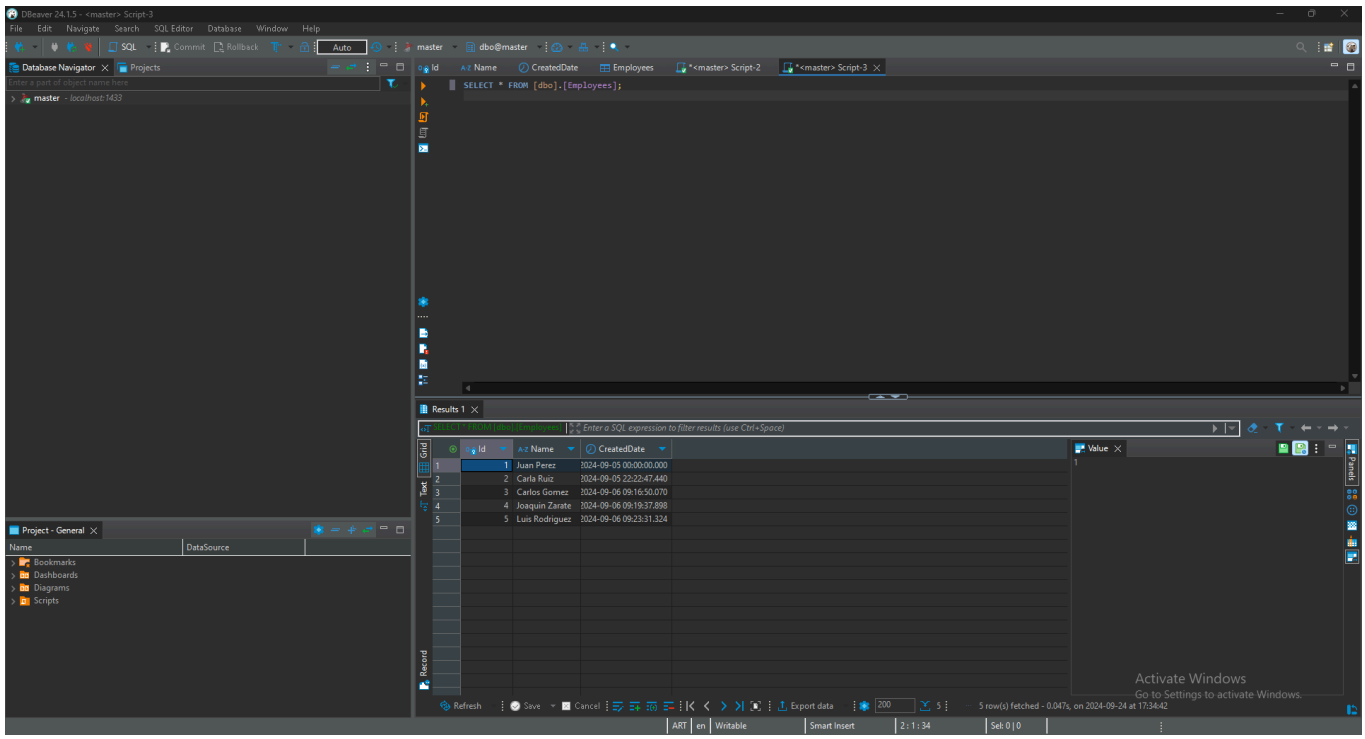
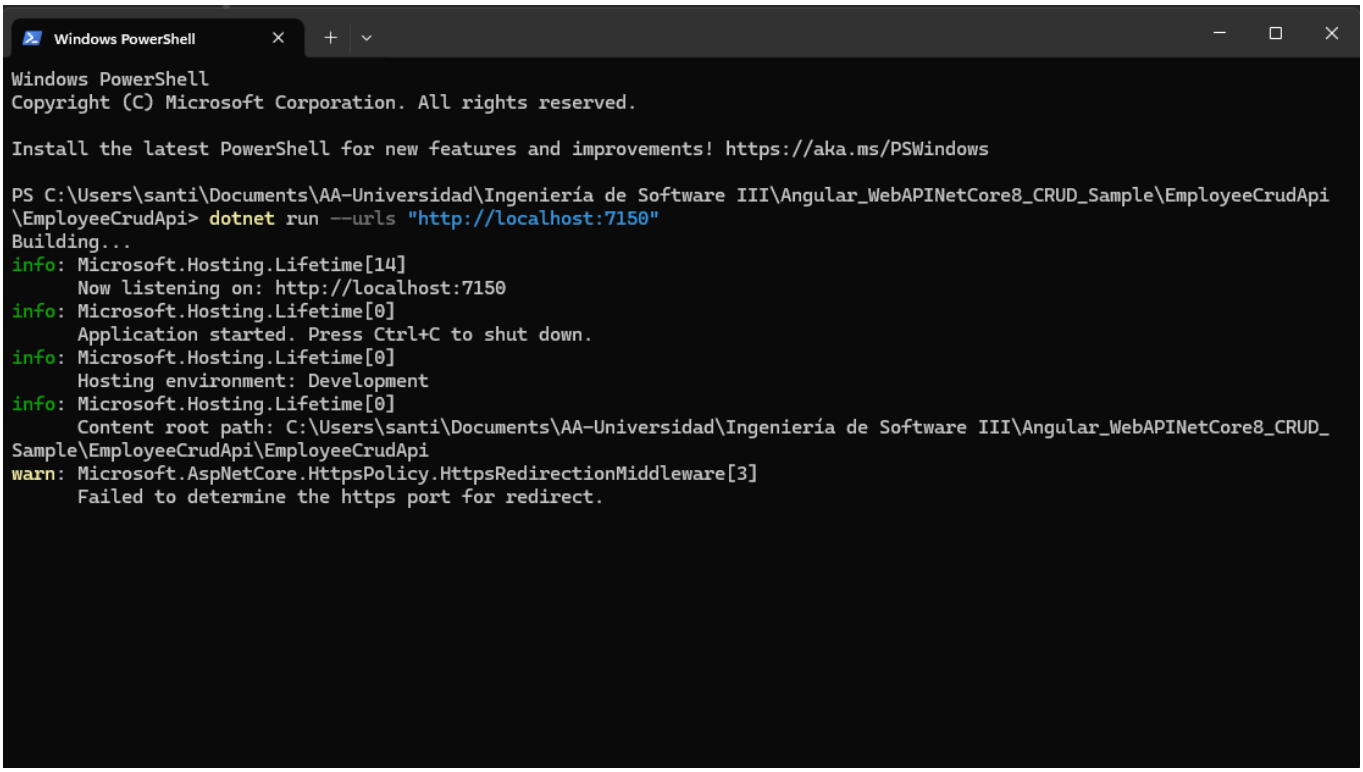
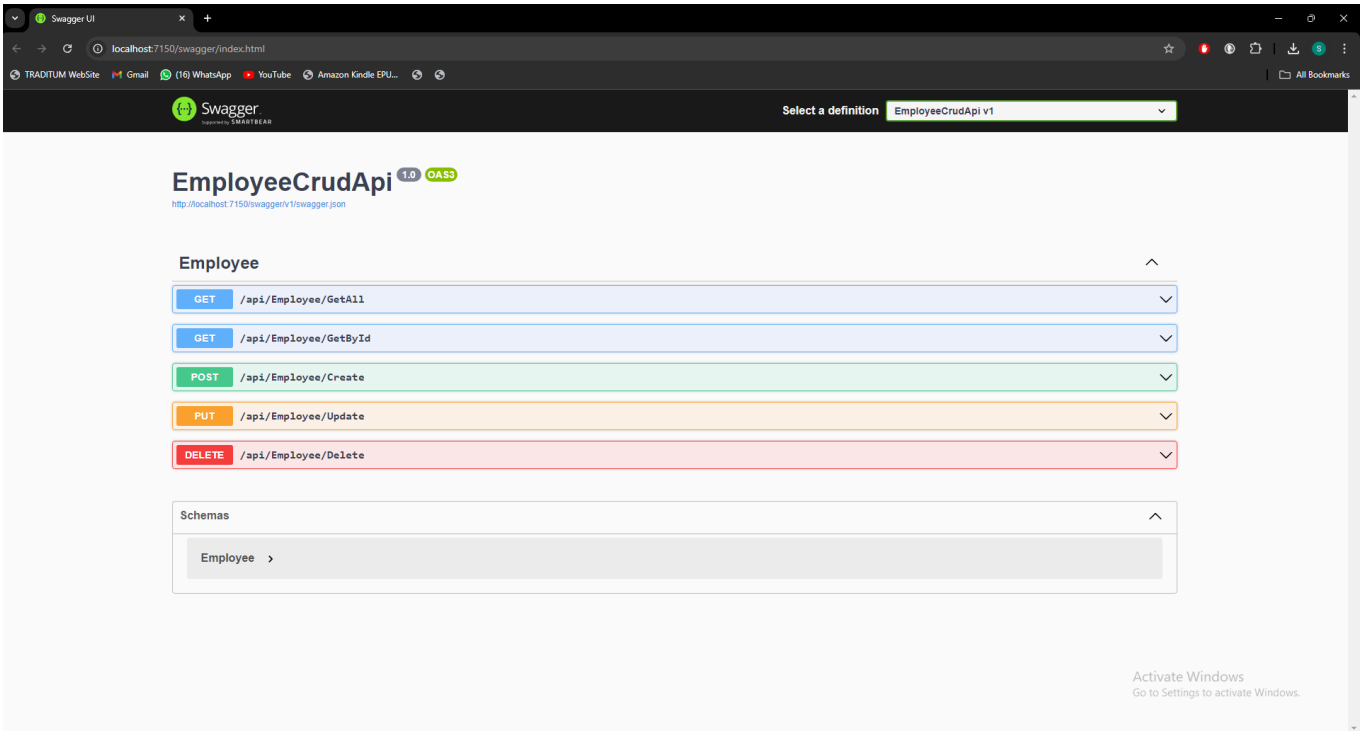
TP6

4.1 Creación de una BD SQL Server para nuestra App



4.2 Obtener nuestra App


Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi\EmployeeCrudApi> dotnet run --urls "http://localhost:7150"
Building...
info: Microsoft.Hosting.Lifetime[14]
Now listening on: http://localhost:7150
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
Content root path: C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi\EmployeeCrudApi
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
Failed to determine the https port for redirect.

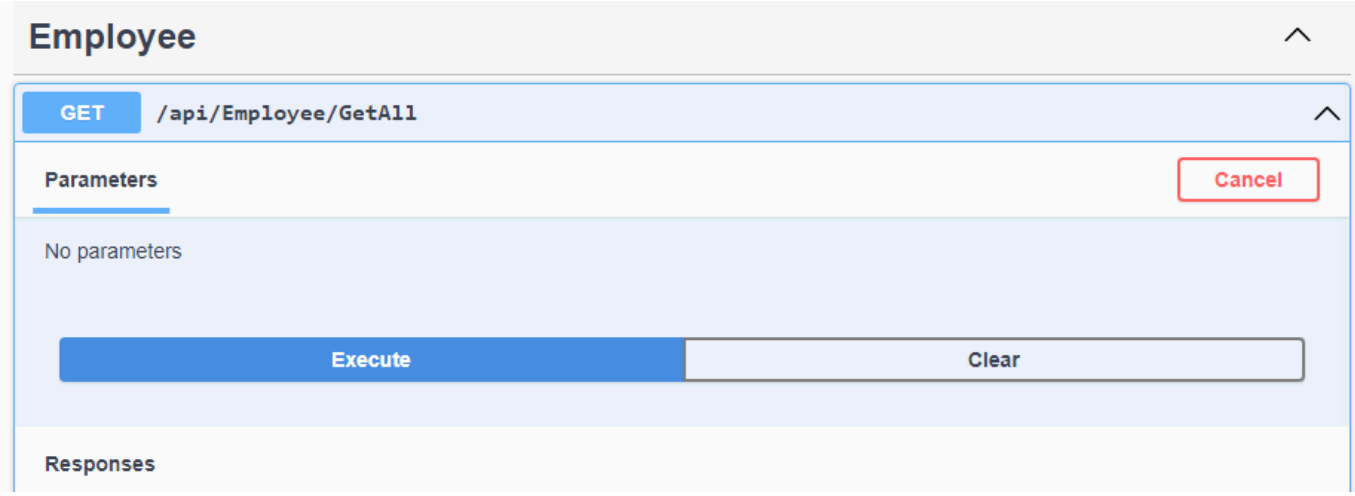

Swagger UI
localhost:7150/swagger/index.html
Swagger
Select a definition
EmployeeCrudApi v1
EmployeeCrudApi
1.0
OAS3
http://localhost:7150/swagger/v1/swagger.json
Employee
GET /api/Employee/GetAll
GET /api/Employee/GetById
POST /api/Employee/Create
PUT /api/Employee/Update
DELETE /api/Employee/Delete
Schemas
Employee
Activate Windows
Go to Settings to activate Windows.

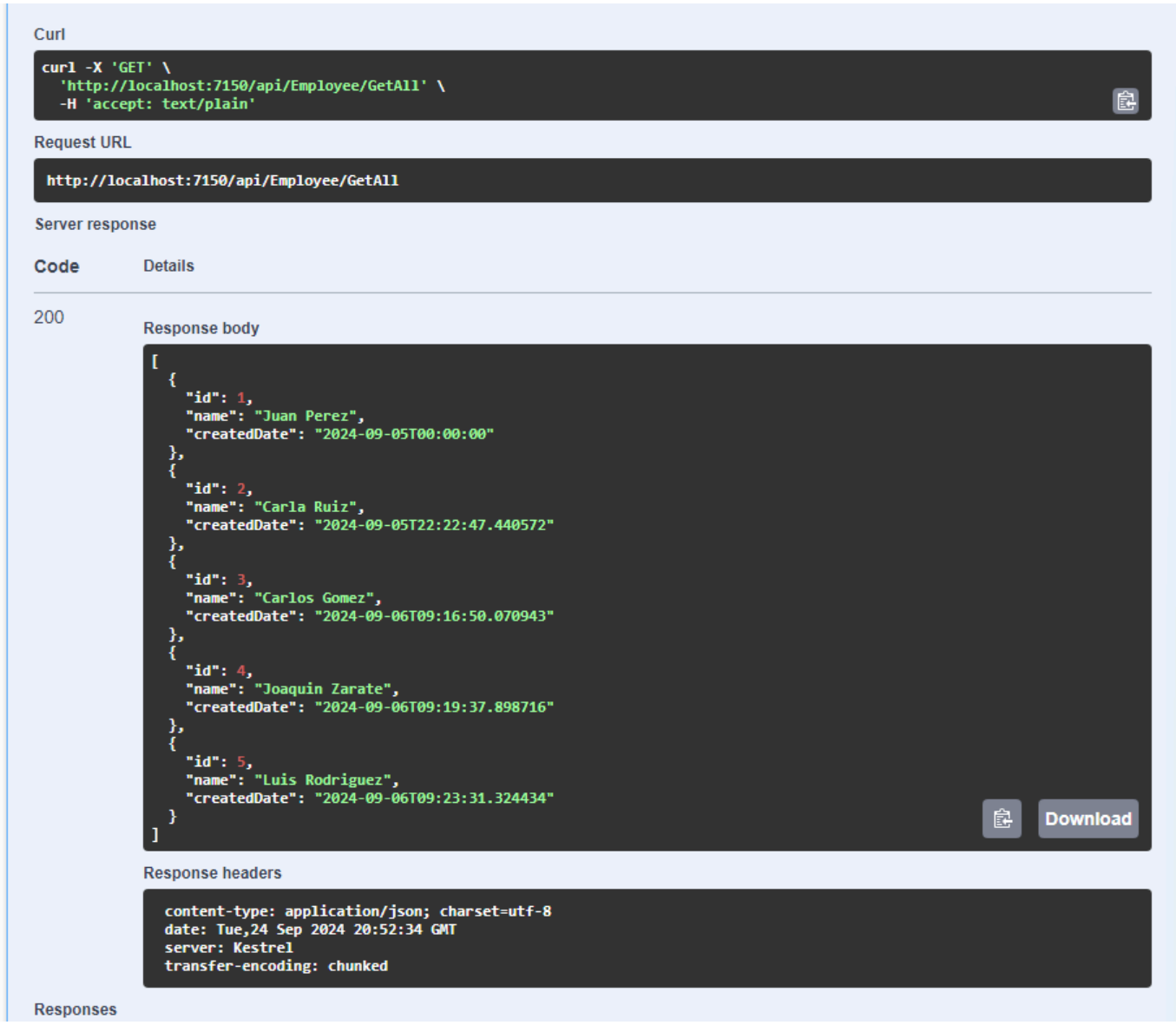
EmployeeCrudAngular

Employees: Type to search... Add New Employee

Id	Name	Created Date
1	Juan Perez	05/09/2024 00:00:00
2	Carla Ruiz	05/09/2024 22:22:47
3	Carlos Gomez	06/09/2024 09:16:50
4	Joaquin Zarate	06/09/2024 09:19:37
5	Luis Rodriguez	06/09/2024 09:23:31

PRUEBA EMPLOYEE CRUD API:





4.3 Crear Pruebas Unitarias para nuestra API

```
  Determining projects to restore...
  Restored C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests\EmployeeCrudApi.Tests.csproj (in
   1.57 sec).
  1 of 2 projects are up-to-date for restore.
  EmployeeCrudApi -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi\EmployeeCrudApi\bin\Debug\ne
  t8.0\EmployeeCrudApi.dll
  EmployeeCrudApi.Tests -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests\bin\Debug\net8.0
  \EmployeeCrudApi.Tests.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:03.85
PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests> dotnet test
  Determining projects to restore...
  All projects are up-to-date for restore.
  EmployeeCrudApi -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi\EmployeeCrudApi\bin\Debug\ne
  t8.0\EmployeeCrudApi.dll
  EmployeeCrudApi.Tests -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests\bin\Debug\net8.0
  \EmployeeCrudApi.Tests.dll
Test run for C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests\bin\Debug\net8.0\EmployeeCrudAp
i.Tests.dll (.NETCoreApp,Version=v8.0)
VSTest version 17.11.0 (x64)

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed!  - Failed:     0, Passed:     5, Skipped:     0, Total:     5, Duration: 98 ms - EmployeeCrudApi.Tests.dll (net8.0)
```

I)

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost,1433;Initial Catalog=master;Persist Security Info=False;User ID=SA;Password=passwordincorrecta;MultipleActiveResultSets=False;Encrypt=False;TrustServerCertificate=True;Conne
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

J)

Curl

```
curl -X 'GET' \
  'http://localhost:7150/api/Employee/GetAll' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:7150/api/Employee/GetAll
```

Server response

Code	Details
500 *Undocumented*	Error: Internal Server Error

Response body

```
Microsoft.Data.SqlClient.SqlException (0x80131904): Login failed for user 'SA'.
   at Microsoft.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at Microsoft.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
   at Microsoft.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateOb
j, Boolean& dataReady)
   at Microsoft.Data.SqlClient.TdsParser.Run(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj)
   at Microsoft.Data.SqlClient.SqlInternalConnectionTds.CompleteLogin(Boolean enlistOK)
   at Microsoft.Data.SqlClient.SqlInternalConnectionTds.AttemptOneLogin(ServerInfo serverInfo, String newPassword, SecureString newSecurePassword, Boolean ignoreSniOpenTimeout, TimeoutTimer t
imeout, Boolean withFailover)
   at Microsoft.Data.SqlClient.SqlInternalConnectionTds.LoginNoFailover(ServerInfo serverInfo, String newPassword, SecureString newSecurePassword, Boolean redirectedUserInstance, SqlConnectio
nString connectionOptions, SqlCredential credential, TimeoutTimer timeout)
   at Microsoft.Data.SqlClient.SqlInternalConnectionTds.OpenLoginEnlist(TimeoutTimer timeout, SqlConnectionString connectionOptions, SqlCredential credential, String newPassword, SecureString
newSecurePassword, Boolean redirectedUserInstance)
   at Microsoft.Data.SqlClient.SqlInternalConnectionTds..ctor(DbConnectionPoolIdentity identity, SqlConnectionString connectionOptions, SqlCredential credential, Object providerInfo, String n
ewPassword, SecureString newSecurePassword, Boolean redirectedUserInstance, SqlConnectionString userConnectionOptions, SessionData reconnectSessionData, Boolean applyTransientFaultHandling, S
tring accessToken, DbConnectionPool pool)
   at Microsoft.Data.SqlClient.SqlConnectionFactory.CreateConnection(DbConnectionOptions options, DbConnectionPoolKey poolKey, Object poolGroupProviderInfo, DbConnectionPool pool, DbConnectio
n owningConnection, DbConnectionOptions userOptions)
   at Microsoft.Data.ProviderBase.DbConnectionFactory.CreatePooledConnection(DbConnectionPool pool, DbConnection owningObject, DbConnectionOptions options, DbConnectionPoolKey poolKey, DbConn
ectionOptions userOptions)
   at Microsoft.Data.ProviderBase.DbConnectionPool.CreateObject(DbConnection owningObject, DbConnectionOptions userOptions, DbConnectionInternal oldConnection)
   at Microsoft.Data.ProviderBase.DbConnectionPool.UserCreateRequest(DbConnection owningObject, DbConnectionOptions userOptions, DbConnectionInternal oldConnection)
   at Microsoft.Data.ProviderBase.DbConnectionPool.TryGetConnection(DbConnection owningObject, UInt32 waitForMultipleObjectsTimeout, Boolean allowCreate, Boolean onlyOneCheckConnection, DbCon
nectionOptions userOptions, DbConnectionInternal& connection)
   at Microsoft.Data.ProviderBase.DbConnectionPool.WaitForPendingOpen()
--- End of stack trace from previous location ---
   at Microsoft.EntityFrameworkCore.Storage.RelationalConnection.OpenInternalAsync(Boolean errorsExpected, CancellationToken cancellationToken)
   at Microsoft.EntityFrameworkCore.Storage.RelationalConnection.OpenInternalAsync(Boolean errorsExpected, CancellationToken cancellationToken)
```

Download

Response headers

```
content-type: text/plain; charset=utf-8
date: Tue,24 Sep 2024 21:04:24 GMT
server: Kestrel
transfer-encoding: chunked
```

L)

```
PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests> dotnet build
  Determining projects to restore...
  All projects are up-to-date for restore.
  EmployeeCrudApi -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi\EmployeeCrudApi\bin\Debug\ne
  t8.0\EmployeeCrudApi.dll
  EmployeeCrudApi.Tests -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests\bin\Debug\net8.0
  \EmployeeCrudApi.Tests.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:01.23
PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests> dotnet test
  Determining projects to restore...
  All projects are up-to-date for restore.
  EmployeeCrudApi -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi\EmployeeCrudApi\bin\Debug\ne
  t8.0\EmployeeCrudApi.dll
  EmployeeCrudApi.Tests -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests\bin\Debug\net8.0
  \EmployeeCrudApi.Tests.dll
Test run for C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi.Tests\bin\Debug\net8.0\EmployeeCrudAp
i.Tests.dll (.NETCoreApp,Version=v8.0)
VSTest version 17.11.0 (x64)

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed!  - Failed:     0, Passed:     5, Skipped:     0, Total:     5, Duration: 89 ms - EmployeeCrudApi.Tests.dll (net8.0)
```

N)

Curl

```
curl -X 'GET' \
  'http://localhost:7150/api/Employee/GetAll' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:7150/api/Employee/GetAll
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": 1,
    "name": "Juan Perez",
    "createdDate": "2024-09-05T00:00:00"
  },
  {
    "id": 2,
    "name": "Carla Ruiz",
    "createdDate": "2024-09-05T22:22:47.440572"
  },
  {
    "id": 3,
    "name": "Carlos Gomez",
    "createdDate": "2024-09-06T09:16:50.070943"
  },
  {
    "id": 4,
    "name": "Joaquin Zarate",
    "createdDate": "2024-09-06T09:19:37.898716"
  },
  {
    "id": 5,
    "name": "Luis Rodriguez",
    "createdDate": "2024-09-06T09:23:31.324434"
  }
]
```


Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,24 Sep 2024 21:07:00 GMT
server: Kestrel
transfer-encoding: chunked
```

4.4 Creamos pruebas unitarias para nuestro front de Angular:

B)

C: > Users > santi > Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > src > app > TS app.component.spec.ts > ...

```typescript
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component'; // Ajusta la ruta si es necesario

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [AppComponent], // Usa imports en lugar de declarations
    }).compileComponents();
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('h1')?.textContent).toContain('EmployeeCrudAngular');
  });

});
```

C)

C: > Users > santi > Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > src > app > TS employee.service.spec.ts > ...

```typescript
import { TestBed } from '@angular/core/testing';
import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';
import { EmployeeService } from './employee.service';
import { Employee } from './employee.model';
import { DatePipe } from '@angular/common';

describe('EmployeeService', () => {
  let service: EmployeeService;
  let httpMock: HttpTestingController;
  let datePipe: DatePipe;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [
        EmployeeService,
        DatePipe
      ]
    });

    service = TestBed.inject(EmployeeService);
    httpMock = TestBed.inject(HttpTestingController);
    datePipe = TestBed.inject(DatePipe);
  });

  afterEach(() => {
    httpMock.verify();
  });



  it('should retrieve all employees', () => {
    const today = new Date();
    const expectedDateTime = datePipe.transform(today, 'dd/MM/yyyy HH:mm:ss', undefined) ?? '';  // Consistente con el servicio

    const dummyEmployees: Employee[] = [
      new Employee(1, 'John Doe', expectedDateTime),
      new Employee(2, 'Jane Smith', expectedDateTime)
    ];
```

D)

C: > Users > santi > Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > src > app > employee > TS employee.component.spec.ts > ...

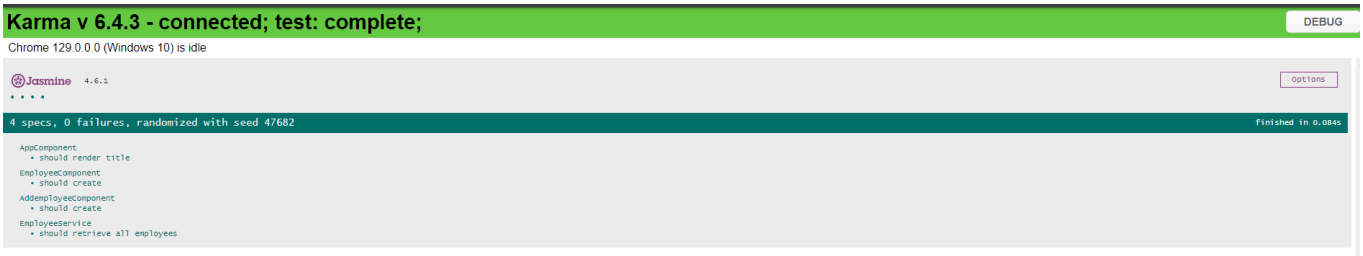
```typescript
import { TestBed } from '@angular/core/testing';
import { EmployeeComponent } from './employee.component';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { DatePipe } from '@angular/common';

describe('EmployeeComponent', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [EmployeeComponent, HttpClientTestingModule],
      providers: [DatePipe] // Añade DatePipe a los proveedores
    });
  });

  it('should create', () => {
    const fixture = TestBed.createComponent(EmployeeComponent);
    const component = fixture.componentInstance;
    expect(component).toBeTruthy();
  });
});
```

E)

C: > Users > santi > Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > src > app > addemployee > TS addemployee.component.spec.ts > ...

```ts
import { TestBed } from '@angular/core/testing';
import { AddemployeeComponent } from './addemployee.component';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { ActivatedRoute } from '@angular/router';
import { of } from 'rxjs'; // para simular observables
import { DatePipe } from '@angular/common';

describe('AddemployeeComponent', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [AddemployeeComponent, HttpClientTestingModule],
      providers: [
        DatePipe,
        {
          provide: ActivatedRoute, // Simula ActivatedRoute
          useValue: {
            params: of({ id: 1 }) // simula el parámetro id en la URL
          }
        }
      ]
    });
  });

  it('should create', () => {
    const fixture = TestBed.createComponent(AddemployeeComponent);
    const component = fixture.componentInstance;
    expect(component).toBeTruthy();
  });
});
```

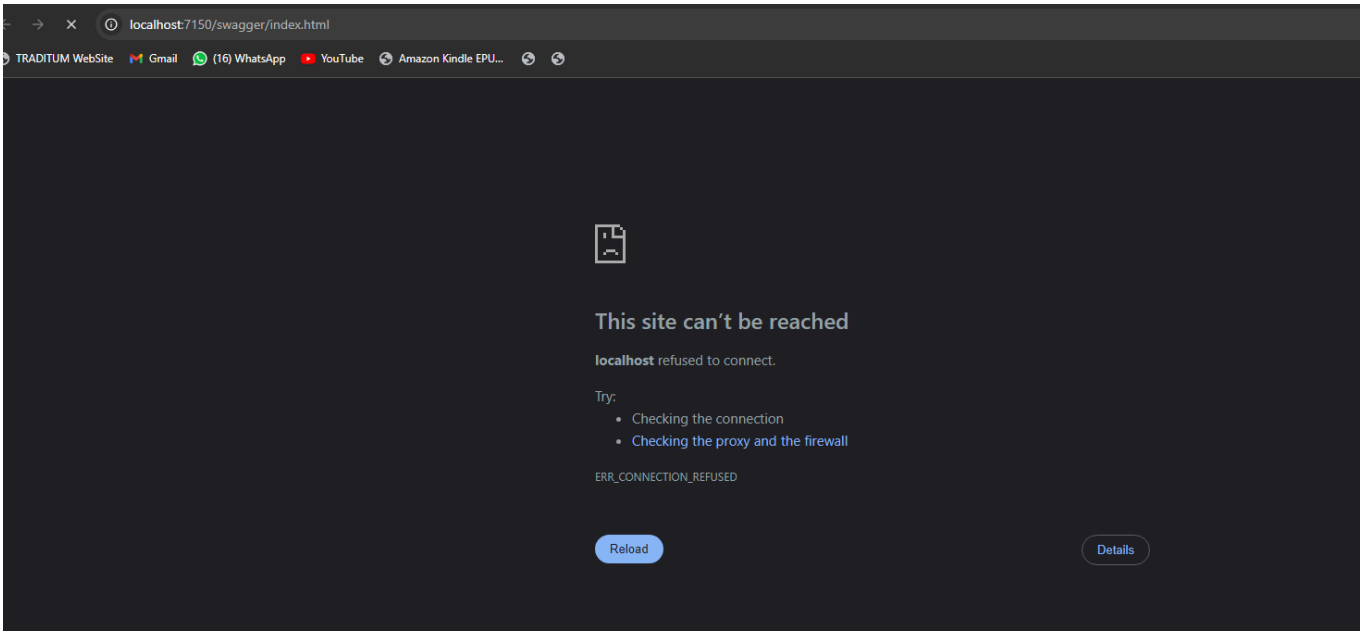
G)



H)

```
PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudAng
ular> ng test
✔Browser application bundle generation complete.
24 09 2024 18:28:38.605:WARN [karma]: No captured browser, open http://localhost:9876/
24 09 2024 18:28:38.629:INFO [karma-server]: Karma v6.4.3 server started at http://localhost:9876/
24 09 2024 18:28:38.629:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
24 09 2024 18:28:38.635:INFO [launcher]: Starting browser Chrome
24 09 2024 18:28:39.307:INFO [Chrome 129.0.0.0 (Windows 10)]: Connected on socket _wn5lrUfX-l4QSlYAAAB with id 77651704
Chrome 129.0.0.0 (Windows 10): Executed 4 of 4 SUCCESS (0.121 secs / 0.074 secs)
TOTAL: 4 SUCCESS
```

I)



4.5 Agregamos generación de reporte XML de nuestras pruebas de front.

A)

```
PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudAng
ular> npm install karma-junit-reporter --save-dev

added 2 packages, and audited 937 packages in 3s

120 packages are looking for funding
  run `npm fund` for details

11 vulnerabilities (7 moderate, 4 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

B)

appsettings.json | app.component.spec.ts | employee.service.spec.ts | karma.conf.js

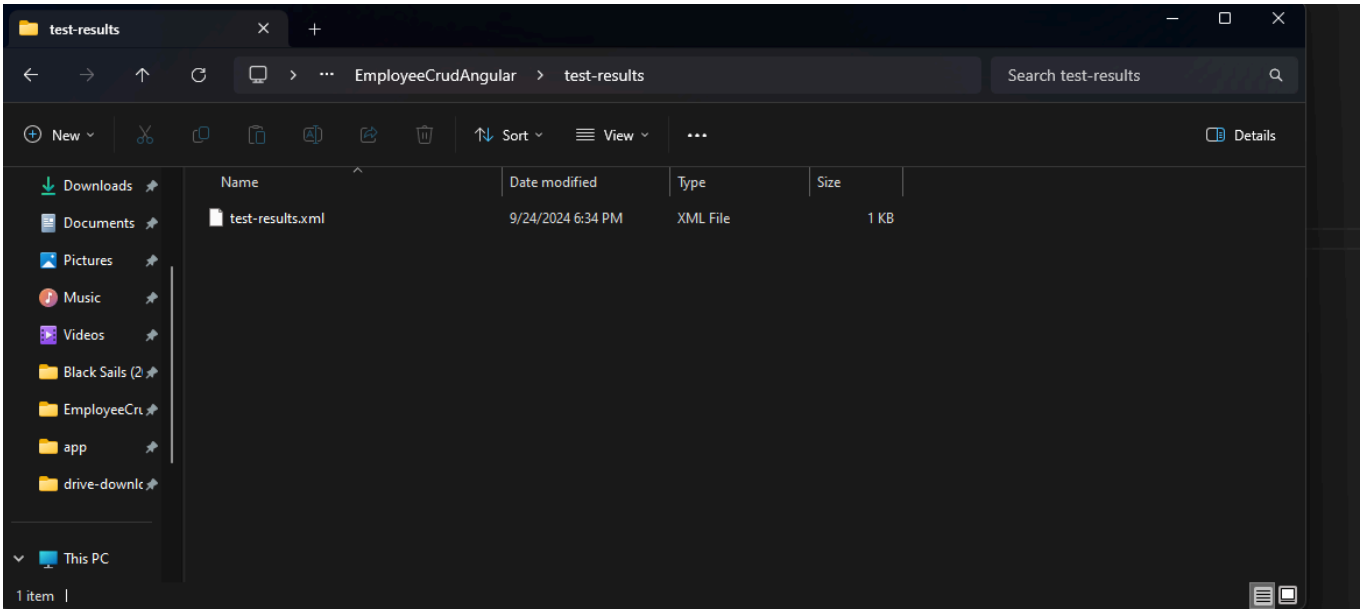
C: > Users > santi > Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > karma.conf.js > ...

```
module.exports = function (config) {
  config.set({
    frameworks: ['jasmine', '@angular-devkit/build-angular'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      require('karma-junit-reporter'),
      require('@angular-devkit/build-angular/plugins/karma')
    ],
    reporters: ['progress', 'junit'],
    junitReporter: {
      outputDir: 'test-results',
      outputFile: 'test-results.xml',
      useBrowserName: false
    },
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['ChromeHeadless'],
    singleRun: true,
    restartOnFileChange: true
  });
};
```

C)

```
PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudAng
ular> ng test --karma-config=karma.conf.js --watch=false --browsers ChromeHeadless
✔Browser application bundle generation complete.
24 09 2024 18:34:46.961:INFO [karma-server]: Karma v6.4.3 server started at http://localhost:9877/
24 09 2024 18:34:46.963:INFO [launcher]: Launching browsers ChromeHeadless with concurrency unlimited
24 09 2024 18:34:46.968:INFO [launcher]: Starting browser ChromeHeadless
24 09 2024 18:34:47.666:INFO [Chrome Headless 129.0.0.0 (Windows 10)]: Connected on socket fuHJfze2HGR6_GX8AAAB with id
30921374
Chrome Headless 129.0.0.0 (Windows 10): Executed 4 of 4 SUCCESS (0.127 secs / 0.069 secs)
TOTAL: 4 SUCCESS
```

D)



4.6 Modificamos el código de nuestra API y creamos nuevas pruebas unitarias:

1. Al agregar y al editar un empleado, controlar que el nombre del empleado no esté repetido.

```
36          public async Task<IActionResult> Create([FromBody] Employee employee)
37          {
38              // Verificar si ya existe un empleado con el mismo nombre
39              var existingEmployee = await _context.Employees
40                  .AnyAsync(e => e.Name == employee.Name);
41
42              if (existingEmployee)
43              {
44                  return BadRequest("El nombre del empleado ya existe.");
45              }
46
47              employee.CreatedDate = DateTime.Now;
48              await _context.Employees.AddAsync(employee);
49              await _context.SaveChangesAsync();
50
51              return Ok(employee);
52          }
53
54          [HttpPut]
55          public async Task<IActionResult> Update([FromBody] Employee employee)
56          {
57              // Verificar si ya existe un empleado con el mismo nombre (exceptuando el actual)
58              var existingEmployee = await _context.Employees
59                  .AnyAsync(e => e.Name == employee.Name && e.Id != employee.Id);
60
61              if (existingEmployee)
62              {
63                  return BadRequest("El nombre del empleado ya existe.");
64              }
65
66              var employeeToUpdate = await _context.Employees.FindAsync(employee.Id);
67
68              if (employeeToUpdate == null)
69              {
70                  return NotFound("El empleado no existe.");
71              }
72
73              employeeToUpdate.Name = employee.Name;
74              await _context.SaveChangesAsync();
75
76              return Ok(employeeToUpdate);
77          }
78
```

2. La longitud máxima del nombre y apellido del empleado debe ser de 100 caracteres.

```
public async Task<IActionResult> Create([FromBody] Employee employee)
{
    // Verificar si la longitud del nombre excede los 100 caracteres
    if (employee.Name.Length > 100)
    {
        return BadRequest("El nombre no puede tener más de 100 caracteres.");
    }

    // Verificar si ya existe un empleado con el mismo nombre
    var existingEmployee = await _context.Employees
        .AnyAsync(e => e.Name == employee.Name);

    if (existingEmployee)
    {
        return BadRequest("El nombre del empleado ya existe.");
    }

    employee.CreatedDate = DateTime.Now;
    await _context.Employees.AddAsync(employee);
    await _context.SaveChangesAsync();

    return Ok(employee);
}
```

```csharp
[HttpPut]
public async Task<IActionResult> Update([FromBody] Employee employee)
{
    // Verificar si la longitud del nombre excede los 100 caracteres
    if (employee.Name.Length > 100)
    {
        return BadRequest("El nombre no puede tener más de 100 caracteres.");
    }

    // Verificar si ya existe un empleado con el mismo nombre (exceptuando el actual)
    var existingEmployee = await _context.Employees
        .AnyAsync(e => e.Name == employee.Name && e.Id != employee.Id);

    if (existingEmployee)
    {
        return BadRequest("El nombre del empleado ya existe.");
    }

    var employeeToUpdate = await _context.Employees.FindAsync(employee.Id);

    if (employeeToUpdate == null)
    {
        return NotFound("El empleado no existe.");
    }

    employeeToUpdate.Name = employee.Name;
    await _context.SaveChangesAsync();

    return Ok(employeeToUpdate);
}
```

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:7150/api/Employee/Create' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 0,
  "name": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
  "createdDate": "2024-09-24T21:57:10.365Z"
}'
```

Request URL

```
http://localhost:7150/api/Employee/Create
```

Server response

Code	Details
400 *Undocumented*	Error: Bad Request

Response body

```
El nombre no puede tener más de 100 caracteres.
```

Download

Response headers

```
access-control-allow-origin: *
content-type: text/plain; charset=utf-8
date: Tue,24 Sep 2024 21:57:26 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	*No links*

3. Validar que el nombre tenga un número mínimo de caracteres, por ejemplo, al menos dos caracteres para evitar entradas inválidas como "A".

```
[HttpPost]
public async Task<IActionResult> Create([FromBody] Employee employee)
{
    // Verificar si la longitud del nombre es menor a 2 caracteres
    if (employee.Name.Length < 2)
    {
        return BadRequest("El nombre debe tener al menos 2 caracteres.");
    }

    // Verificar si la longitud del nombre excede los 100 caracteres
    if (employee.Name.Length > 100)
    {
        return BadRequest("El nombre no puede tener más de 100 caracteres.");
    }

    // Verificar si ya existe un empleado con el mismo nombre
    var existingEmployee = await _context.Employees
        .AnyAsync(e => e.Name == employee.Name);

    if (existingEmployee)
    {
        return BadRequest("El nombre del empleado ya existe.");
    }

    employee.CreatedDate = DateTime.Now;
    await _context.Employees.AddAsync(employee);
    await _context.SaveChangesAsync();

    return Ok(employee);
}
```

```
[HttpPut]
public async Task<IActionResult> Update([FromBody] Employee employee)
{
    // Verificar si la longitud del nombre es menor a 2 caracteres
    if (employee.Name.Length < 2)
    {
        return BadRequest("El nombre debe tener al menos 2 caracteres.");
    }

    // Verificar si la longitud del nombre excede los 100 caracteres
    if (employee.Name.Length > 100)
    {
        return BadRequest("El nombre no puede tener más de 100 caracteres.");
    }

    // Verificar si ya existe un empleado con el mismo nombre (exceptuando el actual)
    var existingEmployee = await _context.Employees
        .AnyAsync(e => e.Name == employee.Name && e.Id != employee.Id);

    if (existingEmployee)
    {
        return BadRequest("El nombre del empleado ya existe.");
    }

    var employeeToUpdate = await _context.Employees.FindAsync(employee.Id);

    if (employeeToUpdate == null)
    {
        return NotFound("El empleado no existe.");
    }

    employeeToUpdate.Name = employee.Name;
    await _context.SaveChangesAsync();

    return Ok(employeeToUpdate);
}
```

4. Verificar que el nombre no contenga números, ya que no es común en los nombres de empleados.

```
[HttpPost]
public async Task<IActionResult> Create([FromBody] Employee employee)
{
    // Verificar si el nombre contiene números
    if (Regex.IsMatch(employee.Name, @"\d"))
    {
        return BadRequest("El nombre no puede contener números.");
    }

    // Verificar si la longitud del nombre es menor a 2 caracteres
    if (employee.Name.Length < 2)
    {
        return BadRequest("El nombre debe tener al menos 2 caracteres.");
    }

    // Verificar si la longitud del nombre excede los 100 caracteres
    if (employee.Name.Length > 100)
    {
        return BadRequest("El nombre no puede tener más de 100 caracteres.");
    }

    // Verificar si ya existe un empleado con el mismo nombre
    var existingEmployee = await _context.Employees
        .AnyAsync(e => e.Name == employee.Name);

    if (existingEmployee)
    {
        return BadRequest("El nombre del empleado ya existe.");
    }

    employee.CreatedDate = DateTime.Now;
    await _context.Employees.AddAsync(employee);
    await _context.SaveChangesAsync();

    return Ok(employee);
}
```

```
[HttpPut]
public async Task<IActionResult> Update([FromBody] Employee employee)
{
    // Verificar si el nombre contiene números
    if (Regex.IsMatch(employee.Name, @"\d"))
    {
        return BadRequest("El nombre no puede contener números.");
    }

    // Verificar si la longitud del nombre es menor a 2 caracteres
    if (employee.Name.Length < 2)
    {
        return BadRequest("El nombre debe tener al menos 2 caracteres.");
    }

    // Verificar si la longitud del nombre excede los 100 caracteres
    if (employee.Name.Length > 100)
    {
        return BadRequest("El nombre no puede tener más de 100 caracteres.");
    }

    // Verificar si ya existe un empleado con el mismo nombre (exceptuando el actual)
    var existingEmployee = await _context.Employees
        .AnyAsync(e => e.Name == employee.Name && e.Id != employee.Id);

    if (existingEmployee)
    {
        return BadRequest("El nombre del empleado ya existe.");
    }

    var employeeToUpdate = await _context.Employees.FindAsync(employee.Id);

    if (employeeToUpdate == null)
    {
        return NotFound("El empleado no existe.");
    }

    employeeToUpdate.Name = employee.Name;
    await _context.SaveChangesAsync();

    return Ok(employeeToUpdate);
}
```

Curl

```
curl -X 'POST' \
  'http://localhost:7150/api/Employee/Create' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 8,
  "name": "hugo2",
  "createdDate": "2024-09-24T22:04:55.538Z"
}'
```

Request URL

```
http://localhost:7150/api/Employee/Create
```

Server response

Code	Details
400 *Undocumented*	Error: Bad Request

Response body

```
El nombre no puede contener números.
```

Download

Response headers

```
access-control-allow-origin: *
content-type: text/plain; charset=utf-8
date: Tue,24 Sep 2024 22:05:11 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

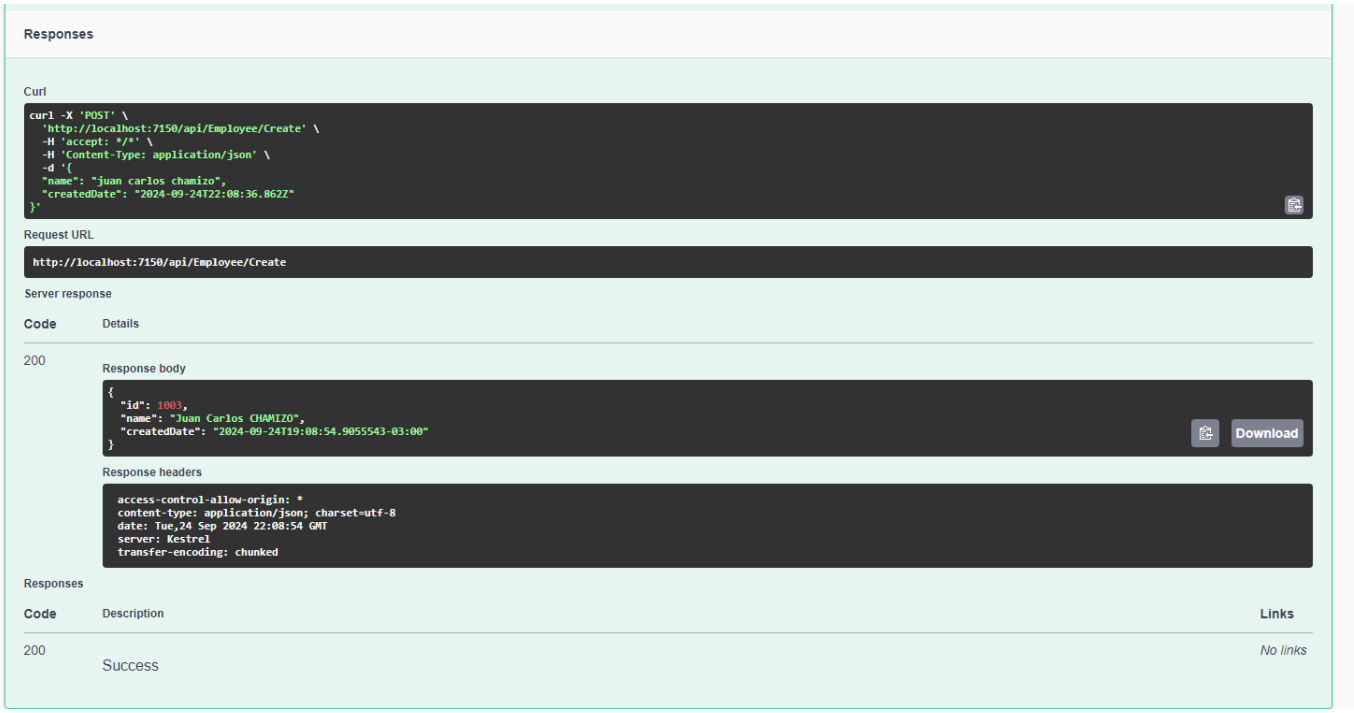
Code	Description	Links
200	Success	*No links*

5. Almacenar el nombre en la BD siempre con la primera letra de los nombres en Mayuscula y todo el apellido en Mayusculas. Ejemplo, si recibo juan carlos chamizo, se debe almacenar como Juan Carlos CHAMIZO.

```csharp
// Función auxiliar para formatear el nombre
private string FormatName(string fullName)
{
    var parts = fullName.Split(' ', StringSplitOptions.RemoveEmptyEntries);
    if (parts.Length == 0) return fullName;

    // Formatear cada parte del nombre con la primera letra en mayúscula
    var firstNames = parts.Take(parts.Length - 1)
                          .Select(p => CultureInfo.CurrentCulture.TextInfo.ToTitleCase(p.ToLower()));

    // Convertir el apellido (última parte) a mayúsculas
    var lastName = parts.Last().ToUpper();

    // Unir todas las partes
    return string.Join(' ', firstNames) + " " + lastName;
}
```

```
[HttpPost]
public async Task<IActionResult> Create([FromBody] Employee employee)
{
    // Verificar si el nombre contiene números
    if (Regex.IsMatch(employee.Name, @"\d"))
    {
        return BadRequest("El nombre no puede contener números.");
    }

    // Verificar si la longitud del nombre es menor a 2 caracteres
    if (employee.Name.Length < 2)
    {
        return BadRequest("El nombre debe tener al menos 2 caracteres.");
    }

    // Verificar si la longitud del nombre excede los 100 caracteres
    if (employee.Name.Length > 100)
    {
        return BadRequest("El nombre no puede tener más de 100 caracteres.");
    }

    // Formatear el nombre correctamente
    employee.Name = FormatName(employee.Name);

    // Verificar si ya existe un empleado con el mismo nombre
    var existingEmployee = await _context.Employees
        .AnyAsync(e => e.Name == employee.Name);

    if (existingEmployee)
    {
        return BadRequest("El nombre del empleado ya existe.");
    }

    employee.CreatedDate = DateTime.Now;
    await _context.Employees.AddAsync(employee);
    await _context.SaveChangesAsync();

    return Ok(employee);
}
```

```
    [HttpPut]
    public async Task<IActionResult> Update([FromBody] Employee employee)
    {
        // Verificar si el nombre contiene números
        if (Regex.IsMatch(employee.Name, @"\d"))
        {
            return BadRequest("El nombre no puede contener números.");
        }

        // Verificar si la longitud del nombre es menor a 2 caracteres
        if (employee.Name.Length < 2)
        {
            return BadRequest("El nombre debe tener al menos 2 caracteres.");
        }

        // Verificar si la longitud del nombre excede los 100 caracteres
        if (employee.Name.Length > 100)
        {
            return BadRequest("El nombre no puede tener más de 100 caracteres.");
        }

        // Formatear el nombre correctamente
        employee.Name = FormatName(employee.Name);

        // Verificar si ya existe un empleado con el mismo nombre (exceptuando el actual)
        var existingEmployee = await _context.Employees
            .AnyAsync(e => e.Name == employee.Name && e.Id != employee.Id);

        if (existingEmployee)
        {
            return BadRequest("El nombre del empleado ya existe.");
        }

        var employeeToUpdate = await _context.Employees.FindAsync(employee.Id);

        if (employeeToUpdate == null)
        {
            return NotFound("El empleado no existe.");
        }

        employeeToUpdate.Name = employee.Name;
        await _context.SaveChangesAsync();

        return Ok(employeeToUpdate);
    }
```



B. Crear las pruebas unitarias necesarias para validar las modificaciones realizadas en el código

```
[Fact]
public async Task Create_Fails_WhenNameTooShort()
{
    // Arrange
    var context = GetInMemoryDbContext();
    var controller = new EmployeeController(context);

    var newEmployee = new Employee { Id = 3, Name = "A" };

    // Act
    var result = await controller.Create(newEmployee);

    // Assert
    Assert.IsType<BadRequestObjectResult>(result);
}
```

```
    [Fact]
    public async Task Create_Fails_WhenNameContainsNumbers()
    {
        // Arrange
        var context = GetInMemoryDbContext();
        var controller = new EmployeeController(context);

        var newEmployee = new Employee { Id = 3, Name = "John123" };

        // Act
        var result = await controller.Create(newEmployee);

        // Assert
        Assert.IsType<BadRequestObjectResult>(result);
    }
```

```
[Fact]
public async Task Create_Fails_WhenNameTooLong()
{
    // Arrange
    var context = GetInMemoryDbContext();
    var controller = new EmployeeController(context);

    var newEmployee = new Employee { Id = 3, Name = new string('a', 101) }; // Nombre con más de 100 caracteres

    // Act
    var result = await controller.Create(newEmployee);

    // Assert
    Assert.IsType<BadRequestObjectResult>(result);
}
```

```
 [Fact]
 public async Task Create_Fails_WhenNameIsDuplicated()
 {
        // Arrange
        var context = GetInMemoryDbContext();
        var controller = new EmployeeController(context);

        var employee1 = new Employee { Id = 1, Name = "John Doe" };
        var employee2 = new Employee { Id = 2, Name = "John Doe" }; // Nombre duplicado

        context.Employees.Add(employee1);
        await context.SaveChangesAsync();

        // Act
        var result = await controller.Create(employee2);

        // Assert
        Assert.IsType<BadRequestObjectResult>(result); // Controlar duplicados correctamente
 }
```

```
    [Fact]
    public async Task Create_FormatsNameCorrectly()
    {
        // Arrange
        var context = GetInMemoryDbContext();
        var controller = new EmployeeController(context);

        var newEmployee = new Employee { Id = 3, Name = "juan carlos chamizo" };

        // Act
        await controller.Create(newEmployee);

        // Assert
        var employee = await context.Employees.FindAsync(3);
        Assert.NotNull(employee);
        Assert.Equal("Juan Carlos CHAMIZO", employee.Name);
    }
```

```
Time Elapsed 00:00:01.41
PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudApi
.Tests> dotnet test
  Determining projects to restore...
  All projects are up-to-date for restore.
  EmployeeCrudApi -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sam
  ple\EmployeeCrudApi\EmployeeCrudApi\bin\Debug\net8.0\EmployeeCrudApi.dll
  EmployeeCrudApi.Tests -> C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CR
  UD_Sample\EmployeeCrudApi.Tests\bin\Debug\net8.0\EmployeeCrudApi.Tests.dll
Test run for C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\Emplo
yeeCrudApi.Tests\bin\Debug\net8.0\EmployeeCrudApi.Tests.dll (.NETCoreApp,Version=v8.0)
VSTest version 17.11.0 (x64)

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed!  - Failed:     0, Passed:    10, Skipped:     0, Total:    10, Duration: 113 ms - EmployeeCrudApi.Tests.dll (net
8.0)
```