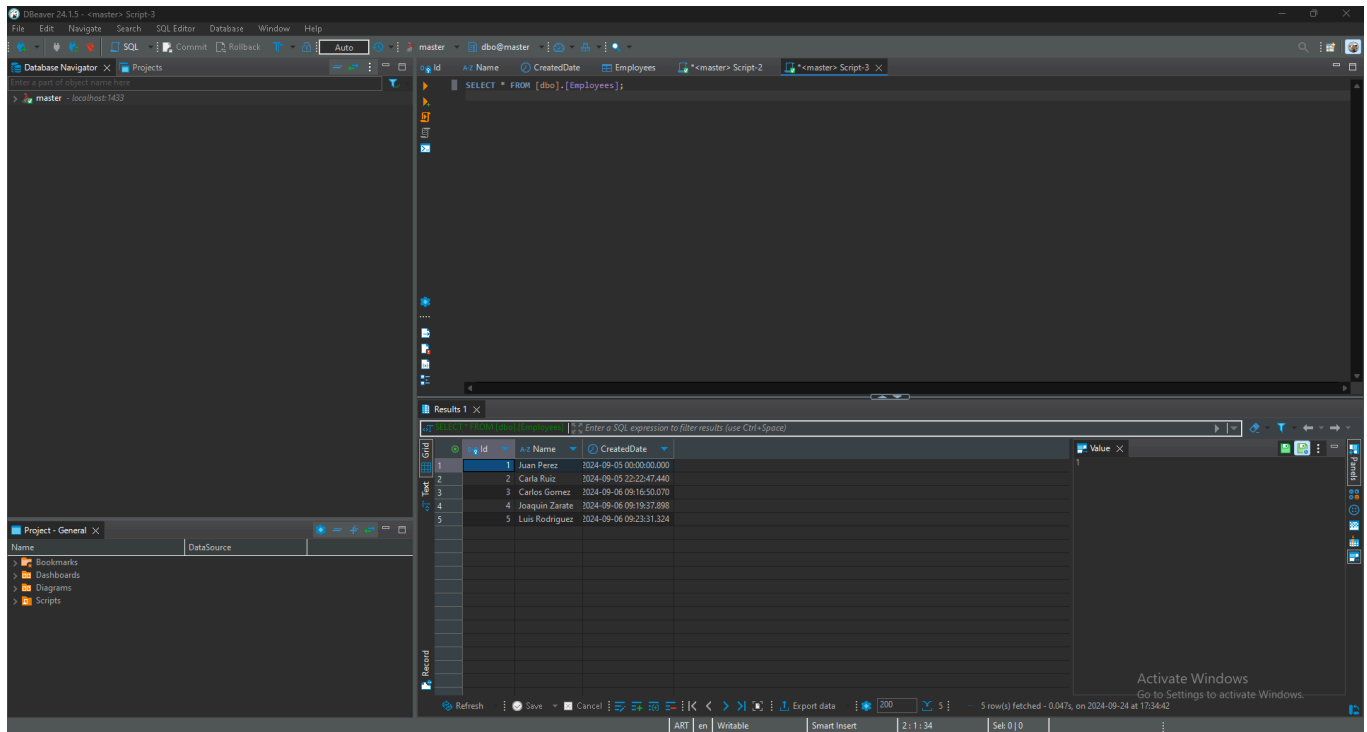


TP6

4.1 Creación de una BD SQL Server para nuestra App



4.2 Obtener nuestra App

```
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.HttpPolicy.HttpsRedirectionMiddleware[3]
    Failed to determine the https port for redirect.
```

SwaggerUI

localhost7150/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: 

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:

Employee

GET

/api/Employee/GetAll

Parameters

Cancel

No parameters

Execute

Clear

Responses

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 20:52:34 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

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\net8.0\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\net8.0\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\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: 5, Skipped: 0, Total: 5, Duration: 98 ms - EmployeeCrudApi.Tests.dll (net8.0)

```

I)

```

1 {
2   "ConnectionStrings": {
3     "DefaultConnection": "Server=localhost,1433;Initial Catalog=master;Persist Security Info=False;User ID=SA;Password=passwordincorrecta;MultipleActiveResultSets=False;Encrypt=False;TrustServerCertificate=True;Connect Timeout=30;Database=master"
4   },
5   "Logging": {
6     "LogLevel": {
7       "Default": "Information",
8       "Microsoft.AspNetCore": "Warning"
9     }
10  },
11  "AllowedHosts": "*"
12 }
13

```

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 stateObj, 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 enlistOW)
at Microsoft.Data.SqlClient.SqlInternalConnectionTds.AttemptOneLogin(ServerInfo serverInfo, String newPassword, SecureString newSecurePassword, Boolean ignoreSniOpenTimeout, TimeoutTimer timeout, Boolean withFailover)
at Microsoft.Data.SqlClient.SqlInternalConnectionTds.LoginNoFailover(ServerInfo serverInfo, String newPassword, SecureString newSecurePassword, Boolean redirectedUserInstance, SqlConnectionString connectionOptions, SqlCredential credential, TimeoutTimer timeout)
at Microsoft.Data.SqlClient.SqlInternalConnectionTds.OpenLoginEnlist(TimeoutTimer timeout, SqlConnectionString connectionOptions, SqlCredential credential, String newPassword, SecureString newSecurePassword, Boolean redirectedUserInstance, SqlConnectionString userConnectionOptions, SessionData reconnectSessionData, Boolean applyTransientFaultHandling, String accessToken, DbConnectionPool pool)
at Microsoft.Data.SqlClient.SqlConnectionFactory.CreateConnection(DbConnectionOptions options, DbConnectionPoolKey poolKey, Object poolGroupProviderInfo, DbConnectionPool pool, DbConnection owningConnection, DbConnectionOptions userOptions)
at Microsoft.Data.ProviderBase.DbConnectionFactory.CreatePooledConnection(DbConnectionPool pool, DbConnection owningObject, DbConnectionOptions options, DbConnectionPoolKey poolKey, DbConnectionOptions 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, DbConnectionOptions 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)

```

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\net8.0\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\net8.0\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\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: 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	<div><div>Response body</div><div><pre>[{ "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" }]</pre></div><div><div>Download</div></div></div>

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 > ...
1 import { TestBed } from '@angular/core/testing';
2 import { AppComponent } from './app.component'; // Ajusta la ruta si es necesario
3
4 describe('AppComponent', () => {
5   beforeEach(async () => {
6     await TestBed.configureTestingModule({
7       imports: [AppComponent], // Usa imports en lugar de declarations
8     }).compileComponents();
9   });
10
11   it('should render title', () => {
12     const fixture = TestBed.createComponent(AppComponent);
13     fixture.detectChanges();
14     const compiled = fixture.nativeElement as HTMLElement;
15     expect(compiled.querySelector('h1')?.textContent).toContain('EmployeeCrudAngular');
16   });
17
18 });
```

C)

```
C:\Users\santi> Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > src > app > TS employee.service.spec.ts > ...
1 import { TestBed } from '@angular/core/testing';
2 import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';
3 import { EmployeeService } from './employee.service';
4 import { Employee } from './employee.model';
5 import { DatePipe } from '@angular/common';
6
7 describe('EmployeeService', () => {
8   let service: EmployeeService;
9   let httpMock: HttpTestingController;
10   let datePipe: DatePipe;
11
12   beforeEach(() => {
13     TestBed.configureTestingModule({
14       imports: [HttpClientTestingModule],
15       providers: [
16         EmployeeService,
17         DatePipe
18       ]
19     });
20
21     service = TestBed.inject(EmployeeService);
22     httpMock = TestBed.inject(HttpTestingController);
23     datePipe = TestBed.inject(DatePipe);
24   });
25
26   afterEach(() => {
27     httpMock.verify();
28   });
29
30
31   it('should retrieve all employees', () => {
32     const today = new Date();
33     const expectedDateTime = datePipe.transform(today, 'dd/MM/yyyy HH:mm:ss', undefined) ?? ''; // Consistente con el servicio
34
35     const dummyEmployees: Employee[] = [
36       new Employee(1, 'John Doe', expectedDateTime),
37       new Employee(2, 'Jane Smith', expectedDateTime)
38     ];
39   });
```

D)

```
C:\Users\santi> Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > src > app > employee > TS employee.component.spec.ts > ...
1 import { TestBed } from '@angular/core/testing';
2 import { EmployeeComponent } from './employee.component';
3 import { HttpClientTestingModule } from '@angular/common/http/testing';
4 import { DatePipe } from '@angular/common';
5
6 describe('EmployeeComponent', () => {
7   beforeEach(() => {
8     TestBed.configureTestingModule({
9       imports: [EmployeeComponent, HttpClientTestingModule],
10      providers: [DatePipe] // Añade DatePipe a los proveedores
11    });
12  });
13
14  it('should create', () => {
15    const fixture = TestBed.createComponent(EmployeeComponent);
16    const component = fixture.componentInstance;
17    expect(component).toBeTruthy();
18  });
19 });
```

E)

```


C:\> Users > santi > Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > src > app > addemployee > TS addemployee.component.spec.ts > ...
1  import { TestBed } from '@angular/core/testing';
2  import { AddemployeeComponent } from './addemployee.component';
3  import { HttpClientTestingModule } from '@angular/common/http/testing';
4  import { ActivatedRoute } from '@angular/router';
5  import { of } from 'rxjs'; // para simular observables
6  import { DatePipe } from '@angular/common';
7
8  describe('AddemployeeComponent', () => {
9    beforeEach(() => {
10      TestBed.configureTestingModule({
11        imports: [AddemployeeComponent, HttpClientTestingModule],
12        providers: [
13          DatePipe,
14          {
15            provide: ActivatedRoute, // Simula ActivatedRoute
16            useValue: {
17              params: of({ id: 1 }) // simula el parámetro id en la URL
18            }
19          }
20        ]
21      });
22    });
23
24    it('should create', () => {
25      const fixture = TestBed.createComponent(AddemployeeComponent);
26      const component = fixture.componentInstance;
27      expect(component).toBeTruthy();
28    });
29  });

```

G)

Karma v 6.4.3 - connected; test: complete; DEBUG

Chrome 129.0.0.0 (Windows 10) is idle

 Jasmine 4.6.1
 Options

4 specs, 0 failures, randomized with seed 47682 finished in 0.084s

- AppComponent
 - should render title
- EmployeeComponent
 - should create
- AddemployeeComponent
 - should create
- Employeeservice
 - should retrieve all employees

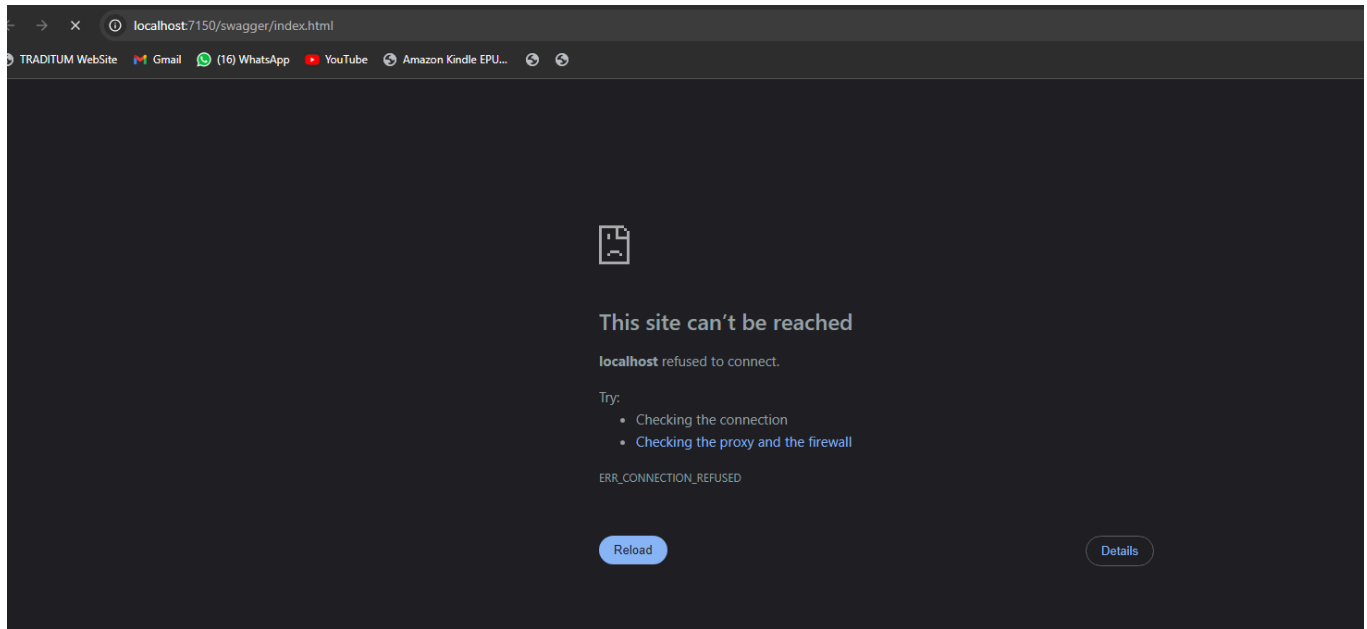
H)

```

PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudAngular> 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_WebAPI\NetCore8_CRUD_Sample\EmployeeCrudAngular> 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)

```

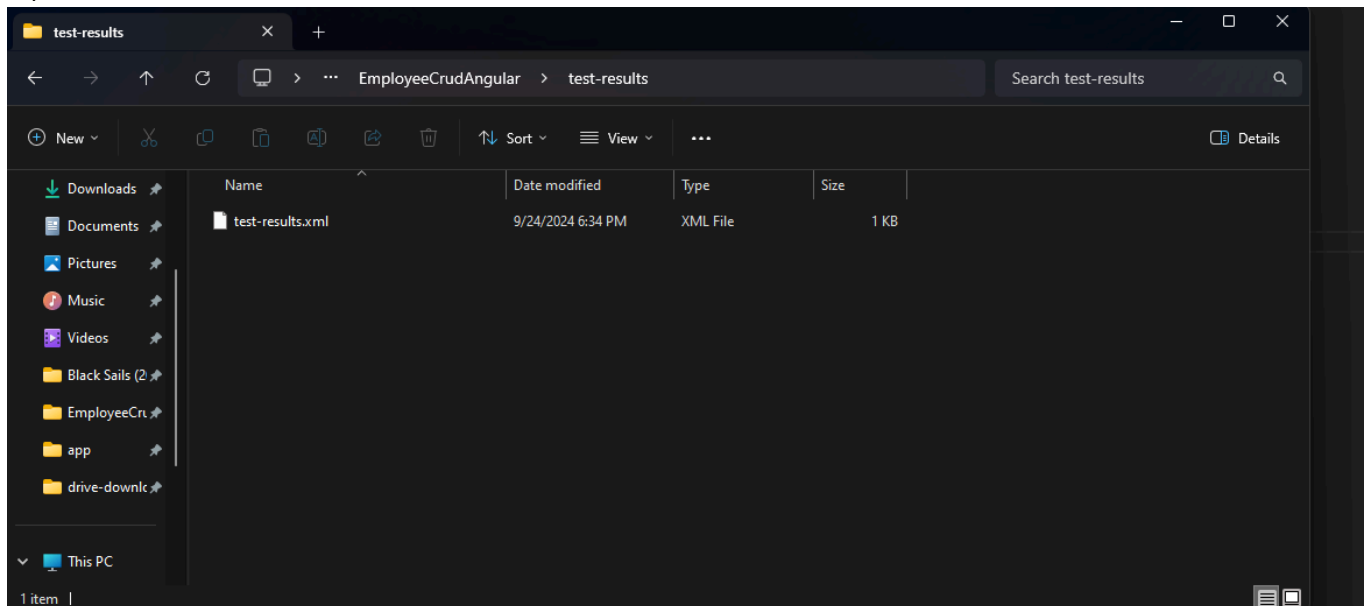
appsettings.json  TS app.component.spec.ts  TS employee.service.spec.ts  karma.conf.js x
C: > Users > santi > Documents > AA-Universidad > Ingeniería de Software III > Angular_WebAPINetCore8_CRUD_Sample > EmployeeCrudAngular > karma.conf.js > ...
1  module.exports = function (config) {
2      config.set({
3          frameworks: ['jasmine', '@angular-devkit/build-angular'],
4          plugins: [
5              require('karma-jasmine'),
6              require('karma-chrome-launcher'),
7              require('karma-junit-reporter'),
8              require('@angular-devkit/build-angular/plugins/karma')
9          ],
10         reporters: ['progress', 'junit'],
11         junitReporter: {
12             outputDir: 'test-results',
13             outputFile: 'test-results.xml',
14             useBrowserName: false
15         },
16         port: 9876,
17         colors: true,
18         logLevel: config.LOG_INFO,
19         autoWatch: true,
20         browsers: ['ChromeHeadless'],
21         singleRun: true,
22         restartOnFileChange: true
23     });
24 };
```

C)

```

PS C:\Users\santi\Documents\AA-Universidad\Ingeniería de Software III\Angular_WebAPINetCore8_CRUD_Sample\EmployeeCrudAngular> 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
```

Curl

```
curl -X 'POST' \
  'http://localhost:7150/api/Employee/Create' \
  -H 'accept: */*' \
  -H 'Content-Type: text/json' \
  -d '{
    "id": 1,
    "name": "Juan Perez",
    "createdDate": "2024-09-24T21:45:41.157Z"
  }'
```

Request URL

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

Server response

Code	Details
400	Error: Bad Request

Response body

El nombre del empleado ya existe.

Response headers

```
access-control-allow-origin: *
content-type: text/plain; charset=utf-8
date: Tue, 24 Sep 2024 21:47:55 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	No links

Curl

```
curl -X 'PUT' \
  'http://localhost:7150/api/Employee/Update' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 1,
    "name": "Carla Ruiz",
    "createdDate": "2024-09-24T21:49:48.003Z"
  }'
```

Request URL

http://localhost:7150/api/Employee/Update

Server response

Code	Details
400	Error: Bad Request

Response body

El nombre del empleado ya existe.

Response headers

```
access-control-allow-origin: *
content-type: text/plain; charset=utf-8
date: Tue, 24 Sep 2024 21:50:02 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	No links

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);
}
```

```
[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);
}
```

[illegible]

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);
}
```

Curl

```
curl -X 'POST' \
  'http://localhost:7150/api/Employee/Create' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 6,
    "name": "a",
    "createdDate": "2024-09-24T22:01:05.098Z"
  }'
```

Request URL

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

Server response

Code	Details
400	Error: Bad Request

Undocumented

Response body

El nombre debe tener al menos 2 caracteres.

Response headers

```
access-control-allow-origin: *
content-type: text/plain; charset=utf-8
date: Tue, 24 Sep 2024 22:01:11 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	No links

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	Error: Bad Request

Response body

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

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.

```
// 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);
}

```

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:7150/api/Employee/Create' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "juan carlos chamizo",
    "createdDate": "2024-09-24T22:08:36.862Z"
  }'
```

Request URL

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

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": 1003, "name": "Juan Carlos CHAMIZO", "createdDate": "2024-09-24T19:08:54.9055543-03:00" }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * content-type: application/json; charset=utf-8 date: Tue, 24 Sep 2024 22:08:54 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses

Code	Description	Links
200	Success	No links

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\Empl
oyeeCrudApi.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)
```

4.7 Modificamos el código de nuestro Front y creamos nuevas pruebas unitarias:

A. Realizar en el código del front las mismas modificaciones hechas a la API.

B. Las validaciones deben ser realizadas en el front sin llegar a la API, y deben ser mostradas en un toast como por ejemplo

EmployeeCrudAngular

Create new employee:

Name

s

Create

Error



El nombre debe tener al menos 2 caracteres.

EmployeeCrudAngular

Create new employee:

Name

Juan PEREZ

Create

Error



El nombre del empleado ya existe.

EmployeeCrudAngular

Create new employee:

Name

hugo2

Create

Error



El nombre no puede contener números.

EmployeeCrudAngular

Create new employee:

Name

hugo racca

Create

1005 Hugo RACCA

C. Crear las pruebas unitarias necesarias en el front para validar las modificaciones realizadas en el código del front.

