

DOCUMENTACIÓ

MODELS DE SEQUELIZE:

Els models que hem utilitzat per crear les taules en el adminer gràcies al sequelize estan en el zip.

SCRIPTS DE CÀRREGA DE DADES I DE CREACIÓ DE LA BD

Els nostres script que fan la creació de la BD són els js que estan dins de la carpeta models. A continuació explicarem el **incidencies.js** com està configurat i com surt en la nostra base de dades (Adminer).

incidencies.js

```
const { DataTypes } = require('sequelize');
const sequelize = require('../db');
const Departament = require('./departament');
const Estat = require('./estat');
const Tecnic = require('./tecnic');
const Actuacio = require('./actuacio');
const Prioritat = require('./prioritat');
const Tipus = require('./tipus');
```

Primer de tot importem tots els altres recursos en constants ja que no volem que els valor canviïn. Aquest recursos són les altres taules. Els importem per poder fer les relacions, més endavant ho expliquem.

```
const Incidencia = sequelize.define('Incidencia', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  nom: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  departament_id: {
    type: DataTypes.INTEGER,
    allowNull: false,
    references: {
      model: Departament,
      key: 'id',
    },
  },
  tipus_id: {
    type: DataTypes.INTEGER,
    allowNull: false,
    references: {
      model: Tipus,
      key: 'id',
    },
  },
  descripcio: {
    type: DataTypes.TEXT,
    allowNull: false,
  },
  datacreacio: {
    type: DataTypes.DATEONLY,
    defaultValue: DataTypes.NOW,
  },
  estat_id: {
    type: DataTypes.INTEGER,
    references: {
      model: Estat,
      key: 'id'
    },
    allowNull: true,
  },
  tecnic_id: {
    type: DataTypes.INTEGER,
    references: {
      model: Tecnic,
      key: 'id'
    },
  },
  prioritat_id: {
    type: DataTypes.INTEGER,
    references: {
      model: Prioritat,
      key: 'id'
    },
    allowNull: true,
  },
  dataresolucio: {
    type: DataTypes.DATEONLY,
    allowNull: true,
  },
}, {
  tableName: 'Incidencia',
  timestamps: false,
});
```

Tot aquest codi és la definició de la taula on sequelize s'encarrega de fer el **insert** a la nostra base de dades. Primer anomenem el camp i després el configurem si volem que sigui un text o sigui un data.

Quan es tracta d'una foreign key, fem un **references** on posem el model (taula) de referència i automàticament sequelize s'encarrega de fer els joins.

```
// Associations
Incidencia.belongsTo(Departament, {
  foreignKey: 'departament_id',
  as: 'departament',
});
Incidencia.belongsTo(Tipus, {
  foreignKey: 'tipus_id',
  as: 'tipus',
});
Incidencia.belongsTo(Estat, {
  foreignKey: 'estat_id',
  as: 'estat',
});
Incidencia.belongsTo(Tecnic, {
  foreignKey: 'tecnic_id',
  as: 'tecnic'
});
Incidencia.belongsTo(Prioritat, {
  foreignKey: 'prioritat_id',
  as: 'prioritat'
});
Incidencia.hasMany(Actuacio, {
  foreignKey: 'incidencia_id',
  as: 'actuacions'
});
Actuacio.belongsTo(Incidencia, {
  foreignKey: 'incidencia_id',
  as: 'incidencia'
});

module.exports = Incidencia;
```

Finalment, al final del nostre incidencies.js fem les relacions entre les diferents models (taules) que tenim, gràcies als **const**, es carrega correctament i troba les taules.

Les relacions podríem haver-les fet en un altre js i connectar-lo en l'app.js, però podria haber causat errors.

Gràcies a les relacions podem fer-ne consultes amb més facilitat.

departament.js

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

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

module.exports = Departament;
```

estat.js

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

const Estat = sequelize.define('Estat', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  nom: {
    type: DataTypes.STRING(255),
    allowNull: false,
    defaultValue: 'Pendent d\'assignar',
    unique: true,
  },
}, {
  tableName: 'Estat',
  timestamps: false,
});

module.exports = Estat;
```

prioritat.js

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

const Prioritat = sequelize.define('Prioritat', [
  {
    id: {
      type: DataTypes.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    nom: {
      type: DataTypes.STRING,
      allowNull: false,
      unique: true
    }
  },
  {
    timestamps: false
  }
]);

module.exports = Prioritat;
```

tecnic.js

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

const Tecnic = sequelize.define('Tecnic', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  nom: {
    type: DataTypes.STRING(255),
    allowNull: false,
  },
  cognoms: {
    type: DataTypes.STRING(255),
    allowNull: false,
  },
  email: {
    type: DataTypes.STRING(255),
    allowNull: false,
    unique: true,
    validate: {
      isEmail: true,
    },
  },
}, {
  tableName: 'Tecnic',
  timestamps: false,
});

module.exports = Tecnic;
```

CONSULTES PRINCIPALS

Abans de començar amb les consultes, la connexió dels .ejs amb els routers és gràcies al **app.js**, ja que en aquell fitxer escrivim quin fitxer amb quin fitxer s'ha de connectar.

```
const express = require('express');
const path = require('path');
const sequelize = require('./db');
const mongoose = require('./mongo');
const app = express();
const incidenciaRoutes = require('./routes/incidencies.routes');
const admin = require('./routes/admin.routes');
const tecnic = require('./routes/tecnic.routes');
const Estat = require('./models/estat');
const Departamento = require('./models/departament');
const Tecnico = require('./models/tecnic');
const Prioritat = require('./models/prioritat');
const Tipus = require('./models/tipus');
const Incidencia = require('./models/incidencia');
const Actuacio = require('./models/actuacio');
const Log = require('./models/logModel');
const logsRoutes = require('./routes/logs.routes');
```

Importem els arxius necessaris per a que funcioni: La càrrega de sequelize, la de Mongo, les rutes, els models i els logs

```
app.use('/incidencies', incidenciaRoutes);
app.use('/admin', admin);
app.use('/tecnic', tecnic);
```

App.use activa les constants escrites abans.

Per exemple: Si entro a **/admin**, el servidor farà servir el ./routes/admin.routes, i gràcies això s'activaran les URL amb /admin

Les nostres consultes les fem amb sequelize, com hem dit, gràcies al routes. A continuació explicarem algunes consultes bàsiques del nostre projecte.

Esta organitzat d'aquesta manera:

```
routes/
├── admin.routes.js
├── incidencies.routes.js
├── logs.routes.js
└── tecnic.routes.js
```

Admin.routes.js

Formulari d'edició

```
router.get('/:id/edit', async (req, res) => {
  try {
    const incidencia = await Incidencia.findByPk(req.params.id);
    if (!incidencia) {
      return res.status(404).send('Incidencia no trobada');
    }

    const departamentos = await Departamento.findAll();
    const estats = await Estat.findAll();
    const tecnicos = await Tecnic.findAll();
    const prioritats = await Prioritat.findAll();
    const tipus = await Tipus.findAll();

    res.render('admin/edit', {
      incidencia,
      departamentos,
      estats,
      tecnicos,
      prioritats,
      tipus
    });
  } catch (error) {
    console.error('Error al cargar la incidencia:', error);
    res.status(500).send('Error al cargar la incidencia ' + error);
  }
});
```

Aquest codi carrega el formulari d'edició d'una incidència al panell d'administració. Aquesta ruta GET agafa la **id** d'incidència amb la qual apareix en la URL, si no troba aquesta URL, envia el següent missatge "Incidencia no trobada". Seguidament, gràcies a les constats fan el select a la nostra base de dades i després renderitza la pàgina que tenim en la carpeta admin anomenada edit per mostrar aquelles dades

admin/edit.ejs

```
<form id="editForm" action="/admin/<%= incidencia.id %>/update" method="POST" novalidate>

  <div class="mb-3">
    <label for="nom" class="form-label fw-semibold text-secondary">Nom</label>
    <div class="input-group">
      <span class="input-group-text bg-secondary text-white"><i class="bi bi-card-text"></i></span>
      <input type="text" class="form-control border-secondary" id="nom" name="nom" value="<%= incidencia.nom %>" required>
    </div>
    <div class="invalid-feedback">El nom és obligatori.</div>
  </div>

  <div class="row">
    <div class="col-md-6 mb-3">
      <label for="departament_id" class="form-label fw-semibold text-secondary">Departament</label>
      <select class="form-select border-secondary" id="departament_id" name="departament_id" required>
        <option disabled value="">Selecioneu...</option>
        <%= departamentos.forEach(departamento => { %>
          <option value="<%= departamento.id %>" <%= departamento.id === incidencia.departament_id ? 'selected' : '' %>><%= departamento.nom %></option>
        <%= }) %>
      </select>
      <div class="invalid-feedback">Escriu un departament.</div>
    </div>

    <div class="col-md-6 mb-3">
      <label for="estat_id" class="form-label fw-semibold text-secondary">Estat</label>
      <select class="form-select border-secondary" id="estat_id" name="estat_id" required>
        <%= estats.forEach(estat => { %>
          <option value="<%= estat.id %>" <%= estat.id === incidencia.estat_id ? 'selected' : '' %>><%= estat.nom %></option>
        <%= }) %>
      </select>
      <div class="invalid-feedback">Escriu un estat.</div>
    </div>
  </div>
```

En l'edit.ejs, que està dins de la carpeta admin, creem un <form> on les dades les enviarà a /admin/<%= incidencia.id %>/update, el <%= incidencia.id %> posarà automàticament l'ID de l'incidència que s'està editant.

Fem ús de <%= %> per a que mostri en pantalla les dades de la base de dades. S'ha d'escriure exactament igual al nombre que tens en la base de dades, és a dir, si tu tens en la base de dades **noms** en la taula **departamento**, i després alhora de mostrar en pantalla poses <%=departamento.nom %> no mostrarà la dada guardada.

The screenshot shows a web browser at localhost:3000/admin/2/edit. The page title is 'PORTAL D'INCIDÈNCIES'. The navigation bar includes links for 'Inici', 'Crear incidències', 'Llistat d'incidències', 'Admin', 'Tècnic', 'Logs', and 'Estadístiques'. The main content area is titled 'Editar Incidència' and contains a form with the following fields:

- Nom:** A text input field containing 'No es pot accedir a la xarxa Wi-Fi'.
- Departament:** A dropdown menu with 'Departament d'informàtica' selected.
- Estat:** A dropdown menu with 'En procés' selected.
- Prioritat:** A dropdown menu with 'Mitjana' selected.
- Tècnic:** A dropdown menu with 'Harsh' selected.
- Tipus:** A dropdown menu with 'Xarxes' selected.
- Descripció:** A text area containing 'Cap dispositiu es connecta a la xarxa del segon pis.'
- Data Resolució:** A date input field with the format 'dd/mm/aaaa'.

At the bottom of the form is a button labeled 'Actualizar Incidència'. Below the form, there is a link that says 'Tornar a la llista d'incidències'.

Aixó seria el que es mostraria en pantalla, que es el formulari que està creat dins de l'edit.ejs.

Actualitzar incidència

```
router.post('/:id/update', async (req, res) => {
  try {
    const {
      nom,
      departament_id,
      tipus_id,
      descripcio,
      datacreacio,
      estat_id,
      dataresolucio,
      tecnic_id
    } = req.body;

    const incidencia = await Incidencia.findByPk(req.params.id);
    if (!incidencia) {
      return res.status(404).send('Incidencia no encontrada');
    }

    incidencia.nom = nom;
    incidencia.departament_id = departament_id;
    incidencia.tipus_id = tipus_id;
    incidencia.descripcio = descripcio;
    incidencia.datacreacio = datacreacio;
    incidencia.estat_id = estat_id;
    incidencia.dataresolucio = dataresolucio ? dataresolucio : null;
    incidencia.tecnic_id = tecnic_id || null;

    await incidencia.save();

    console.log('Incidencia actualizada:', incidencia);
    return res.redirect('/admin');
  } catch (error) {
    console.error('Error al actualizar la incidencia:', error);
    if (!res.headersSent) {
      return res.status(500).send('Error al actualizar la incidencia: ' + error);
    }
  }
});
```

Abans hem vist com agafa les taules de la nostra base de dades i després es mostrava en pantalla... pero que fa quan volem actualitzar aquesta informació?

Bé, aquest codi agafa l'informació guardada dins del <form> que l'usuari proporciona en el edit.ejs i bàsicament les torna a guardar (**actualitzat**) en la taula incidencia amb el **await incidencia.save()**. No oblidar-nos cap variable del form ja que així no dona cap error.

Per últim, si funciona, mostra un missatge en consola "Incidència creada", i et redirigeix a la pantalla /admin, sino, mostra en consola "Error al actualitzar la incidència" i mostra l'error.

Incidencies.routes.js

Crear incidencia

new.ejs

```
<%- include('../partials/header', { title: 'Crear Incidència' }) %>

<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>

<div class="container my-5" style="max-width: 600px;">
  <div class="card shadow-sm">
    <div class="card-header bg-dark text-white text-center fs-4 fw-bold">
      Crear Nova Incidència
    </div>
    <div class="card-body">
      <form action="/incidencies/create" method="POST" class="needs-validation" novalidate aut>
        <div class="mb-4">
          <label for="nom" class="form-label fw-semibold">Nom de la incidència</label>
          <input
            type="text"
            class="form-control"
            id="nom"
            name="nom"
            placeholder="Ex: Impressora no funciona"
            required
            maxlength="150"
          />
          <div class="invalid-feedback">El nom és obligatori i ha de tenir menys de 150 caràcters</div>
        </div>

        <div class="mb-4">
          <label for="departament_id" class="form-label fw-semibold">Departament</label>
          <select class="form-select" id="departament_id" name="departament_id" required>
            <option value="" disabled selected>Selecciona un departament</option>
            <% departamentos.forEach(departamento => { %>
              <option value="<%= departamento.id %>"><%= departamento.nom %></option>
            } %>
          </select>
        </div>
      </form>
    </div>
  </div>
</div>
```

```
<script>
  (() => {
    'use strict';
    const forms = document.querySelectorAll('.needs-validation');
    Array.from(forms).forEach(form => {
      form.addEventListener('submit', e => {
        if (!form.checkValidity()) {
          e.preventDefault();
          e.stopPropagation();
        }
        form.classList.add('was-validated');
      }, false);
    });
  })();
</script>
```

```
<% if (successMessage && incidenciaId) { %>
<script>
  Swal.fire({
    icon: 'success',
    title: 'Incidència creada amb èxit',
    html:
      '<p><strong><%= successMessage %></strong></p>' +
      '<p>El teu ID d'incidència és: <span class="fw-bold"><%= incidenciaId %></span></p>',
    confirmButtonText: 'D'acord',
    confirmButtonColor: '#343a40',
  }).then((result) => {
    if (result.isConfirmed) {
      window.location.href = '/';
    }
  });
</script>
<% } %>
```

Gràcies aquest codi, mostrarà per pantalla en **/incidencies/new** un formulari per poder crear incidència.

Comença carregant el bootstrap per si el header no el carrega bé i a sota càrrega l'script de l'alerta que surt quan es crea l'incidència.

Seguidament, posem totes les variables que volem saber sobre el problema de l'usuari, el departament, quin tipus d'incidència és i una breu descripció.

Per últim, l'script què hem posat és per a que validi el formulari, és a dir, que sigui vàlid o que no hi hagi espais en blanc. A més, ahora de crear **correctament** una incidència, amb **sweetalert** mostra una alerta notificant-nos que s'ha creat correctament i l'id d'incidència.

```
router.post('/create', async (req, res) => {
  try {
    const { nom, departament_id, tipus_id, descripcio } = req.body;

    const dataCreacio = moment().tz('Europe/Madrid').toDate();
    const estat_id = 1;
    const dataResolucio = null;
    const tecnic_id = null;

    const incidencia = await Incidencia.create({
      nom,
      departament_id,
      tipus_id,
      descripcio,
      datacreacio: dataCreacio,
      estat_id,
      dataresolucio: dataResolucio,
      tecnic_id
    });

    console.log('Incidencia creada:', incidencia);

    const departamentos = await Departamento.findAll();
    const tipus = await Tipus.findAll();

    res.render('incidencies/new', {
      successMessage: 'Incidència creada correctament!',
      incidenciaId: incidencia.id,
      departamentos,
      tipus
    });
  } catch (error) {
    console.error('Error al crear la incidencia:', error);
    res.status(500).send("No s'ha pogut crear l'incidència: " + error);
  }
});
```

Gràcies a aquest codi, podem fer una nova incidència i que la emmagatzemi dins de la base de dades. Per començar, agafa l'informació del formulari ejs i també inicialitza les altres dades per completar l'incidència. Seguidament, crea l'incidència en la base de dades amb **await incidencia.create** agafant totes les variables.

Per últim, carrega les taules de departament i tipus per a que no doni error en les FK. Finalment si es crea correctament, carrega un script que es una alerta dient "Incidència creada correctament", si no es crea correctament, donarà l'error per pantalla.

Així es mostra en pantalla el formulari que hem creat en el **new.ejs**.

The screenshot shows a web application interface for creating a new incident. At the top, there is a dark navigation bar with the title 'PORTAL D'INCIDÈNCIES' and several menu items: 'Inici', 'Crear incidències', 'Llistat d'incidències', 'Admin', 'Tècnic', 'Logs', and 'Estadístiques'. Below this, the main content area features a form titled 'Crear Nova Incidència'. The form has four main sections: 'Nom de la incidència' with a text input field containing 'Ex: Impressora no funciona'; 'Departament' with a dropdown menu showing 'Selecciona un departament'; 'Tipus d'incidència' with a dropdown menu showing 'Selecciona un tipus'; and 'Descripció' with a large text area containing 'Descripció detallada del problema'. At the bottom of the form is a dark button labeled 'Crear Incidència'.

Alerta de notificació amb id mostrat

The screenshot shows a success notification alert. It features a large green checkmark icon at the top. Below the icon, the text reads: 'Incidència creada amb èxit', 'Incidència creada correctament!', and 'El teu ID d'incidència és: 7'. At the bottom of the alert is a dark button labeled 'D'acord'.

Consultar estat d'incidència

```
<form action="/incidencies/buscar" method="post" class="row g-3 mb-3">
  <div class="col-auto">
    <input type="text" name="id" class="form-control" placeholder="ID de l'incidència" />
  </div>
  <div class="col-auto">
    <button type="submit" class="btn btn-dark">Cerca</button>
  </div>
</form>
```

En l'ejs principal creem un formulari on l'acció la farà el /buscar en el **incidencies.routes.js**.

És un formulari molt simple on només posem un input **text** amb un botó per fer la recerca

Si s'ha trobat l'incidència, mostrarà l'id, el departament, la descripció, l'estat d'aquella incidència i les actuacions visibles.

```
<% if (incidencia) { %>
  <div class="card shadow-sm">
    <div class="card-body">
      <h5 class="card-title">Resultat de la cerca</h5>
      <p><strong>ID:</strong> <%= incidencia.id %></p>
      <p><strong>Departament:</strong> <%= incidencia.departament %></p>
      <p><strong>Descripció:</strong> <%= incidencia.descripcion %></p>
      <p><strong>Estat:</strong> <span class="badge bg-secondary"><%= incidencia.estado %></span></p>

      <h6 class="mt-4">Actuacions visibles per <%= incidencia.usuario %></h6>
      <ul class="list-group mt-2">
        <%
          const visibles = (incidencia.actuacions || []).filter(a => a.usuario === incidencia.usuario);
          if (visibles && visibles.length > 0) {
            visibles.forEach(actuacio => {
              <li class="list-group-item">
                <i class="bi bi-check-circle text-success me-2"></i><%= actuacio.descripcion %>
              </li>
            <% })
          } else {
            <li class="list-group-item text-muted fst-italic">Cap actuació registrada.
          <% } %>
        <%
      </ul>
    </div>
  </div>
<% }
```

router /buscar

```
router.post('/buscar', async (req, res) => {
  const { id } = req.body;
  if (!id || id.trim() === '') {
    return res.render('index', { error: 'Has d'introduir una ID.', incidencia: null });
  }
  try {
    const incidencia = await Incidencia.findOne({
      where: { id: id },
      include: [
        { model: Departamento, as: 'departament' },
        { model: Estat, as: 'estat' },
        { model: Actuacio, as: 'actuacions' },
        { model: Tipus, as: 'tipus' }
      ]
    });

    if (incidencia) {
      incidencia.actuacions = incidencia.actuacions.filter(
        a => a.visible_per_usuari === true || a.visible_per_usuari === 1 || a.visible_per_usuari === 2
      );


      return res.render('index', { incidencia, error: null });
    } else {
      return res.render('index', { error: 'No s'ha trobat la incidència.', incidencia: null });
    }
  } catch (error) {
    console.error(error);
    return res.render('index', { error: 'Error al buscar la incidència.', incidencia: null });
  }
});
```

Comença verificant si l'usuari a posat una id, si està buit, ens tornarà un error “Has d'introduir una ID”. Seguidament, cerca l'incidència a la base de dades amb **await Incidencia.findOne**. A continuació, filtra les actuacions que són visibles per l'usuari (**true**, **1**, **'true'**).

Per últim, si troba l'incidència, mostra el **if (incidencia)** explicat abans, en canvi, si no el troba, mostrarà el missatge següent “No s'ha trobat l'incidència”

Així es veu la pàgina principal amb el **<form>** per escriure el número d'incidència

Benvingut al projecte Incidències

 Crear incidència

Consulta l'estat d'una incidència

Cerca

Resultat de la cerca

ID: 2

Departament: Departament d'Informàtica

Descripció: Cap dispositiu es connecta a la xarxa del segon pis.

Estat:

En procés

Actuacions visibles per l'usuari:

☒ Reinici del punt d'accés del segon pis.

☒ Es detecta avaria a l'switch. Es demana reemplaçament.