Prueba Técnica GOAL SYSTEMS

Junio 2021

Beatriz Martín Guadaño

Contenido

[Base de Datos 3](#_Toc75793286)

[Modelo de Base de Datos 3](#_Toc75793287)

[Creación tablas 4](#_Toc75793288)

[Empresa 4](#_Toc75793289)

[Empleado 4](#_Toc75793290)

[Tarea 4](#_Toc75793291)

[Incidencia 5](#_Toc75793292)

[Estructura del Código 5](#_Toc75793293)

[Instrucciones de Ejecución 6](#_Toc75793294)

[Respuestas 6](#_Toc75793295)

[1. Métodos que expone la API 6](#_Toc75793296)

[2. Notificar que se ha creado una incidencia a un Empleado 7](#_Toc75793297)

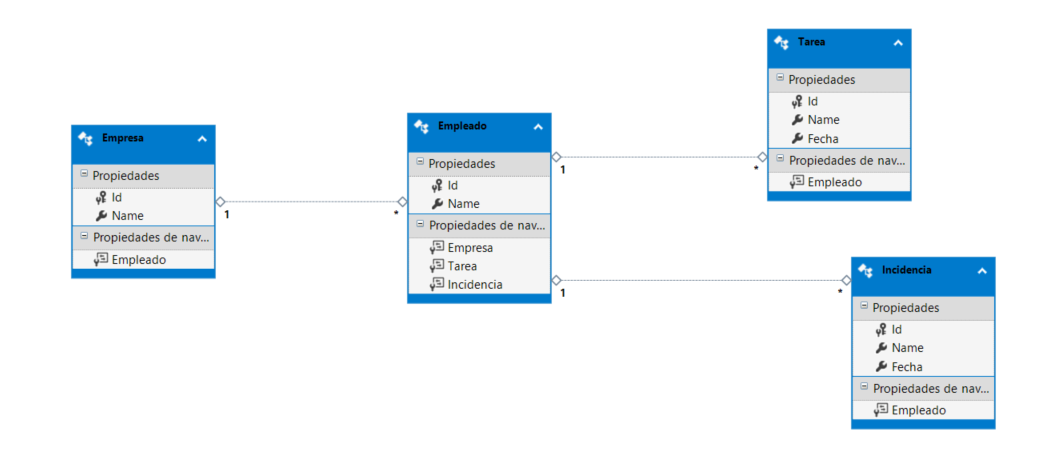
[3. Base de Datos 7](#_Toc75793298)

[4. Seguridad 7](#_Toc75793299)

[Anexo: Mejoras 7](#_Toc75793300)

# Base de Datos

## Modelo de Base de Datos



Se ha diseñado la base de datos con 4 tablas:

1. Empresa
   * Id: Identidicador de la empresa. PK. Identity.
   * Name: Nombre de la empresa.
2. Empleado
   * Id: Identidicador del empleado. PK. Identity.
   * Name: Nombre del empleado.
   * IdEmpresa: FK al identificador de la tabla empresa.
3. Tarea
   * Id: Identidicador de la tarea. PK. Identity.
   * Name: Nombre de la tarea.
   * Fecha: Fecha de la tarea. Unique con IdEmpleado.
   * IdEmpleado: FK al identificador de la tabla empleado. Unique con Fecha.
4. Incidencia
   * Id: Identidicador de la incidencia. PK. Identity.
   * Name: Nombre de la incidencia.
   * Fecha: Fecha de la incidencia.
   * IdEmpleado: FK al identificador de la tabla empleado.

Relaciones entre tablas:

* FK:
  + Empresas – Empleados (1:N)
  + Empleados – Tareas (1:N)
  + Empleados – Incidencias (1:N)
* Unique Key
  + Tareas (Fecha/Empleado)
* PK:
  + Empresas (Id)
  + Empleado (Id)
  + Tareas (Id)
  + Incidencias (Id)

## Creación tablas

Se especifican como serían las queries necesarias para la creación de la base de datos, aunque en la aplicación no es necesario crearlas, porque se ha optado por utilizar code first con EF y es la propia aplicación la encargada de generarlas, siendo totalmente transparente para el usuario. En todo caso, si diera algún error se debería emplear migrations.

### Empresa

CREATE TABLE [dbo].[Empresa] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Name] NVARCHAR (100) NULL

);

### Empleado

CREATE TABLE [dbo].[Empleado] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Name] NVARCHAR (100) NULL,

[EmpresaId] INT NOT NULL

);

GO

CREATE NONCLUSTERED INDEX [IX\_EmpresaId]

ON [dbo].[Empleado]([EmpresaId] ASC);

GO

ALTER TABLE [dbo].[Empleado]

ADD CONSTRAINT [PK\_dbo.Empleado] PRIMARY KEY CLUSTERED ([Id] ASC);

GO

ALTER TABLE [dbo].[Empleado]

ADD CONSTRAINT [FK\_dbo.Empleado\_dbo.Empresa\_EmpresaId] FOREIGN KEY ([EmpresaId]) REFERENCES [dbo].[Empresa] ([Id]) ON DELETE CASCADE;

### Tarea

CREATE TABLE [dbo].[Tarea] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Name] NVARCHAR (200) NULL,

[Fecha] DATETIME NOT NULL,

[EmpleadoId] INT NOT NULL

);

GO

CREATE UNIQUE NONCLUSTERED INDEX [Ix\_EmpleadoFecha]

ON [dbo].[Tarea]([Fecha] ASC, [EmpleadoId] ASC);

GO

ALTER TABLE [dbo].[Tarea]

ADD CONSTRAINT [PK\_dbo.Tarea] PRIMARY KEY CLUSTERED ([Id] ASC);

GO

ALTER TABLE [dbo].[Tarea]

ADD CONSTRAINT [FK\_dbo.Tarea\_dbo.Empleado\_EmpleadoId] FOREIGN KEY ([EmpleadoId]) REFERENCES [dbo].[Empleado] ([Id]) ON DELETE CASCADE;

### Incidencia

CREATE TABLE [dbo].[Incidencia] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Name] NVARCHAR (100) NULL,

[Fecha] DATETIME NOT NULL,

[EmpleadoId] INT NOT NULL

);

GO

CREATE NONCLUSTERED INDEX [IX\_EmpleadoId]

ON [dbo].[Incidencia]([EmpleadoId] ASC);

GO

ALTER TABLE [dbo].[Incidencia]

ADD CONSTRAINT [PK\_dbo.Incidencia] PRIMARY KEY CLUSTERED ([Id] ASC);

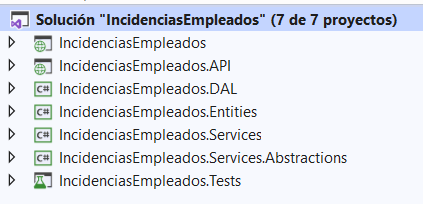
GO

ALTER TABLE [dbo].[Incidencia]

ADD CONSTRAINT [FK\_dbo.Incidencia\_dbo.Empleado\_EmpleadoId] FOREIGN KEY ([EmpleadoId]) REFERENCES [dbo].[Empleado] ([Id]) ON DELETE CASCADE;

# Estructura del Código

Se ha creado una solución .NET con varios proyectos MVC, Web API y de biblioteca de clases



IncidenciasEmpleados: Proyecto MVC que consume la API. Permite realizar gestiones de manera visual sobre las entidades Empleados e Incidencias.

IncidenciasEmpleados.API: Proyecto Web API que expone la interfaz para realizar gestiones sobre las entidades de la aplicación.

IncidenciasEmpleados.DAL: Biblioteca de clases que funciona como capa de acceso a base de datos.

IncidenciasEmpleados.Services.Abstractions: Biblioteca de clases que expone las interfaces de los servicios

IncidenciasEmpleados.Services: Biblioteca de clases que implementa las interfaces de la capa de servicios.

IncidenciasEmpleadosEntities: Biblioteca de clases que implementa las entidades, se ha optado por un modelo de Entity FW utilizando Code First para el desacoplamiento de bbdd. A fines de testing y hacer la aplicación funcional se ha implementado el acceso a bbdd sql server local db.

IncidenciassEmpleados.Test: Proyecto con los test unitarios.

# Instrucciones de Ejecución

El desarrollo se ha realizado con:

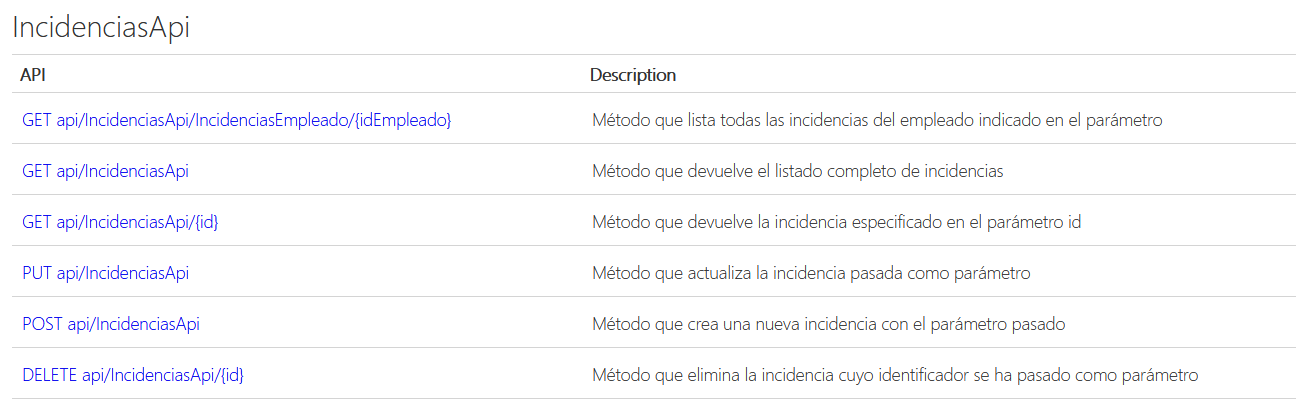
* Visual Studio 2019
* .NET Framework 4.7.2
* C#
* IIS Express
* SQL Server LocalDB (a fin de hacer la aplicación funcional y ejecutar un test unitario)

Para la ejecución de la aplicación, si se dispone de un equipo con la misma versión de SDK, Visual Studio 2012 o superior y este con las características de IIS Express y LocalDB sería suficiente para su funcionamiento.

# Respuestas

## Métodos que expone la API

|  |  |
| --- | --- |
| Método | Descripción |
| POST api/IncidenciasApi | Añadir incidencia a un empleado. |
| GET api/IncidenciasApi/IncidenciasEmpleado/{idEmpleado} | Obtener todas las incidencias de un empleado. |



## Notificar que se ha creado una incidencia a un Empleado

Para notificar la creación de una nueva incidencia asociada a un empleado podemos utilizar diferentes estrategias o alternativas, en función de la utilidad que se desee:

* Trigger de base de datos: Nos permite lanzar una ejecución en base de datos asociada al insert en la tabla Incidencias, que sería el disparador.
* SignalR: Haciendo uso de las notificaciones/hub de esta librería podemos notificar cambios a otros sistemas.
* Batch/Listener: Servicio en la aplicación o en bbdd encargado de detectar cambios y dejar registro en otros servicios.

## Base de Datos

SELECT TOP(1) Empleado.Id, Empleado.Name, COUNT(\*) as numIncidencias

FROM Empleado

INNER JOIN Empresa ON Empresa.Id = Empleado.EmpresaId

INNER JOIN Incidencia ON Incidencia.EmpleadoId = Empleado.Id

WHERE Empresa.Name = 'Mutiny'

AND Incidencia.Fecha BETWEEN CONVERT(datetime, '2021-05-01') AND CONVERT(datetime, '2021-05-31')

GROUP BY Empleado.Id, Empleado.Name

ORDER BY numIncidencias DESC

## Seguridad

De cara a mejorar la seguridad de la API y evitar accesos indeseados se debe de llevar a cabo la implementación de autenticación y autorización basada en usuarios, roles y permisos (por ejemplo, apoyándonos en JWT (JSON Web Token), OAuth, Identity Server, Directorio Activo, etc).

# Anexo: Mejoras

Aparte de las mejoras de autorización y autenticación indicadas en el apartado anterior, considero que se pueden realizar una serie de mejoras:

* Añadir Swagger para exponer la API de forma visual, documentarla y poder realizar test manuales.
* Añadir logging de las acciones y errores de la API (como por ejemplo NLog)
* Comentar/Describir todos o la mayoría de los métodos
* Ampliar el manejo y tratamiento de excepciones
* Ampliar la batería de pruebas unitarias, pruebas de carga/stress, etc
* Usar HTTPS para evitar que el tráfico de red pueda ser fácilmente interceptado e interpretado
* Validación de parámetros de entrada (datos, tipos de contenido, evitar mensajes de error explícitos)
* Evitar métodos que permitan acceso anónimo
* Uso de Tokens Antifalsificación (AntiForgeryToken)
* Separar DTOs de Entidades Lógicas en diferentes capas
* Establecer CSS a gusto del cliente
* Incluir, si es necesario, índices en las tablas para mejorar el rendimiento
* …