <div>
        <strong>ID:</strong> <%= actuacio.id %> |
        <strong>Descripció:</strong> <%= actuacio.descripcio_actuacio %> |
        <strong>Temps:</strong> <%= actuacio.temps %> min
      </div>
      <div class="mt-2">
        <a href="/actuacions/<%= actuacio.id %>/edit" class="btn btn-sm btn-outline-warning">✎ Editar</a>
        <form action="/actuacions/<%= actuacio.id %>/delete" method="POST" class="d-inline">
          <button type="submit" class="btn btn-sm btn-outline-danger" onclick="return confirm('Segur que vols eliminar aquesta actuació?')">
            🗑 Eliminar
          </button>
        </form>
      </div>
    </li>
  <% }) %>
</ul>
<% } else { %>
<p class="text-muted">No hi ha actuacions registrades per aquesta incidència.</p>
<% } %>
```

Així és com quedaria el la nostra pagina de llistat.

 Projecte Incidències



Tècnics i incidències



Joan Pérez	▼
Maria López	▼
Carles Garcia	▼
Anna Martínez	▼
Pere Font	▼
Laura Vidal	▼
Marc Soler	▼
Clara Roca	▼
Jordi Serra	▼
Marta Bosch	▼

Aquí mostrem l'exemple d'un tècnic amb incidència i actuacions.

Joan Pérez

ID Incidència 1 - Error en la impressora

Eliminar

Actuacions:

- ID: 1 | **Descripció:** Revisió inicial | **Temps:** 30 min

Editar Eliminar
- ID: 2 | **Descripció:** Resolució de l'error | **Temps:** 40 min

Editar Eliminar
- ID: 3 | **Descripció:** Verificació final | **Temps:** 60 min

Editar Eliminar

Aquí el cas que un tècnic no tingui cap incidència assignada.

Laura Vidal

Aquest tècnic no té incidències assignades.

Aquí el cas que un tècnic tingui una incidència assignada, però no tingui cap actuació registrada.

Joan Pérez

ID Incidència 1 - Error en la impressora

Eliminar

No hi ha actuacions registrades per aquesta incidència.