

Documentación de la prueba Técnica Edgar Mauricio Ruiz

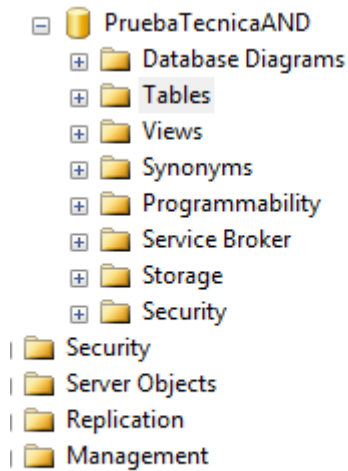
Definición de la arquitectura:

Para el FrontEnd, se utilizará Angular.

Para el BackEnd, utilizo Servicios Web API Rest mediante IDE Visual Studio 2019. La Base de Datos la construí en Sql Server

Construcción de la Base de Datos:

Se construyó una Base de Datos denominada PruebaTecnicaAND,



en ella se crearon cuatro tablas denominadas:

-Tramites

-Opinion

-Cuentanos

-OtrosTemas

| PORTATILN.Prueba...ND - dbo.Tramites ✕ | | | |
|----------------------------------------|-----------------|---------------|--------------------------|
| | Column Name | Data Type | Allow Nulls |
| 🔑 | id | int | <input type="checkbox"/> |
| | nombre | nvarchar(MAX) | <input type="checkbox"/> |
| | entidad | nvarchar(100) | <input type="checkbox"/> |
| | disponiblelinea | bit | <input type="checkbox"/> |
| | concosto | bit | <input type="checkbox"/> |
| ▶ | detalletramite | nvarchar(MAX) | <input type="checkbox"/> |

| PORTATILN.Prueba...AND - dbo.Opinion ✕ | | | |
|----------------------------------------|-------------|---------------|--------------------------|
| | Column Name | Data Type | Allow Nulls |
| ▶🔑 | id | int | <input type="checkbox"/> |
| | estado | int | <input type="checkbox"/> |
| | nombre | nvarchar(MAX) | <input type="checkbox"/> |
| | entidad | nvarchar(100) | <input type="checkbox"/> |

| PORTATILN.Prueba...D - dbo.Cuentanos ✕ PORTATILN.Prueba...AND - dbo.Opinion | | | |
|-----------------------------------------------------------------------------|-------------|---------------|--------------------------|
| | Column Name | Data Type | Allow Nulls |
| ▶🔑 | id | int | <input type="checkbox"/> |
| | idopinion | int | <input type="checkbox"/> |
| | descripcion | nvarchar(MAX) | <input type="checkbox"/> |

| PORTATILN.PruebaT...- dbo.OtrosTemas ✕ PORTATILN.Prueba...D - dbo.C | | | |
|---------------------------------------------------------------------|-------------|---------------|--------------------------|
| | Column Name | Data Type | Allow Nulls |
| ▶🔑 | id | int | <input type="checkbox"/> |
| | titulo | nvarchar(MAX) | <input type="checkbox"/> |
| | subtitulo | nvarchar(MAX) | <input type="checkbox"/> |
| | enlace | nvarchar(MAX) | <input type="checkbox"/> |
| | imagen | nvarchar(MAX) | <input type="checkbox"/> |

Se llenan datos en las Tablas

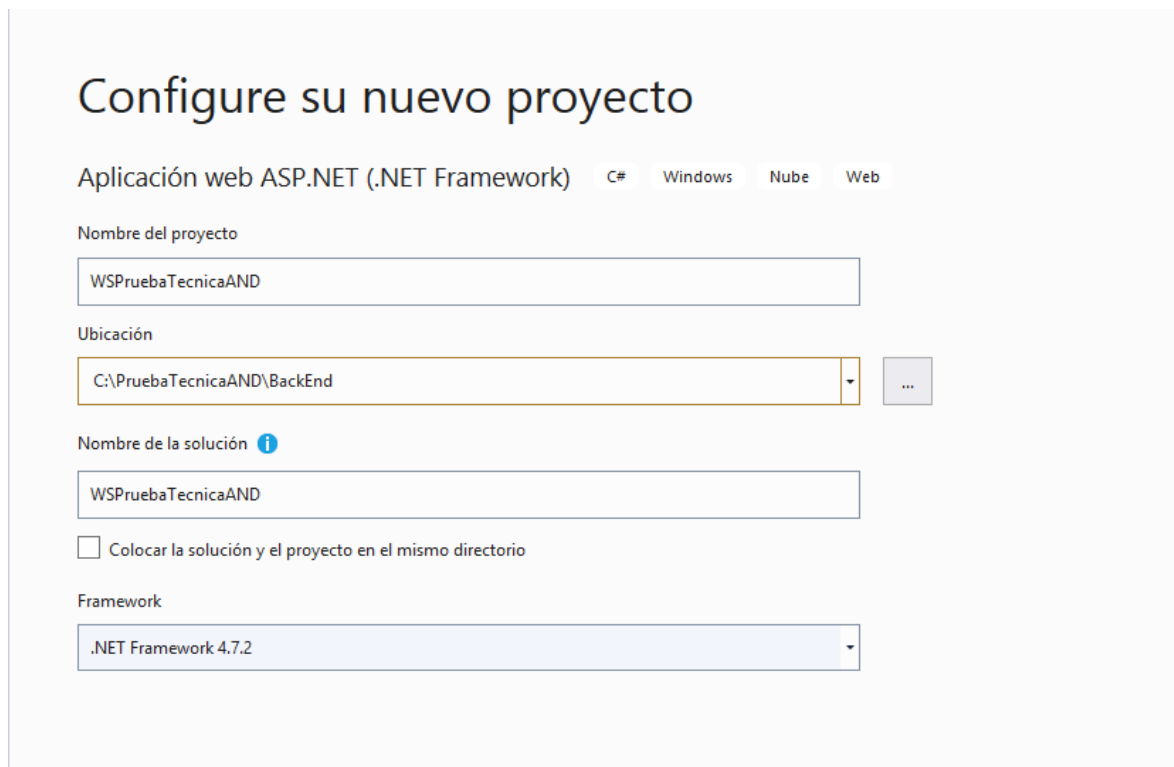
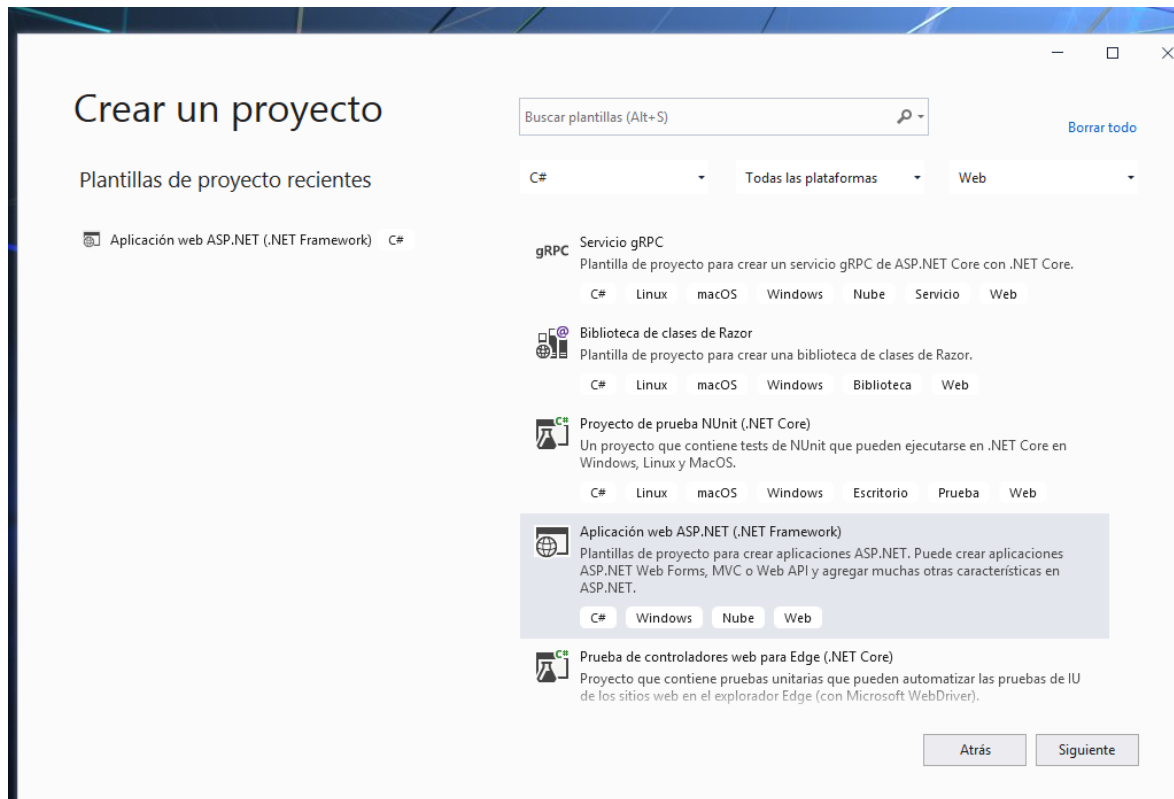
| PORTATILN.PruebaT....- dbo.OtrosTemas | | PORTATILN.Prueba...ND - dbo.Tramites | PORTATILN.PruebaT....- dbo.OtrosTemas | | PORTATILN.Prueba...D - dbo.Cuentanos | PORTATILN |
|---------------------------------------|------|-------------------------------------------------------|---------------------------------------------------|-----------------|--------------------------------------|-------------------------------------|
| | id | nombre | entidad | disponiblelinea | concosto | detalletramite |
| | 1 | Certificado de antecedentes de responsabilidad fiscal | Contraloria General de la Republica | True | False | Detalle del Tra... |
| | 2 | Legalización de documentos de educación superio... | Ministerio de Educación | True | False | Detalle del Tra... |
| | 3 | Adopción de un niño, niña o adolescente por perso... | Instituto Colombiano de Bienestar Familiar- IC... | False | False | Detalle del Tra... |
| | 4 | Agendamiento de citas para consultorio juridico | Ministerio de Justicia | True | False | Detalle del Tra... |
| ▶ | 5 | Certificados y constancia académicas | Servicio Nacional de Aprendizaje- SENA | True | False | Detalle del Trámite |
| * | NULL | NULL | NULL | NULL | NULL | NULL |

| PORTATILN.PruebaT...- dbo.OtrosTemas | | PORTATILN.Prueba...ND - dbo.Tramites | PORTATILN.PruebaT...- dbo.OtrosTemas | PORTATILN.Prueba...D - dbo.Cuentanos | PORTATILN.Prueba...AND - dbo.O |
|--------------------------------------|------|--------------------------------------|-------------------------------------------------------|---------------------------------------------------------------------------------|--------------------------------|
| | id | titulo | subtitulo | enlace | imagen |
| | 1 | Sabes que son los datos abiertos? | La información que producen la entidad... | https://www.datosabiertos.gov.co | DatosAbiertos.png |
| ▶ | 3 | Conoce más sobre nuestro país | y queremos que conozca más sobre ella | https://www.colombia.co | ColombiaCo.png |
| | 4 | Este portal está pensado para tí | GOV.CO nace para facilitarle a los ciud... | https://www.gov.co | GovCo.png |
| | 5 | Otra tema | Este es otro tema de interes | https://www.google.com | Google.png |
| • | NULL | NULL | NULL | NULL | NULL |

En la carpeta del Git, se incluye el script de esta base de datos.

Construcción del API

Se construyó un Servicio Web API denominado WSPuebaTecnicaAND en Visual Studio 2019



Crear una aplicación web ASP.NET



Vacío

Una plantilla de proyecto vacía para crear aplicaciones ASP.NET. Esta plantilla no tiene contenido.



Web Forms

Una plantilla de proyecto para crear aplicaciones de ASP.NET Web Forms. ASP.NET Web Forms le permite crear sitios web dinámicos con un modelo familiar controlado por eventos para arrastrar y colocar. Una superficie de diseño y cientos de controles y componentes le permiten crear rápidamente sofisticados y eficaces sitios controlados por la interfaz de usuario y con acceso a datos.



MVC

Una plantilla de proyecto para crear aplicaciones ASP.NET MVC. ASP.NET MVC permite compilar aplicaciones mediante la arquitectura de controlador de vista de modelos. ASP.NET MVC incluye muchas características que permiten un desarrollo rápido orientado a pruebas para crear aplicaciones que usan los últimos estándares.



API web

Plantilla de proyecto para crear servicios HTTP REST que pueden llegar a una amplia gama de clientes, como, por ejemplo, exploradores y dispositivos móviles.



Aplicación de página única

Una plantilla de proyectos para crear aplicaciones HTML5 atractivas controladas por JavaScript del lado cliente mediante ASP.NET Web API. Las aplicaciones de una sola página proporcionan una experiencia de usuario atractiva que incluye interacciones del lado cliente mediante HTML5, CSS3 y JavaScript.

Autenticación

Sin autenticación

[Cambiar](#)

Agregar carpetas y referencias principales

☐ Formularios Web Forms

☒ MVC

☒ API web

Avanzado

☒ Configurar para HTTPS

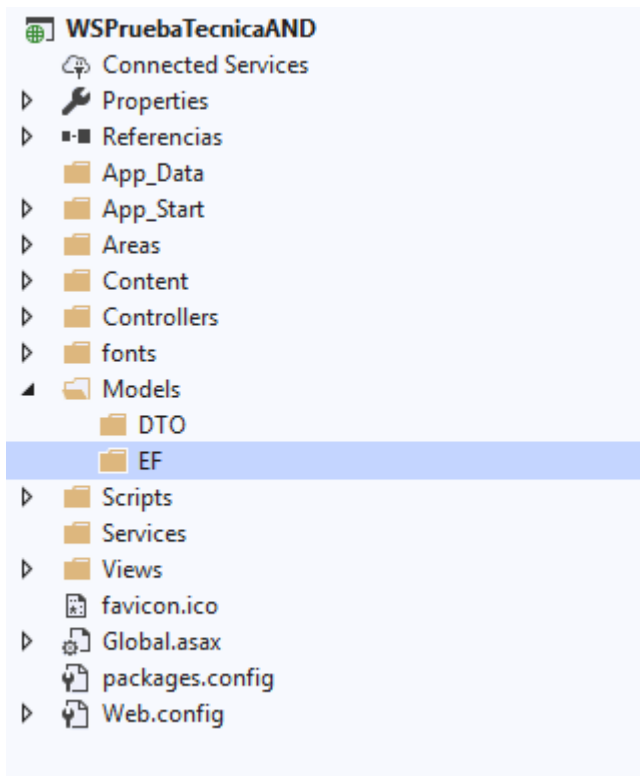
☐ Compatibilidad con Docker

(Requiere [Docker Desktop](#))

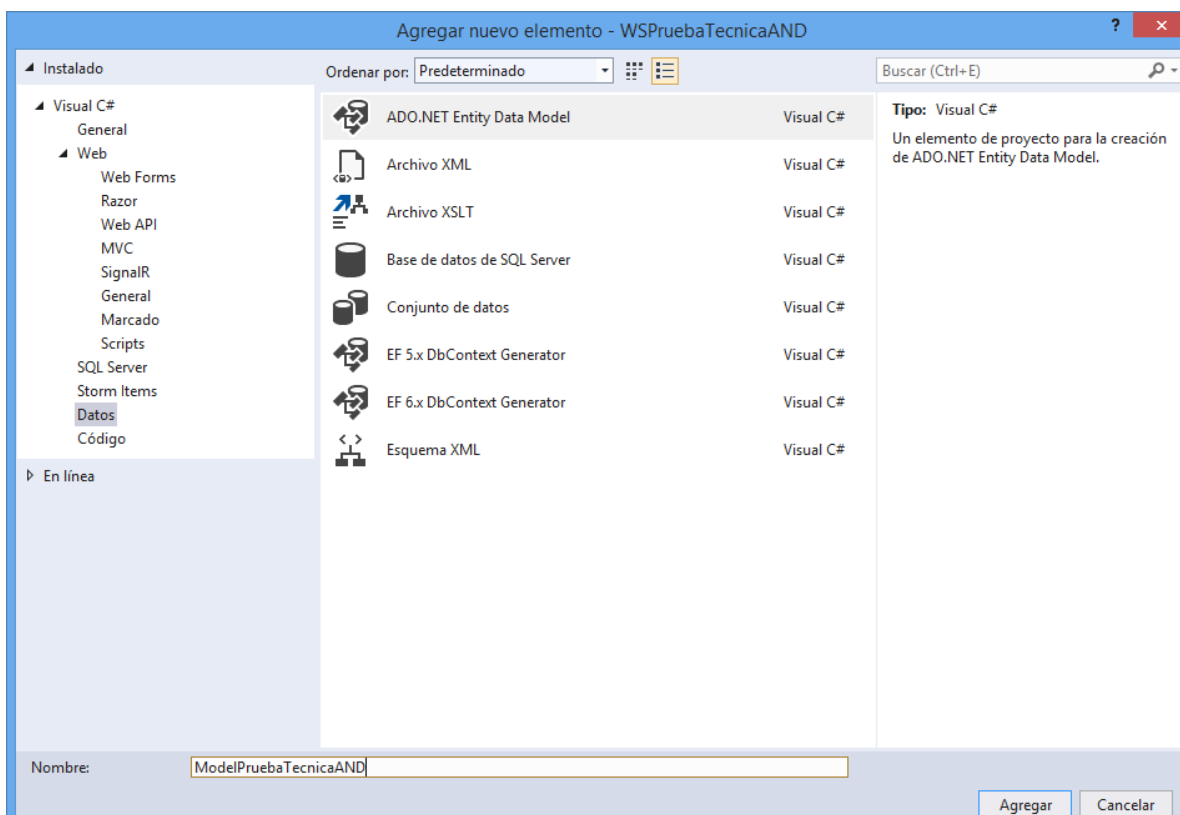
☐ Crear también un proyecto para pruebas unitarias

WSPruebaTecnicaAND.Tests

Dentro del proyecto del API se construyó una carpeta para alojar las clases para Services y dentro de la carpeta Models, se construyó una subcarpeta DTO y EF para el Entity Data Model



Se crea la conexión a la bd mediante el Entity Data Model



?

×

Propiedades de la conexión

Especifique la información para conectarse al origen de datos seleccionado o haga clic en "Cambiar" para elegir otro origen o proveedor de datos.

Origen de datos:

Microsoft SQL Server (SqlClient)

Cambiar...

Nombre del servidor:

PORTATILN

Actualizar

Conexión con el servidor

Autenticación: Autenticación de SQL Server

Nombre de usuario: sa

Contraseña:

☐ Guardar mi contraseña

Establecer conexión con una base de datos

☒ Seleccionar o escribir el nombre de la base de datos:

PruebaTecnicaAND

☐ Adjuntar un archivo de configuración de base de datos:

Nombre lógico:

Probar conexión

Aceptar

Cancelar

Microsoft Visual Studio

×

i

La prueba de conexión se realizó correctamente.

Aceptar

Asistente para Entity Data Model



Elegir la conexión de datos

¿Qué conexión de datos debe usar la aplicación para conectarse a la base de datos?

portatiln.PruebaTecnicaAND.dbo

Nueva conexión...

Esta cadena de conexión parece contener datos confidenciales (por ejemplo, una contraseña) que son necesarios para conectarse con la base de datos. Almacenar datos confidenciales en la cadena de conexión puede suponer un riesgo para la seguridad. ¿Desea incluir estos datos en la cadena de conexión?

- ☐ No, excluir datos confidenciales de la cadena de conexión. Los estableceré en el código de mi aplicación.
- ☒ Sí, incluir datos confidenciales en la cadena de conexión.

Cadena de conexión:

```
metadata=res://*/Models.EF.ModelPruebaTecnicaAND.csdl|
res://*/Models.EF.ModelPruebaTecnicaAND.ssdl|
res://*/Models.EF.ModelPruebaTecnicaAND.msl;provider=System.Data.SqlClient;provider connection
string="data source=PORTATILN;initial catalog=PruebaTecnicaAND;user
id=sa;password=*****;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Guardar configuración de conexión en Web.Config como:

PruebaTecnicaANDEntities


< Anterior

Siguiente >

Finalizar

Cancelar

Asistente para Entity Data Model

Elegir los objetos y la configuración de la base de datos

¿Qué objetos de la base de datos desea incluir en su modelo?

☒ Tablas

☒ dbo

☒ Cuentanos

☒ Opinion

☒ OtrosTemas

☒ Tramites

☐ Vistas

☐ Funciones y procedimientos almacenados

☐ Poner en plural o en singular los nombres de objeto generados

☒ Incluir columnas de clave externa en el modelo

☐ Importar procedimientos almacenados y funciones seleccionados en Entity Model

Espacio de nombres del modelo:

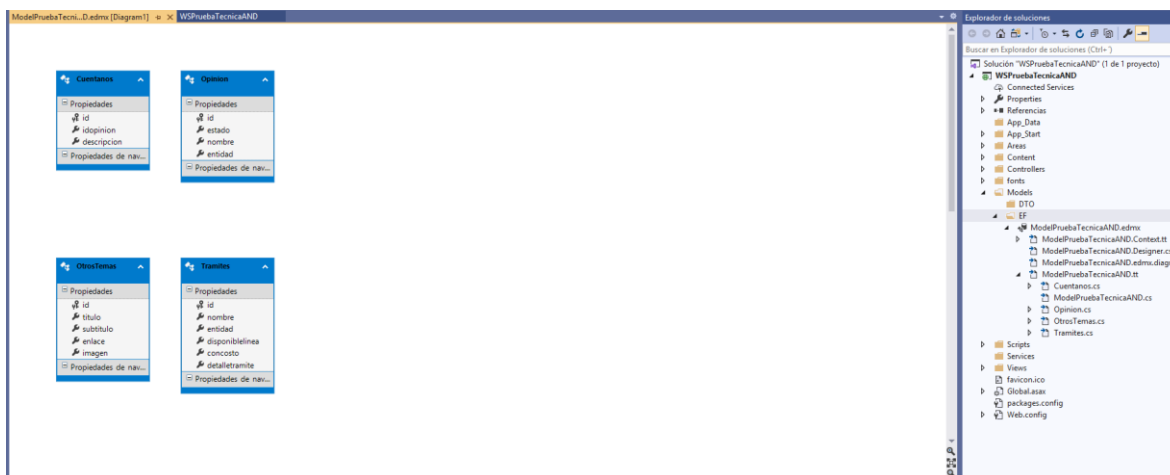
PruebaTecnicaANDModel

< Anterior

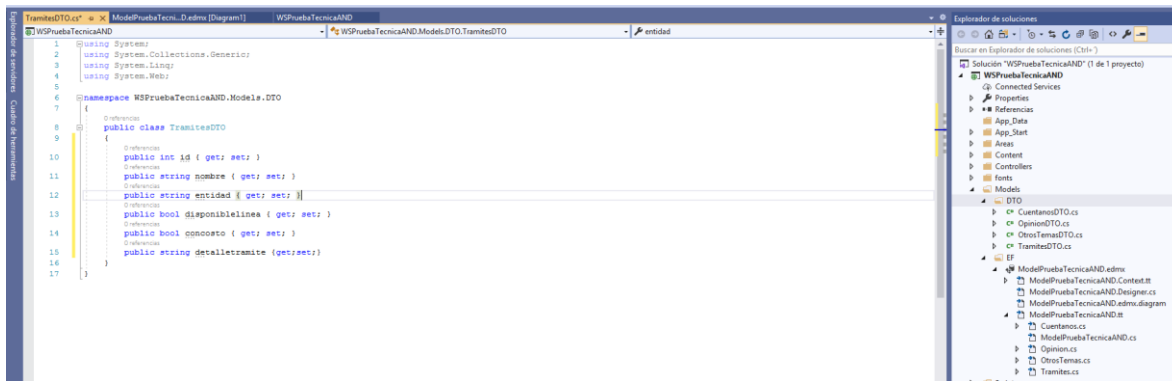
Siguiente >

Finalizar

Cancelar



Una vez creado el Entiy Data Model, se procede con la creación de los DTO correspondientes



Se construye la clase PruebaTecnicaService dentro de la subcarpeta Services (se realiza una sola con fines de agilizar el desarrollo de la prueba)

```

}

using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Security.Cryptography;
using System.Web;
using WSPruebaTecnicaAND.Models.DTO;
using WSPruebaTecnicaAND.Models.EF;
using System.Threading.Tasks;

using Newtonsoft.Json.Schema;

namespace WSPruebaTecnicaAND.Services
{
    public class PruebaTecnicaService
    {
        PruebaTecnicaANDEntities db = new PruebaTecnicaANDEntities();

        public async Task<ListaTramites> GetTramites()
        {
            ListaTramites respuesta = new ListaTramites();
            var lista = db.SP_LIST_TRAMITES().ToList();
            respuesta.Lista = lista;
            return respuesta;
        }

        public async Task<ListaOtrosTemas> GetOtrosTemas()
        {
            ListaOtrosTemas respuesta = new ListaOtrosTemas();
            var lista = db.SP_LIST_OTROSTEMAS().ToList();
            respuesta.Lista = lista;
            return respuesta;
        }

        public async Task<ListaOpinion> GetOpinion()
        {
            ListaOpinion respuesta = new ListaOpinion();
            var lista = db.SP_LIST_OPINION().ToList();
            respuesta.Lista = lista;
        }
    }
}

```

```

        return respuesta;
    }

    public async Task<ListaCuentanos> GetCuentanos()
    {
        ListaCuentanos respuesta = new ListaCuentanos();
        var lista = db.SP_LIST_CUENTANOS().ToList();
        respuesta.Lista = lista;
        return respuesta;
    }

    public CuentanosDTO addCuentanos(CuentanosDTO cuentanos)
    {
        CuentanosDTO respuesta = new CuentanosDTO();
        try
        {
            Cuentanos c = new Cuentanos { idopinion =
cuentanos.idopinion, descripcion = cuentanos.descripcion };

            db.Cuentanos.Add(c);
            db.SaveChanges();
            respuesta.OperacionExitosa = true;

        }
        catch (Exception ex)
        {
            respuesta.Mensaje = "Error al Ingresar datos en Cuentanos -"
+ ex.ToString() + "-";
            respuesta.OperacionExitosa = false;
        }
        return respuesta;
    }

}
}

```

Se construye el archivo PruebaTecnicaController

```

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Web.Http;
using System.Web.Http.Description;
using WSPruebaTecnicaAND.Models.DTO;
using WSPruebaTecnicaAND.Services;
using System.Net.Http;
using System.Net;
using System.Web.Configuration;
using System.IO;
using System.Web;
using System.Configuration;
using WSPruebaTecnicaAND.Models.EF;
using static WSPruebaTecnicaAND.Services.PruebaTecnicaService;

namespace WSPruebaTecnicaAND.Controllers
{
    [RoutePrefix("api/pruebatecnica")]
}

```

```

public class PruebaTecnicaController : ApiController
{

    private PruebaTecnicaService srv;
    private string keyToken = "97221cdc42-8661-4ab9-a04e-
51785baa88da39";
    public PruebaTecnicaController()
    {
        srv = new PruebaTecnicaService();
    }

    //Obtener Lista Tramites
    [Route("GetTramites")]
    [HttpGet]
    [ResponseType(typeof(ListaTramites))]
    public async Task<IHttpActionResult> GetTramites(string token)
    {
        try
        {
            if (token.Equals(keyToken))
            {
                var respuesta = srv.GetTramites();
                return Ok(respuesta);
            }
            else
                return null;
        }
        catch (Exception e)
        {
            string error = "Error desde el servicio GetTramites" +
e.Message;
            return Ok(error);
        }
    }

    //Obtener Lista Otros Temas
    [Route("GetOtrosTemas")]
    [HttpGet]
    [ResponseType(typeof(ListaOtrosTemas))]
    public async Task<IHttpActionResult> GetOtrosTemas(string token)
    {
        try
        {
            if (token.Equals(keyToken))
            {
                var respuesta = srv.GetOtrosTemas();
                return Ok(respuesta);
            }
            else
                return null;
        }
        catch (Exception e)
        {
            string error = "Error desde el servicio GetOtrosTemas" +
e.Message;
            return Ok(error);
        }
    }
}

```

```

    }
}

//Obtener Lista Opiniones
[Route("GetOpinion")]
[HttpGet]
[ResponseType(typeof(ListaOpinion))]
public async Task<IHttpActionResult> GetOpinion(string token)
{
    try
    {
        if (token.Equals(keyToken))
        {
            var respuesta = srv.GetOpinion();
            return Ok(respuesta);
        }
        else
            return null;
    }
    catch (Exception e)
    {
        string error = "Error desde el servicio GetOpinion" +
e.Message;
        return Ok(error);
    }
}

//Obtener Lista Cuentanos
[Route("GetCuentanos")]
[HttpGet]
[ResponseType(typeof(CuentanosDTO))]
public async Task<IHttpActionResult> GetCuentanos(string token)
{
    try
    {
        if (token.Equals(keyToken))
        {
            var respuesta = srv.GetCuentanos();
            return Ok(respuesta);
        }
        else
            return null;
    }
    catch (Exception e)
    {
        string error = "Error desde el servicio GetTramites" +
e.Message;
        return Ok(error);
    }
}

[Route("addCuentanos")]
[HttpPost]

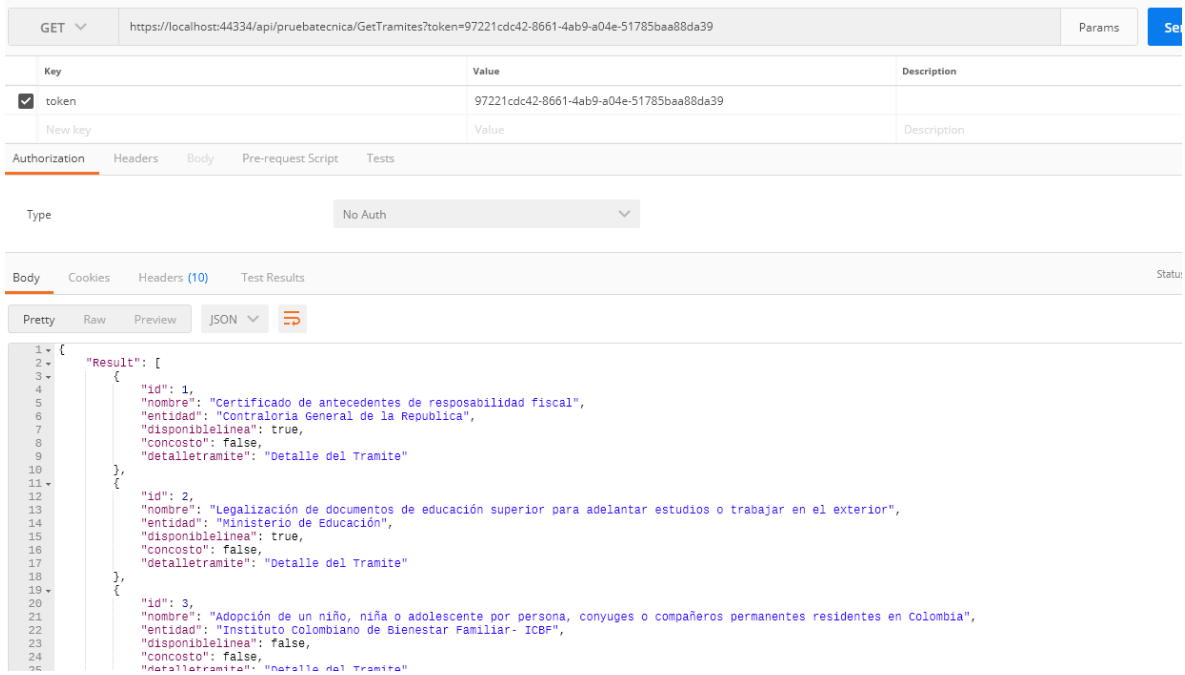
```

```

[ResponseType(typeof(RespuestaBase))]
public IHttpActionResult addCuentanos(string token, CuentanosDTO
cuentanos)
{
    try
    {
        if (token.Equals(keyToken))
        {
            var respuesta = srv.addCuentanos(cuentanos);
            return Ok(respuesta);
        }
        else
            return null;
    }
    catch (Exception e)
    {
        string error = "Error desde el servicio addCuentanos" +
e.Message;
        return Ok(error);
    }
}
}
}

```

Se prueba que el servicio devuelve los datos en formato JSON



GET https://localhost:44334/api/pruebatecnica/GetTramites?token=97221cdc42-8661-4ab9-a04e-51785baa88da39

| Key | Value | Description |
|-------------------------------------------|------------------------------------------|-------------|
| <input checked="" type="checkbox"/> token | 97221cdc42-8661-4ab9-a04e-51785baa88da39 | |
| New key | Value | Description |

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (10) Test Results

Pretty Raw Preview JSON

```

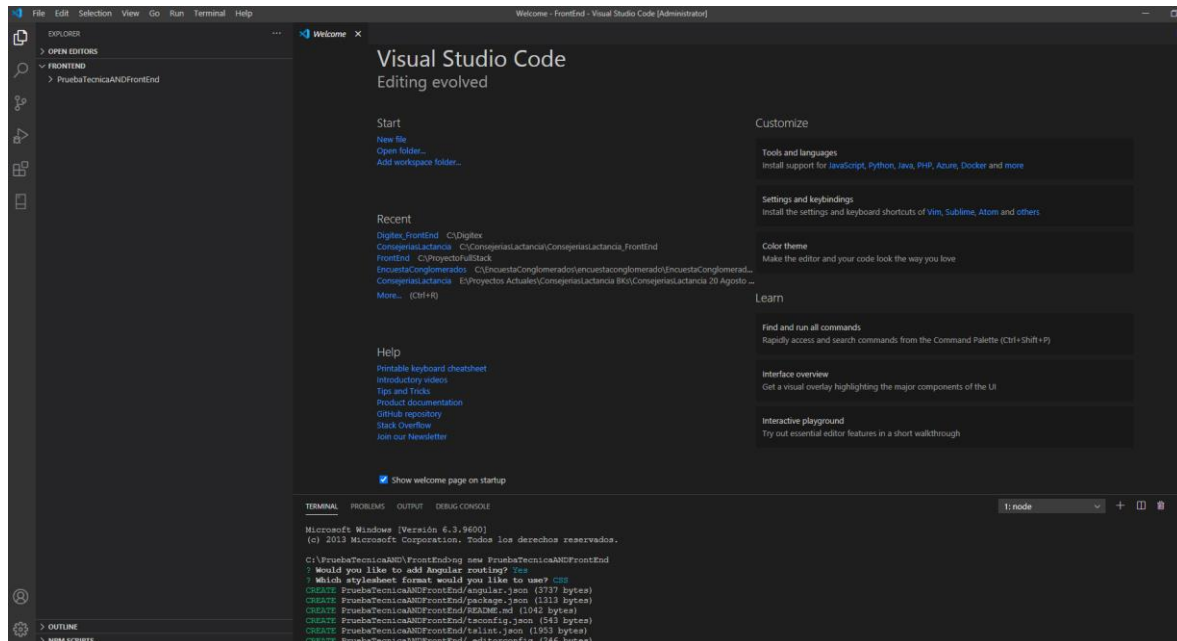
1 {
2   "Result": [
3     {
4       "id": 1,
5       "nombre": "Certificado de antecedentes de responsabilidad fiscal",
6       "entidad": "Contraloría General de la República",
7       "disponiblelinea": true,
8       "concosto": false,
9       "detalletramite": "Detalle del Tramite"
10    },
11    {
12      "id": 2,
13      "nombre": "Legalización de documentos de educación superior para adelantar estudios o trabajar en el exterior",
14      "entidad": "Ministerio de Educación",
15      "disponiblelinea": true,
16      "concosto": false,
17      "detalletramite": "Detalle del Tramite"
18    },
19    {
20      "id": 3,
21      "nombre": "Adopción de un niño, niña o adolescente por persona, conyuges o compañeros permanentes residentes en Colombia",
22      "entidad": "Instituto Colombiano de Bienestar Familiar- ICBF",
23      "disponiblelinea": false,
24      "concosto": false,
25      "detalletramite": "Detalle del Tramite"
26    }
27  ]
28 }

```

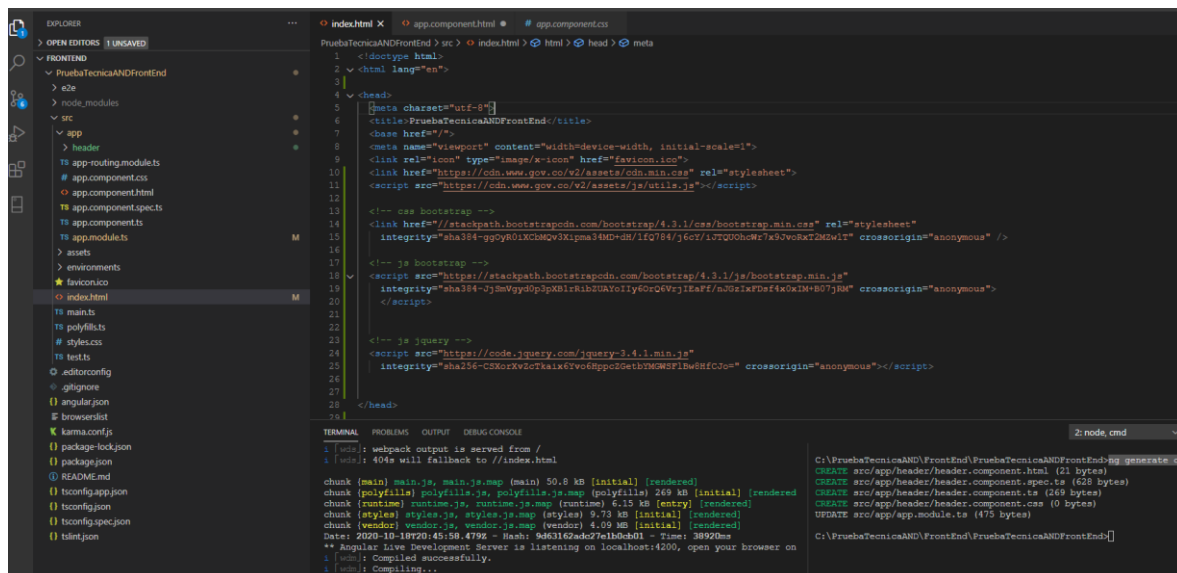
Construcción del FrontEnd

Utilizando Visual Studio Code, se creó un nuevo proyecto en la carpeta
C:\PruebaTecnicaAND\FrontEnd

Mediante el comando ng new PruebaTecnicaFrontEnd



En el index, se registra el llamado a Bootstrap y JQuery, como está indicado en el CDN de Gov.co



Se crea el componente ptheadr

ng generate component ptheadr

y se implementa con el código indicado en el CDN de Gov.co

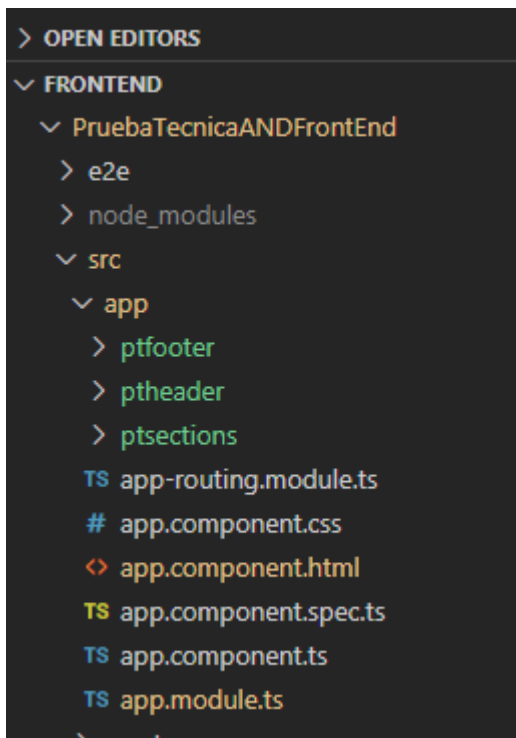
Se crea el componente ptfooter

ng generate component ptfooter

y se implementa con el código indicado en el CDN de Gov.co

Se crea el componente ptections

ng generate component ptsections



Se registran los components en el app.module.ts


```

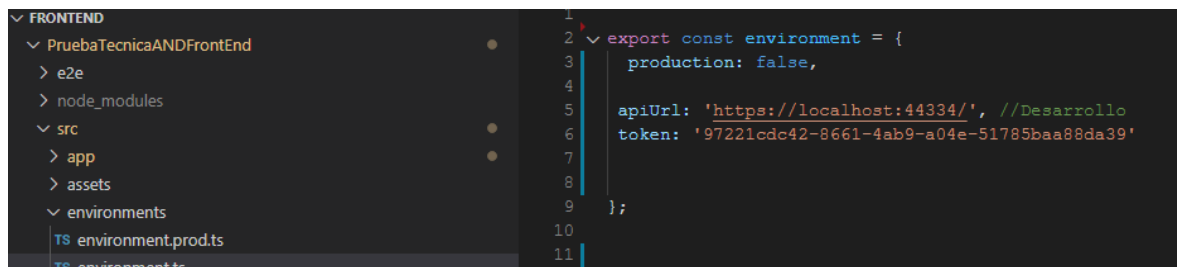
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { PtheadComponent } from './pthead/pthead.component';
import { PtfooterComponent } from './ptfooter/ptfooter.component';
import { PtsectionsComponent } from './ptsections/ptsections.component';
import { PttramitesComponent } from './pttramites/pttramites.component';
import { PtopinionComponent } from './ptopinion/ptopinion.component';
import { PtcuentanosComponent } from './ptcuentanos/ptcuentanos.component';
import { PtotrostemasComponent } from './ptotrostemas/ptotrostemas.component';
import { ptService } from './services/ptservice';

@NgModule({
  declarations: [
    AppComponent,
    PtheadComponent,
    PtfooterComponent,
    PtsectionsComponent,
    PttramitesComponent,
    PtopinionComponent,
    PtcuentanosComponent,
    PtotrostemasComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [ptService],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Se construyen 4 componentes más para cada una de las secciones : ptTramites, ptOpinion, ptCuentanos, ptOtrosTemas

Se configura en los archivos de environment, la conexión al servidor del API y el token



```
1
2 export const environment = {
3   production: false,
4
5   apiUrl: 'https://localhost:44334/', //Desarrollo
6   token: '97221cdc42-8661-4ab9-a04e-51785baa88da39'
7 };
8
9
10
11
```

Ahora se crea una carpeta Models y un archivo llamado ptmodels.ts, que contiene los atributos extraídos desde el API

```
export class Tramites {
  constructor(
    public id: number,
    public nombre: string,
    public entidad : string,
    public disponible linea : boolean,
    public concosto : boolean,
    public detalletramite : string
  ) { }
}

export class OtrosTemas {
  constructor(
    public id: number,
    public titulo: string,
    public subtítulo : string,
    public enlace : boolean,
    public imagen : boolean
  ) { }
}

export class Opinion {
  constructor(
    public id: number,
    public estado: number,
    public nombre : string,
    public entidad : string
  ) { }
}
```

```

export class Cuentanos {
  constructor(
    public id: number,
    public idopinion: number,
    public descripcion : string
  ) { }
}

```

Se crea una carpeta Services y el archivo ptService.ts, que contiene los llamados al API

```

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from "rxjs";
import { environment } from 'src/environments/environment';
import { Cuentanos, Opinion, OtrosTemas, Tramites } from '../models/ptmodels';

@Injectable({
  providedIn: 'root'
})
export class ptService {

  private url: string;
  private token: string;

  constructor(
    private httpClient: HttpClient) {
    this.url = environment.apiUrl;
    this.token = environment.token;
  }

  GetTramites(): Observable<any> {
    return this.httpClient.get<Tramites[]>(this.url + '/api/pruebatecnica/Ge
tTramites?token=' + this.token);
  }
}

```

```

    GetOtrosTemas(): Observable<any> {
        return this.httpClient.get<OtrosTemas[]>(this.url + '/api/pruebatecnica/GetOtrosTemas?token=' + this.token);
    }

    GetOpinion(): Observable<any> {
        return this.httpClient.get<Opinion[]>(this.url + '/api/pruebatecnica/GetOpinion?token=' + this.token);
    }

    GetCuentanos(): Observable<any> {
        return this.httpClient.get<Cuentanos[]>(this.url + '/api/pruebatecnica/GetCuentanos?token=' + this.token);
    }

    addCuentanos(data: Cuentanos): any {
        return this.httpClient.post<any>(this.url + '/encuesta/parametricas/addCuentanos?token=' + this.token, data);
    }
}

```

Construcción de los componentes

Teniendo en cuenta los elementos del CDN de Gov.co, se construyen los elementos visuales.

Ver proyecto en repositorio Git

Se adjunta igualmente un Video de la implementación de esta solución