

Módulos Integrados

Julian Eduardo Garcia Camargo

Competencia, Competencias Técnicas

Servicio Nacional de Aprendizaje, SENA

2521992, Análisis y Desarrollo de Software

Instructor, Alexandra Soraya Beltrán Castro

07 de Septiembre

INTRODUCCIÓN

La realización del código “*Full Stack*” para el proyecto ayudó a precisar el las líneas de código, la sintaxis y las encapsulaciones de datos importados o exportados dentro del *BackEnd* y *FrontEnd*... sin embargo, la modificación de la ruta para la conexión con la DB en la *BackEnd*, fue un error que nunca se me había presentado.

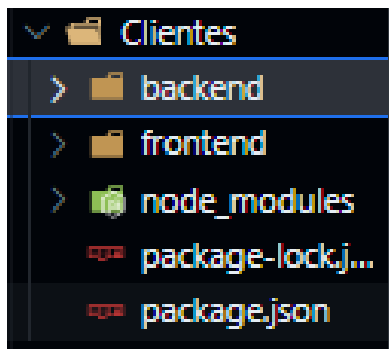
Gracias a la reflexión, cambio de código(por lo general lo escribo desde 0 para acomodarme a la sintaxis de cada aspecto o archivo de código), comparación del código con la guía administrada en el componente formativo y al acompañamiento de los diferentes videos en la plataforma *YouTube*, pude solucionar el pequeño tropiezo del proyecto que demore unas extensas horas.

INTEGRACIÓN DE MODULOS

URL para el Repositorio GitHub

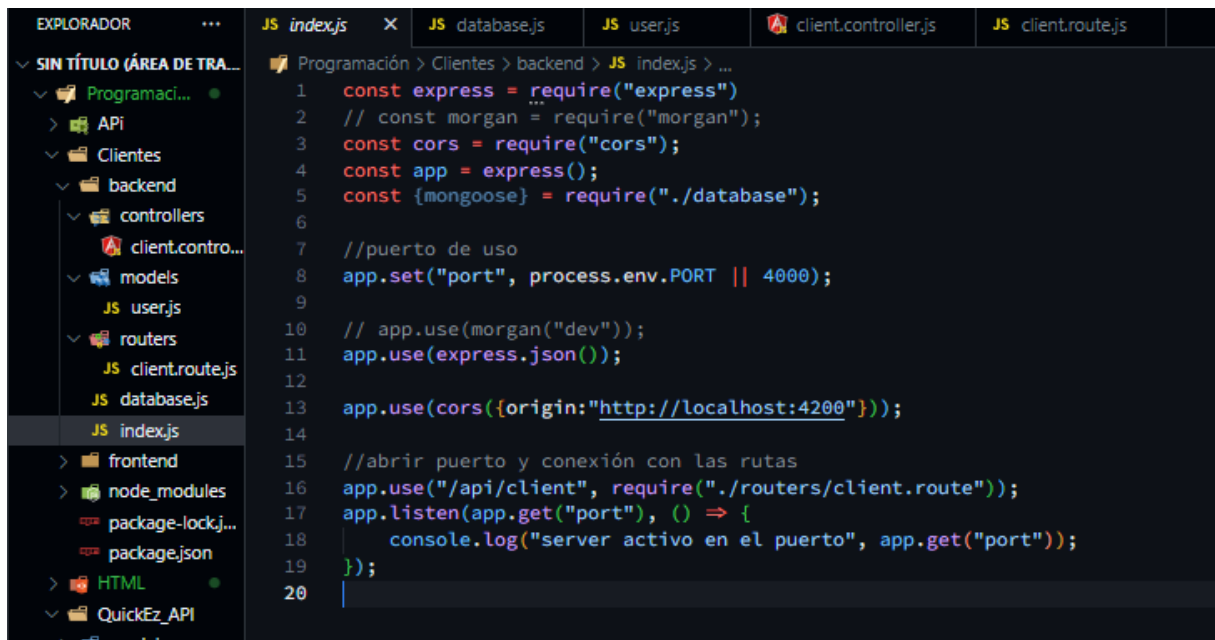
<https://github.com/Dexkx/Proyecto-GA8-AA1>

Carpetas base para la composición del proyecto



BackEnd

Código fuente del *BackEnd*



Código de conexión a la DB, cambiar la conexión “localhost” por “127.0.0.1” para no recibir *error 404 (not found)* en la consola

```
Programación > Clientes > backend > JS database.js > [?] <unknown>
1  +  const mongoose = require("mongoose");
2
3      //conexion con la DB
4      const uri = "mongodb://127.0.0.1:27017/QuickEz";
5      mongoose.connect(uri)
6          .then(console.log("DB Conectada"))
7          .catch(err => console.error(err));
8
9      module.exports = mongoose
```

Código de modelado de estructura de los clientes

```
JS index.js  JS database.js  JS user.js  X  client.controller.js  JS client.route.js
Programación > Clientes > backend > models > JS user.js > [?] <unknown>
1  const mongoose = require("mongoose");
2  const Schema = mongoose.Schema;
3
4  const usuarioSchema = new Schema({
5      num_document : {type : Number, require : true},
6      name_client : {type : String, require : true},
7      email : {type : String, require : true}
8  });
9
10 module.exports = mongoose.model("client", usuarioSchema)
```

Código de controladores en *BackEnd*

```
Programación > Clientes > backend > controllers > client.controller.js > editClient
1 + const Client = require("../models/user");
2   const cCtrol = {};
3
4   //Creación de usuario
5   cCtrol.createClientes = async (req, res) =>{
6     const client = new Client(req.body);
7     await client.save();
8     res.json({
9       status : "Usuario Guardado"
10    });
11  }
12
13  //Filtrado general
14  cCtrol.getClientes = async (req, res) =>{
15    const clientes = await Client.find();
16    res.json(clientes)
17  }
18
19  //Filtrado único
20  cCtrol.getOneClient = async (req, res) =>{
21    const Oneclient = await Client.findById(req.params.id);
22    res.json(Oneclient);
23  }
24
25  //Actualizar usuario
26  cCtrol.editClient = async (req, res) =>{
27    const {id} = req.params;
28
29    const newClient = {
30      num_document : req.body.num_document,
31      name_client : req.body.name_client,
32      email : req.body.email
33    };
34
35    await Client.findByIdAndUpdate(id, {$set : newClient}, {new : true});
36    res.json({
37      status : "Usuario Actualizado"
38    });
39  }
40
```

```

0 +
1 //Eliminar Usuario
2 cCtrl.deleteClient = async (req,res) => {
3     await Client.findByIdAndDelete(req.params.id);
4     res.json({
5         status : "Usuario Elimnado"
6     });
7 }
8
9 module.exports = cCtrl

```

Código de las rutas en el *BackEnd*

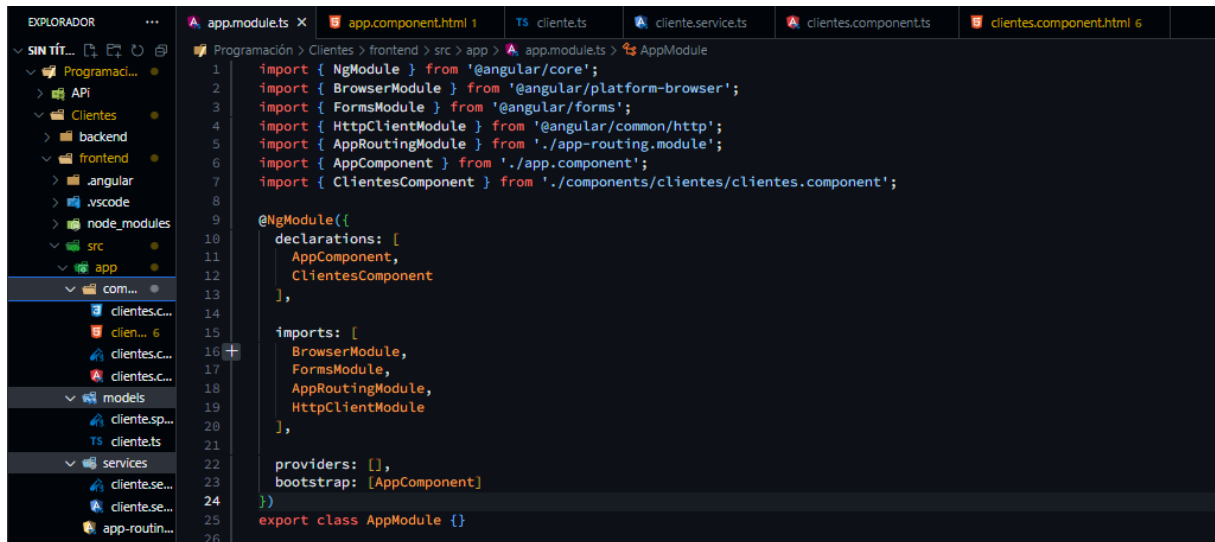
```

Programación > Clientes > backend > routers > JS client.route.js > [?] <unknown>
1 const express = require("express");
2 const router = express.Router();
3 + const cClient = require("../controllers/client.controller");
4
5 router.post("/", cClient.createClientes);
6
7 router.get("/", cClient.getClientes);
8 router.get("/:id", cClient.getOneClient);
9
10 router.put("/:id", cClient.editClient);
11
12 router.delete("/:id", cClient.deleteClient);
13
14 module.exports = router

```

FrontEnd

Principales carpetas y archivos utilizados para el diseño *FronEnd*, código de módulos de librerías de *Angular* para utilizar



```
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { FormsModule } from '@angular/forms';
4 import { HttpClientModule } from '@angular/common/http';
5 import { AppRoutingModule } from './app-routing.module';
6 import { AppComponent } from './app.component';
7 import { ClientesComponent } from './components/clientes/clientes.component';
8
9 @NgModule({
10   declarations: [
11     AppComponent,
12     ClientesComponent
13   ],
14   imports: [
15     BrowserModule,
16     FormsModule,
17     AppRoutingModule,
18     HttpClientModule
19   ],
20   providers: [],
21   bootstrap: [AppComponent]
22 })
23 export class AppModule {}
```

Código principal con extensión *.html* del servidor



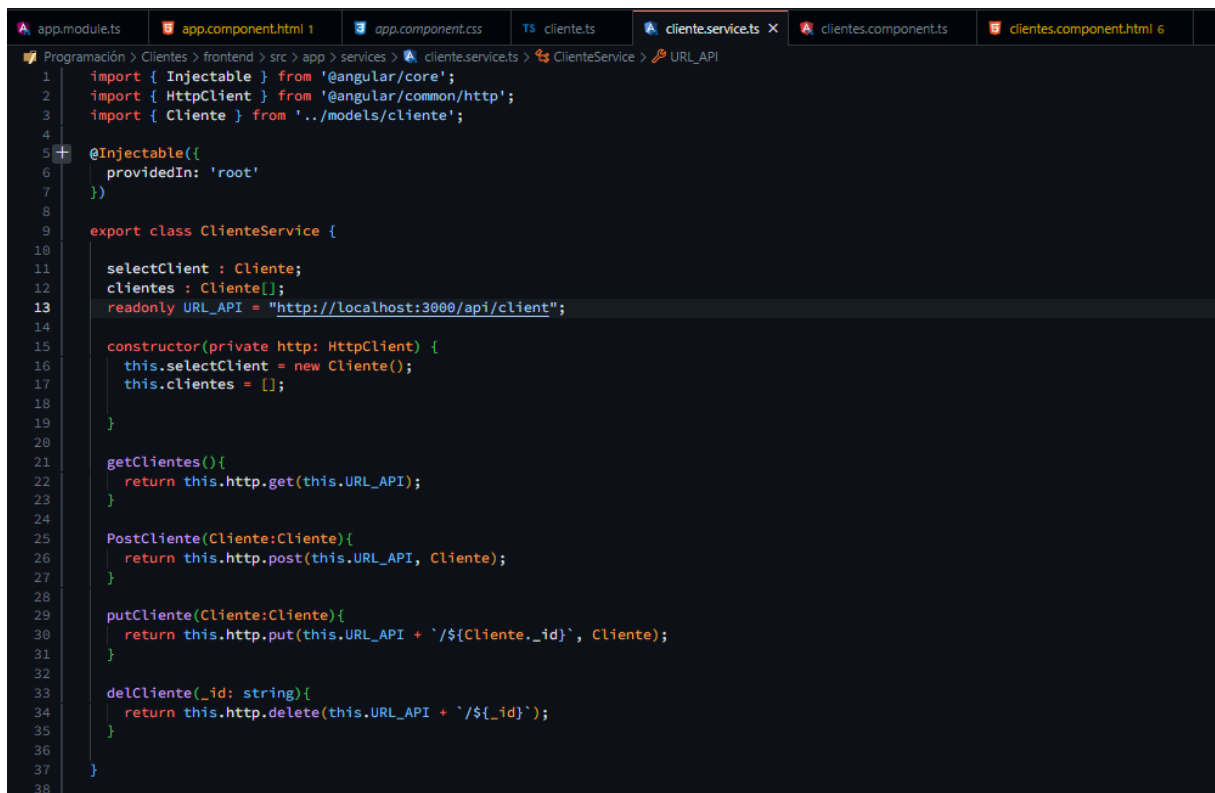
```
1 <nav class="nav-wrapper blue">
2   <div class="container">
3     <a href="/" class="brand-logo">GESTIÓN DE CLIENTES</a>
4   </div>
5 </nav>
6
7 <div class="container p-4">
8   <app-clientes></app-clientes>
9 </div>
```

Código del modulado *FrontEnd* de la estructura para la DB



```
1 export class Cliente {
2
3   _id: number;
4   num_document: number;
5   name_client: string;
6   email: string;
7
8   constructor(_id = 0, num_document = 0, name_client = "", email = "") {
9     this._id = _id;
10    this.num_document = num_document;
11    this.name_client = name_client;
12    this.email = email;
13  }
14
15
16 }
17
```

Código controladores y rutas(*BackEnd*), en realidad de servicios(*FrontEnd*)



```
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Cliente } from '../models/cliente';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8
9 export class ClienteService {
10
11   selectClient : Cliente;
12   clientes : Cliente[];
13   readonly URL_API = "http://localhost:3000/api/client";
14
15   constructor(private http: HttpClient) {
16     this.selectClient = new Cliente();
17     this.clientes = [];
18   }
19
20   getClientes(){
21     return this.http.get(this.URL_API);
22   }
23
24   PostCliente(Cliente:Cliente){
25     return this.http.post(this.URL_API, Cliente);
26   }
27
28   putCliente(Cliente:Cliente){
29     return this.http.put(this.URL_API + `/${Cliente._id}`, Cliente);
30   }
31
32   delCliente(_id: string){
33     return this.http.delete(this.URL_API + `/${_id}`);
34   }
35 }
36
37
38
```


Código de componentes para almacenar los datos del archivo *.html* próximamente
conectado y, enviarlos al *client.service.ts*

```
Programación > Clientes > frontend > src > app > components > clientes > clientes.component.ts > ...
1  import { Component, OnInit } from "@angular/core";
2  import { ClienteService } from "../../services/cliente.service";
3  import { NgForm } from "@angular/forms";
4  import { Cliente } from "src/app/models/cliente";
5
6  declare var M: any;
7
8  @Component({
9    selector: 'app-clientes',
10    templateUrl: './clientes.component.html',
11    styleUrls: ['./clientes.component.css'],
12    providers: [ClienteService]
13  })
14
15  export class ClientesComponent implements OnInit{
16
17    constructor(public clienteService:ClienteService){}
18
19    ngOnInit(): void {
20    }
21
22    addClient(form?:NgForm){
23      this.clienteService.PostCliente(form?.value)
24        .subscribe(res => {
25          this.resetForm(form);
26          M.toast({html:"Guardado Satisfactoriamente"});
27        });
28    }
29
30    resetForm(form?:NgForm){
31      if (form){
32        form.reset();
33        this.clienteService.selectClient = new Cliente();
34      }
35    }
36  }
37
```

Código con extensión *.html* en donde se expone el formulario a realizar por el cliente y almacenado en la DB

```
Programación > Clientes > frontend > src > app > components > clientes > clientes.component.html > div.container > div.row > div.col.s5 > div.card > div.card-content > form > div.row
1
2 <div class="container">
3   <div class="row">
4     <div class="col s5">
5       <div class="card">
6         <div class="card-content">
7           <form name="userForm" #userForm="ngForm" (ngSubmit)="addClient(userForm)">
8             <div class="row">
9               <div class="input-field col s12">
10                <input type="text" name="num_document" #num_document="ngModel" [(ngModel)]="clienteService.selectClient.num_document" placeholder="Ingrese su número de documento">
11              </div>
12              <div class="input-field col s12">
13                <input type="text" name="name_client" #name_client="ngModel" [(ngModel)]="clienteService.selectClient.name_client" placeholder="Ingrese su nombre">
14              </div>
15              <div class="input-field col s12">
16                <input type="text" name="email" #email="ngModel" [(ngModel)]="clienteService.selectClient.email" placeholder="Ingrese su email">
17              </div>
18              <div class="card-action">
19                <div class="input-field col s12">
20                  <button class="btn right" (click)="resetForm(userForm)">Limpiar</button>
21                  <button class="btn" >Guardar </button>
22                </div>
23              </div>
24            </div>
25          </form>
26        </div>
27      </div>
28    </div>
29    <div class="col s7">
30      <div>
31      </div>
32    </div>
  </div>
```

```
Programación > Clientes > frontend > src > app > components > clientes > clientes.component.html > div.container > div.row > div.col.s5 > div.card > div.card-content > form > div.row
1
2
3
4
5
6
7 <form (ngSubmit)="addClient(userForm)">
8
9
10 <div class="input-field col s12">
11   <input type="text" name="num_document" #num_document="ngModel" [(ngModel)]="clienteService.selectClient.num_document" placeholder="Ingrese su número de documento">
12 </div>
13 <div class="input-field col s12">
14   <input type="text" name="name_client" #name_client="ngModel" [(ngModel)]="clienteService.selectClient.name_client" placeholder="Ingrese su nombre">
15 </div>
16 <div class="input-field col s12">
17   <input type="text" name="email" #email="ngModel" [(ngModel)]="clienteService.selectClient.email" placeholder="Ingrese su email">
18 </div>
19 <div class="card-action">
20   <div class="input-field col s12">
21     <button class="btn right" (click)="resetForm(userForm)">Limpiar</button>
22     <button class="btn" >Guardar </button>
23   </div>
24 </div>
25
26
27
28
29
30
31
32
```

Funcionabilidad del Servidor

Empezamos iniciando la conexión con el servidor desde el *BackEnd*



```
1 const express = require("express");
2 // const morgan = require("morgan");
3 const cors = require("cors");
4 const app = express();
5 const {mongoose} = require("../database");
6
7 // puerto de uso
8 app.set("port", process.env.PORT || 3000);
9
10 // app.use(morgan("dev"));
11 app.use(express.json());
12
13 app.use(cors({origin:"http://localhost:4200"}));
14
15 //abrir puerto y conexión con las rutas
16 app.use("/api/client", require("../routes/client.route"));
17 app.listen(app.get("port"), () => {
18   console.log("server activo en el puerto", app.get("port"));
19 });
20
```

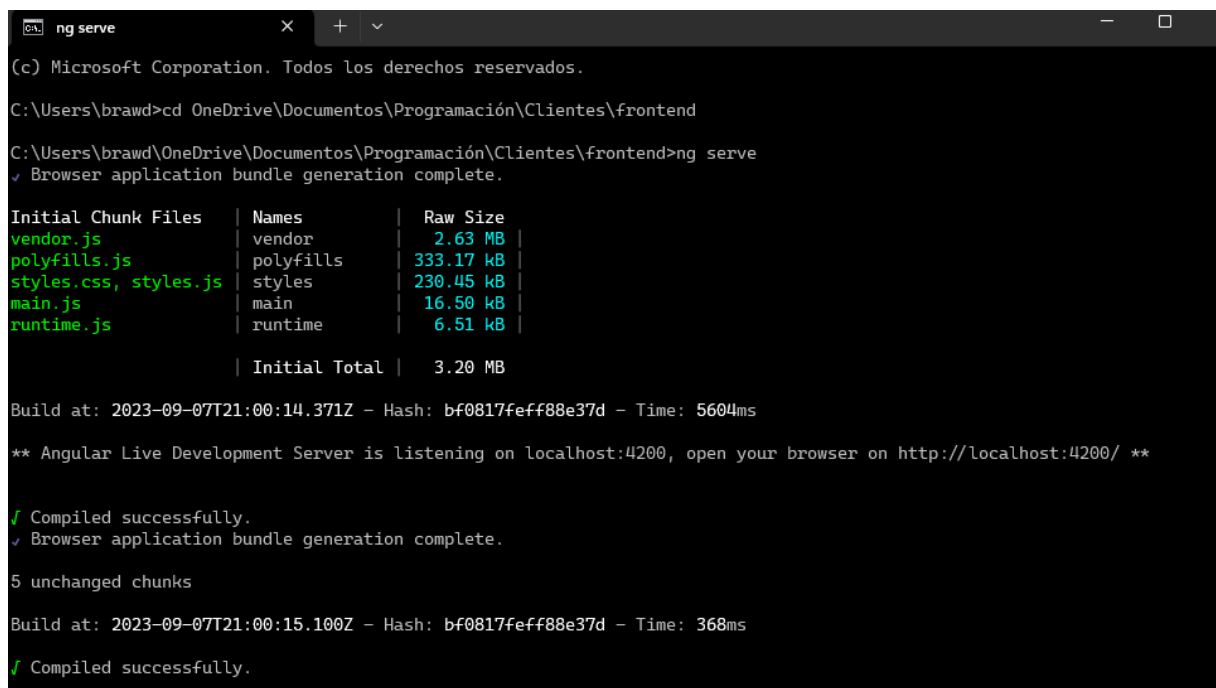
PROBLEMAS TERMINAL SALIDA PUERTOS COMENTARIOS CONSOLA DE DEPURACIÓN

[Running] node "c:\Users\brawd\OneDrive\Documentos\Programación\Clientes\backend\index.js"

DB Conectada

server activo en el puerto 3000

Luego visualizamos la app con el comando `ng serve` desde nuestro archivo *frontend*



```
C:\> ng serve

(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\brawd>cd OneDrive\Documentos\Programación\Clientes\frontend

C:\Users\brawd\OneDrive\Documentos\Programación\Clientes\frontend>ng serve
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size
vendor.js           | vendor | 2.63 MB
polyfills.js        | polyfills | 333.17 kB
styles.css, styles.js | styles | 230.45 kB
main.js             | main | 16.50 kB
runtime.js          | runtime | 6.51 kB
                    | Initial Total | 3.20 MB

Build at: 2023-09-07T21:00:14.371Z - Hash: bf0817feff88e37d - Time: 5604ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

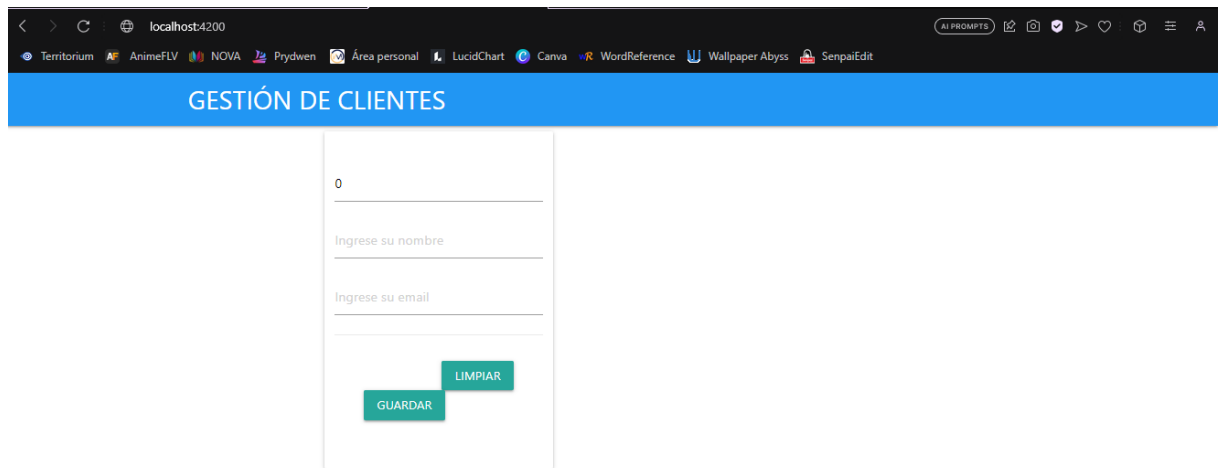
✓ Compiled successfully.
✓ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2023-09-07T21:00:15.100Z - Hash: bf0817feff88e37d - Time: 368ms

✓ Compiled successfully.
```

Verificamos en la ruta <http://localhost:4200>, ruta que hemos modificado para la utilización del servidor del proyecto



The screenshot shows a web browser window with the address bar displaying 'localhost:4200'. The browser's taskbar at the top lists several open applications: Territorium, AnimeFLV, NOVA, Prydwen, Área personal, LucidChart, Canva, WordReference, Wallpaper Abyss, and SenpaiEdit. The webpage itself has a blue header with the text 'GESTIÓN DE CLIENTES'. Below the header, there is a form with a white background and a thin grey border. The form contains a text input field with the number '0' inside. Below this, there are two more text input fields with placeholder text 'Ingrese su nombre' and 'Ingrese su email'. At the bottom of the form, there are two green buttons: 'GUARDAR' on the left and 'LIMPIAR' on the right.

Subimos datos a la DB en *mongoDB*

GESTIÓN DE CLIENTES



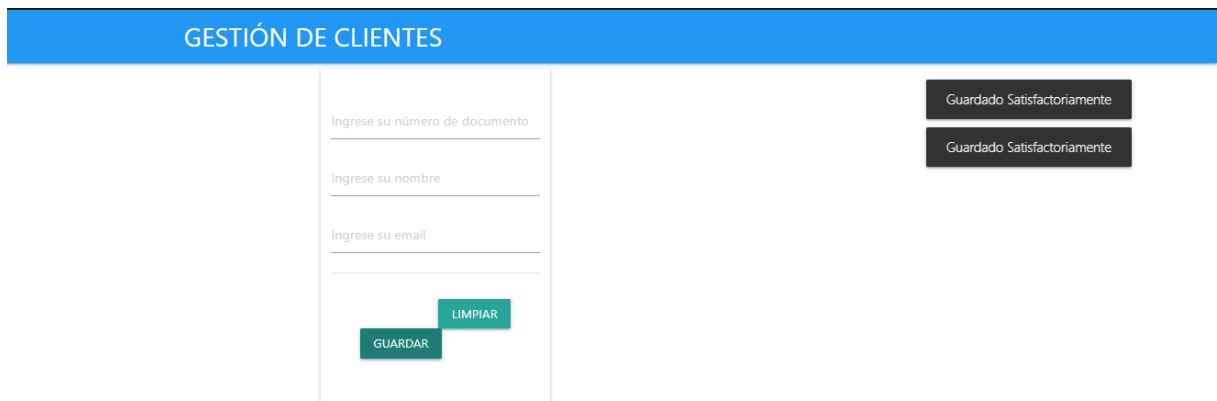
Formulario de gestión de clientes con los siguientes campos pre-llenados:

- Número de documento: 123908
- Nombre: Geraldine Toro
- Email: Gera.toro@gmail.com

Botones de acción:

- GUARDAR
- LIMPIAR

En mi caso me proporcionan dos mensajes del servidor por culpa de mi mouse(hardware) defectuoso, naturalmente solo notifica una vez



Formulario de gestión de clientes con los siguientes campos:

- Ingrese su número de documento
- Ingrese su nombre
- Ingrese su email

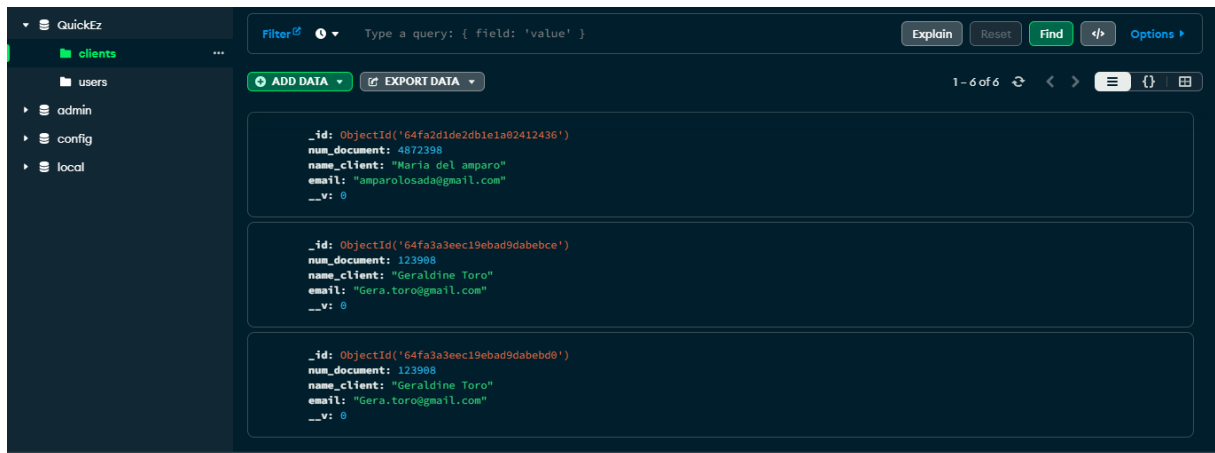
Botones de acción:

- GUARDAR
- LIMPIAR

Mensajes de éxito:

- Guardado Satisfactoriamente
- Guardado Satisfactoriamente

Corroboramos los datos subidos a la DB en *mongoDB*



Como podemos observar se realizo la inserción de dos datos con los mismo parámetros, pero esto esta relacionado con mi mouse defectuoso, por lo que, naturalmente solo se nos vería reflejado un datos almacenado