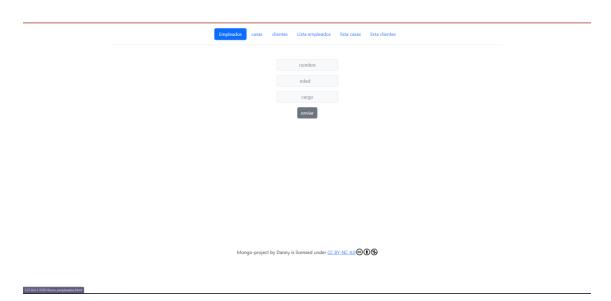
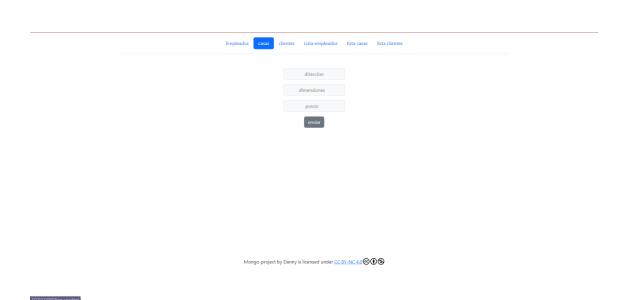
Trabajo de Mongoose + Express

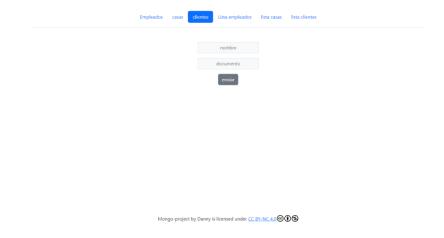
Endpoints y capturas

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.



127.0.0.1:5501/form_empleados.html





127.0.0.1:5501/form_cliente.html

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

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

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

Lista de casas

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

El de las casas

Lista de clientes

```
id | Nombre | documento | acciones | 666dela7d57d3ac67f28adab[carmelo ] 23445649 | eliminar | deltar | 666delab0d57d3ac67f28adaf | fernando | 12543659 | eliminar | deltar | 666delab1d87d3ac67f28adaf | fernando | 124656890 | eliminar | deltar | 646delab1d87d3ac67f28adaf | fernando | 124576890 | eliminar | deltar | 646delab1d87d3ac67f28adaf | fernando | 124576890 | eliminar | deltar | 646delab1d87d3ac67f28adaf | fernando | 124576890 | eliminar | deltar | 646delab1d87d3ac67f28adaf | fernando | 124576890 | eliminar | deltar | 646delab1d87d3ac67f28adaf | fernando | 124576890 | eliminar | deltar | 646delab1d87d3ac67f28adaf | fernando | 124576890 | eliminar | deltar | 646delab1d87d3ac67f28adaf | fernando | 124576890 | eliminar | deltar | 646delab1d87d3ac67f28adaf | 646delab1d87d3ac6
```

El de los clientes

```
const casas = await mongo_casa.find().exec();
  let html = `<h1>Lista de casas</h1>
  <thead>
     id
     direction
     dimensiones
     precio
     fecha creacion
     fecha modificacion
     acciones
     </thead>`
  casas.forEach(casa => {
     html +=
     \t d\{casa._id}
     ${casa.direccion}
     ${casa.precio} €
     ${casa.createdAt}
     ${casa.updatedAt}
     \label{local-host} $$ \downarrow href=$$ \frac{1}{http://localhost:3000/borrado/casa/${casa._id}$$ \eliminar</a</td>
     <a href="http://localhost:3000/update/casa/${casa._id}">editar</a</td>
  html += `
    `;
 res.send(`${html}<a href="http://127.0.0.1:5501/form_casa.html">Volver</a>`);
catch (error) {
    NO SE PUDO ENCONTRAR NADA
     <a href="http://127.0.0.1:5501/form_casa.html">Volver</a>
```

Asi 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 pagina tambien se accede a los otros dos niveles de crud, el update y el delete.

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

```
/update/empleado/:id', async (req, res) =>
         const empleado = await mongo_empleado.findById(req.params.id);
              (form action="http://localhost:3000/update/empleado/${empleado.<u>id}</u>" method="POST" class="d-flex justify-content-center col-6">
                   <div class="d-flex flex-column text-center</pre>
                      cpx.input type="cext name="name" allows="rounded-2 input-group-text" placeholder="edad" value="$[empeado.edad"]">/
<input type="text" name="cargo" class="rounded-2 input-group-text" placeholder="edad" value="$[empleado.edad]">/
<input type="text" name="cargo" class="rounded-2 input-group-text" placeholder="cargo" value="$[empleado.edad]">/
                       \label{lem:class} $$\p>\input type="submit" value="enviar" class="btn btn-secondary">
                  </div>
             </form>
    } catch (error) {
    res('Error al obtener el empleado');
app.post('/update/empleado/:id', async (req, res) => {
         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');
        atch (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