

Trabajo de Mongoose + Express

Endpoints y capturas

Todo empieza en local, usamos el live server para arrancar las paginas html del formulario, entonces se podría decir que el primer endpoint es: 127.0.0.1:5501/form_empleados.html, 127.0.0.1:5501/form_cliente.html, 127.0.0.1:5501/form_casa.html.

Empleados casas clientes Lista empleados lista casas lista clientes

Mongo-project by Danny is licensed under [CC BY-NC 4.0](#) © ⓘ

127.0.0.1:5501/form_empleados.html

127.0.0.1:5501/form_empleados.html

Empleados **casas** clientes Lista empleados lista casas lista clientes

Mongo-project by Danny is licensed under [CC BY-NC 4.0](#) © ⓘ

127.0.0.1:5501/form_casa.html


127.0.0.1:5501/form_casa.html

Empleados casas **clientes** lista empleados lista casas lista clientes

nombre

documento

enviar

Mongo-project by Danny is licensed under [CC BY-NC 4.0](#) 

127.0.0.1:5501/form_cliente.html

Luego pasamos ya a los endpoints de node, de aquí en adelante ya nada es html

```
app.post('/submit/empleado', async (req, res) => {  
  try {  
    const campo1 = req.body.campo1;  
    const campo2 = req.body.campo2;  
    const campo3 = req.body.campo3;  
    const tabla_empleado = new mongo_empleado({  
      name: campo1,  
      edad: campo2,  
      cargo: campo3  
    });  
    const resultado = await tabla_empleado.save();  
    res.redirect('http://localhost:5501/form_empleados.html')  
  } catch (error) {  
    // atencion: si se usa otro puerto para el live server el boton de vuelta no funcionara  
    res.send(`  
      <p>NO SE PUDO INTRODUCIR</p>  
      <a href="http://127.0.0.1:5501/form_empleados.html">Volver</a>  
    `);  
  }  
})
```

Esto es el localhost:3000/submit/empleados para añadir los empleados via formulario, al introducir el usuario te devuelve a la pagina de formulario asi que esta pagina no tiene ventana grafica, lo mismo con el submit de casas y clientes

```

app.post('/submit/casa', async (req, res) => {

  try {
    const campo1 = req.body.campo1;
    const campo2 = req.body.campo2;
    const campo3 = req.body.campo3;
    const tabla_casa = new mongo_casa({
      direccion: campo1,
      dimensiones: campo2,
      precio: campo3,
    });
    const resultado = await tabla_casa.save();
    res.redirect('http://localhost:5501/form_casa.html')
  } catch (error) {
    // atencion: si se usa otro puerto para el live server el boton de vuelta no funcionara
    res.send(`
      <p>NO SE PUDO INTRODUCIR</p>
      <a href="http://127.0.0.1:5501/form_casa.html">Volver</a>
    `);
  }
})

```

```

app.post('/submit/cliente', async (req, res) => {

  try {
    const campo1 = req.body.campo1;
    const campo2 = req.body.campo2;
    const tabla_cliente = new mongo_cliente({
      name: campo1,
      documento: campo2,
    });
    const resultado = await tabla_cliente.save();
    res.redirect('http://localhost:5501/form_cliente.html')
  } catch (error) {
    // atencion: si se usa otro puerto para el live server el boton de vuelta no funcionara
    res.send(`
      <p>NO SE PUDO INTRODUCIR</p>
      <a href="http://127.0.0.1:5501/form_cliente.html">Volver</a>
    `);
  }
})

```

Luego tenemos los endpoints de listas: localhost:3000/lista/empleados, localhost:3000/lista/casas, localhost:3000/lista/clientes.

Lista de empleados

| id | Nombre | edad | cargo | acciones |
|--------------------------|-----------|------|-----------|---|
| 666de143d87d3ac67f28ad9d | Danny | 23 | jefe | eliminar editar |
| 666de14bd87d3ac67f28ad9f | Jeff | 26 | subjefe | eliminar editar |
| 666de154d87d3ac67f28ada1 | Afredo | 45 | Ingeniero | eliminar editar |
| 666de15ed87d3ac67f28ada3 | Sebastian | 32 | Albañil | eliminar editar |

[Volver](#)

El de los empleados

El de los empleados

Lista de casas

| id | direccion | dimensiones | precio | fecha creacion | fecha modificacion | acciones |
|--------------------------|-----------------|-------------|----------|---|---|---|
| 666de16dd87d3ac67f28ada5 | c/ almirante 23 | 120000 m/2 | 250000 € | Sat Jun 15 2024 19:46:05 GMT+0100 (hora de verano de Europa occidental) | Sat Jun 15 2024 19:46:05 GMT+0100 (hora de verano de Europa occidental) | eliminar editar |
| 666de17cd87d3ac67f28ada7 | c/ alioe 54 | 450000 m/2 | 600000 € | Sat Jun 15 2024 19:46:20 GMT+0100 (hora de verano de Europa occidental) | Sat Jun 15 2024 19:46:20 GMT+0100 (hora de verano de Europa occidental) | eliminar editar |
| 666de191d87d3ac67f28ada9 | c/ sesame | 340000 m/2 | 550000 € | Sat Jun 15 2024 19:46:41 GMT+0100 (hora de verano de Europa occidental) | Sat Jun 15 2024 19:46:41 GMT+0100 (hora de verano de Europa occidental) | eliminar editar |

[Volver](#)

El de las casas

Lista de clientes

| id | Nombre | documento | acciones |
|--------------------------|----------|-----------|---|
| 666de1a7d87d3ac67f28ada8 | Carmelo | 2344564P | eliminar editar |
| 666de1b0d87d3ac67f28ada9 | Fernando | 1235436J | eliminar editar |
| 666de1b7d87d3ac67f28adaF | Manuela | 3457680P | eliminar editar |

[Volver](#)

El de los clientes

```
app.get('/lista/casas', async (req, res) => {
  try {
    const casas = await mongo_casa.find().exec();
    let html = `<h1>Lista de casas</h1>
    <table border="1">
      <thead>
        <th>id</th>
        <th>direccion</th>
        <th>dimensiones</th>
        <th>precio</th>
        <th>fecha creacion</th>
        <th>fecha modificacion</th>
        <th colspan="2">acciones</th>
      </thead>`

    casas.forEach(casa => {
      html += `
      <tr>
        <td>${casa._id}</td>
        <td>${casa.direccion}</td>
        <td>${casa.dimensiones} m/2</td>
        <td>${casa.precio} €</td>
        <td>${casa.createdAt}</td>
        <td>${casa.updatedAt}</td>
        <td><a href="http://localhost:3000/borrado/casa/${casa._id}">eliminar</a></td>
        <td><a href="http://localhost:3000/update/casa/${casa._id}">editar</a></td>
      `;
    });
    html += `</tr>
    </table>`;
    res.send(`${html}<a href="http://127.0.0.1:5501/form_casa.html">Volver</a>`);
  } catch (error) {
    res.send(`
    <p>NO SE PUDO ENCONTRAR NADA</p>
    <a href="http://127.0.0.1:5501/form_casa.html">Volver</a>
    `);
  }
})
```

Así es como se ve el código genérico para los 3, básicamente recibe por get que tipo de lista tiene que pintar y luego imprime la tabla por html con toda la información. De esta página también se accede a los otros dos niveles de crud, el update y el delete.

```

<td><a href="http://localhost:3000/borrado/casa/${casa._id}">eliminar</a></td>
<td><a href="http://localhost:3000/update/casa/${casa._id}">editar</a></td>
`;

```

El borrado tampoco tiene vista grafica ya que despues de borrar te devuelve a la pagina de la lista, por lo tanto la acción es inmediata, recibe por get que elemento va a borrar y su id y luego ejecuta la función de mongoose para borrar el elemento por id.

```

app.get('/borrado/empleado/:id', async (req, res) => {
  let id = req.params.id;
  const borrar_empleado = await mongo_empleado.findOneAndDelete(id);
  res.redirect('http://localhost:3000/lista/empleados');
})
app.get('/borrado/casa/:id', async (req, res) => {
  let id = req.params.id;
  const borrar_casa = await mongo_casa.findOneAndDelete(id);
  res.redirect('http://localhost:3000/lista/casas');
})
app.get('/borrado/cliente/:id', async (req, res) => {
  let id = req.params.id;
  const borrar_cliente = await mongo_cliente.findOneAndDelete(id);
  res.redirect('http://localhost:3000/lista/clientes');
})

```

El update es mas o menos similar con la diferencia que son dos pasos en vez de uno. Recibe el id por get, luego busca en la base de datos el elemento por id, pinta un formulario con los campos actuales del elemento encontrado, nosotros modificamos los valores del formulario, estos se envían por POST y se actualizan en otro consulta mas abajo, actualizando asi el registro.

```

app.get('/update/empleado/:id', async (req, res) => {
  try {
    const empleado = await mongo_empleado.findById(req.params.id);

    res.send(`
      <form action="http://localhost:3000/update/empleado/${empleado._id}" method="POST" class="d-flex justify-content-center col-6">
        <div class="d-flex flex-column text-center">
          <p><input type="text" name="nombre" class="rounded-2 input-group-text" placeholder="nombre" value="${empleado.nombre}"></p>
          <p><input type="number" name="edad" class="rounded-2 input-group-text" placeholder="edad" value="${empleado.edad}"></p>
          <p><input type="text" name="cargo" class="rounded-2 input-group-text" placeholder="cargo" value="${empleado.cargo}"></p>
          <p><input type="submit" value="enviar" class="btn btn-secondary"></p>
        </div>
      </form>
    `);
  } catch (error) {
    res('Error al obtener el empleado');
  }
});

app.post('/update/empleado/:id', async (req, res) => {
  try {
    const { nombre, edad, cargo } = req.body;

    await mongo_empleado.findByIdAndUpdate(req.params.id, {
      name: nombre,
      edad: edad,
      cargo: cargo
    });

    res.redirect('http://localhost:3000/lista/empleados');
  } catch (error) {
    res.send('Error al actualizar el empleado');
  }
});

```

Valoracion final:

CRUD completo + utilizo de varios tipos de datos: 5 puntos;

Ampliación

Otra colección: hay 3 colecciones

Frontend WEB: hay

Gestion de control de errores: hay