



TRABAJO FIN DE GRADO
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Gestor de eventos mediante
página web en Angular
y API REST

Alumno: Daniel Gil Calahorrano
Tutor: Rafael Juan Alamañac Garrido

Calahorra, junio, 2022

Índice

Introducción	3
Objetivos.....	3
Principales:	3
Específicos:	3
Definición del problema	4
Estimación de tiempo inicial.....	4
Diseño, Componentes y tecnologías de uso	5
Tecnologías.....	5
Diseño	5
Componentes	6
Análisis	7
Diseño de la aplicación	10
BBDD.....	10
API	11
JWT.....	11
Peticiones.....	12
Angular	13
Servicios.....	13
Routing.....	14
Helper.....	14
Implementación	14
BBDD.....	14
API	16
JWT.....	16
Peticiones.....	18
Angular	18
Pruebas	21
Ampliación y mejoras	21
Manual de usuario	22
Manual de instalación	23
Instalación servidor.....	23
Necesario para Web	24
Necesario para la BBDD.....	25
Cálculo de tiempo final.....	26
Conclusión.....	27
Bibliografía.....	27
Anexos.....	28

Introducción

El principal problema que se quiere tratar es la falta de eventos gratuitos que hay en las localidades. Exceptuando fiestas y ocasiones especiales, apenas se realizan actividades, y, en caso de que se organice alguna, resulta difícil enterarse a tiempo. A esto sumamos que estas pocas actividades no gestionan su aforo, lo que complica todavía más organizar la gente que puede asistir. Mi proyecto es diseñado con la finalidad de promover la creación de eventos y actividades, y con el objetivo de resolver los problemas anteriormente nombrados.

Mi aplicación será fácilmente accesible para cualquier persona, mediante una página web. Dicha página web estará alojada en la página del ayuntamiento, para facilitar su búsqueda. De este modo cada pueblo/ciudad tendrá su propio “Event Master”.

En esta encontrarás el modo de unirse a eventos de la comunidad y también crear los tuyos propios. Se podrá visualizar los eventos en los que participas, los disponibles y los finalizados.

Todos los eventos son 100% gratuitos, además, generará entradas para ayudar a controlar los aforos.

Objetivos

Principales:

- Controlar los aforos.
- Proporcionar herramienta de creación de eventos.
- Facilitar la participación en eventos

Específicos:

- Creación de página web con angular
- Auto generador de entradas
- Documentación de API
- Inicio de sesión
- Registro para nuevos usuarios
- Historial de eventos

Definición del problema

El principal problema es la falta de información sobre los eventos que van a realizarse en cualquier lugar. Generalmente, si quieres enterarte de los eventos que van a suceder en tu zona, tienes que estar atento a las noticias, periódicos o páginas web de los propios organizadores, corriendo el riesgo de no llegar a enterarte de lo que va a suceder.

Además, es difícil gestionar bien la cantidad de personas que pueden entrar, ya que en eventos gratuitos no suele haber entradas ni nada parecido, por lo que el aforo es demasiado variable.

Una forma de unificar todos estos eventos o actividades en una misma página resolvería esta cuestión, pues solamente con una búsqueda tenemos información relacionada con todo lo que va a suceder.

Estimación de tiempo inicial

- Diseño y creación de la Base de datos: 2 Horas.
- Diseño y creación de la API: 15 Horas.
- Diseño y creación de WEB Angular: 13 Horas.
- Documentación: 4 Horas.
- Testing 3 Horas.
- Presentación 3 Horas

Diseño, Componentes y tecnologías de uso

Tecnologías

BBDD SQL Server versión 2022-latest desplegada con Docker.

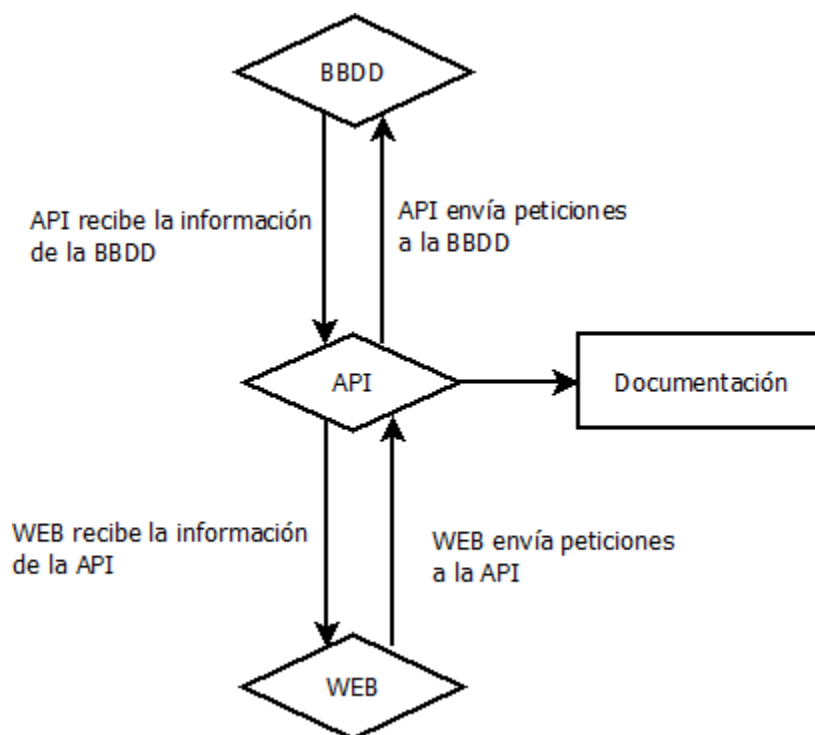
La Página Web con Angular versión 13.1.4.

- Bootstrap5 (CSS3)
- Typescript
- HTML5
- Html-pdf (Librería para generar pdf).

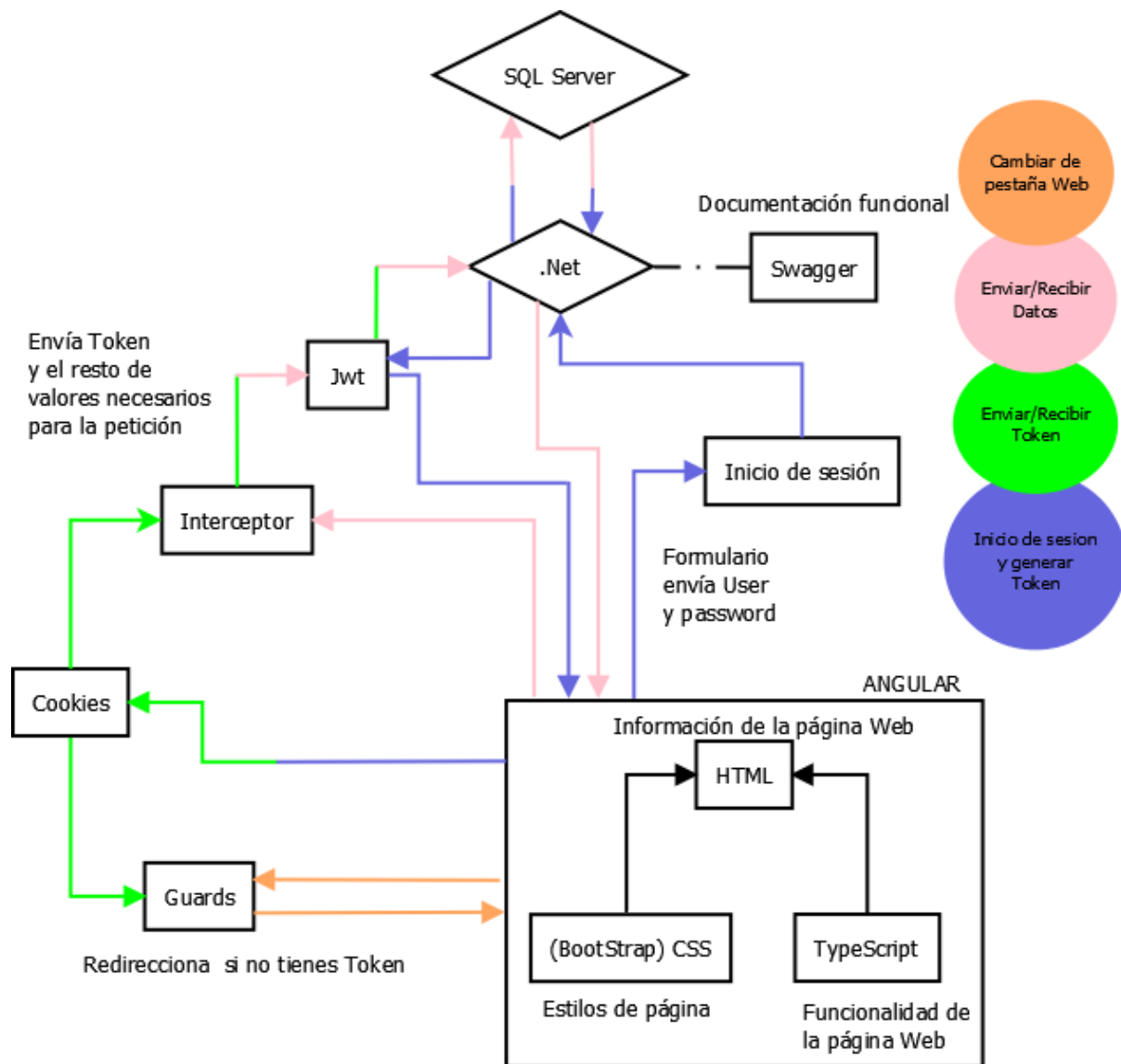
API Rest con C#, .net 6.

- JWT (Login)
- Swagger (Documentación API)

Diseño



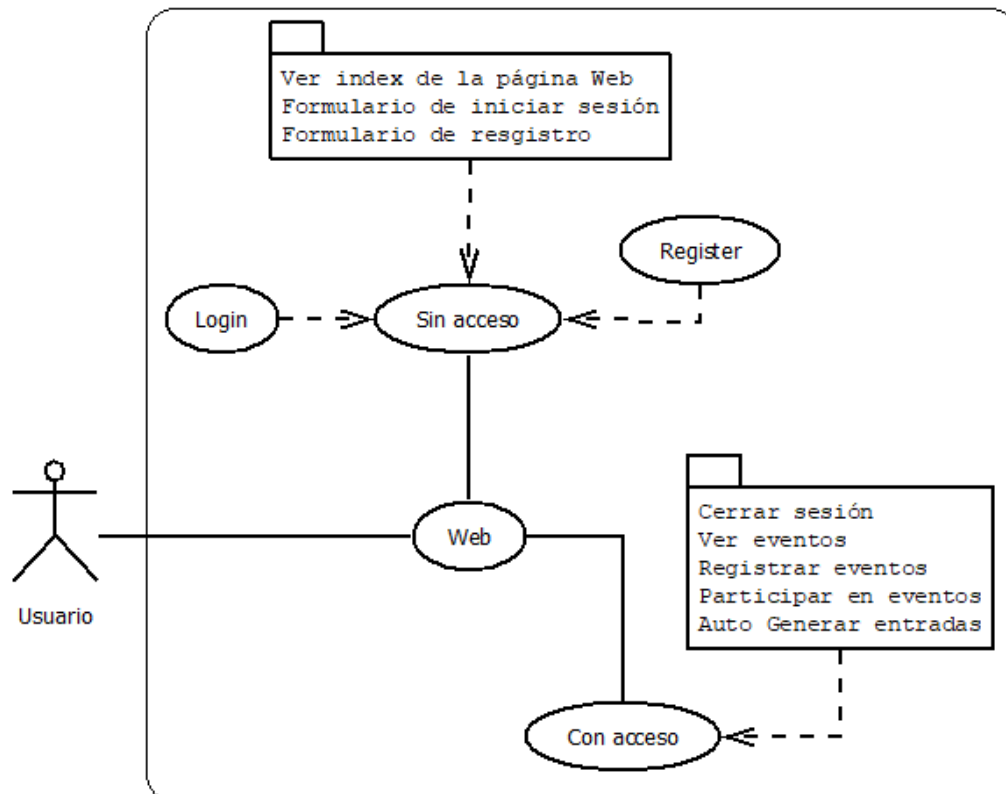
Componentes



Análisis

El proyecto está dividido en tres secciones.

1. BBDD → Almacenar la información.
2. API → Gestiona los intercambios de los datos.
3. Angular → La parte visual donde se muestra toda la información.



Caso de uso	Login
Actor	Usuario
Propósito	Acceder a las áreas privadas de la Web
Pre-condiciones	Tener una cuenta registrada
Flujo básico:	
1. El usuario rellena el formulario	
2. API Comprueba los datos con los de la BBDD	
3. El usuario recibe token (Inicia sesión)	
Fujo alterno:	
1. El usuario rellena el formulario	
2. Se produce un aviso de que los datos están mal.	

Caso de uso	Register
Actor	Usuario
Propósito	Poder iniciar sesión
Pre-condiciones	Ninguna
Flujo básico: 1. El usuario rellenas todos los campos del formulario 2. La API inserta los datos en la BBDD 3. Cuenta creada	
Flujo alternativo: 1. El usuario no rellena todos los campos o escribe datos no válidos 2. El formulario avisa de que los datos no son correctos	

Caso de uso	Índex página Web
Actor	Usuario
Propósito	Ver información sobre el resto de la página
Pre-condiciones	Ninguna
Flujo básico: 1. Visualizas la página, si tienes alguna duda/problema contactas con los administradores con el botón.	
Flujo alternativo: 2. No tienes cuenta de correo agregada en el gestor de correo / No tienes gestor de correo	

Caso de uso	Cerrar sesión
Actor	Usuario
Propósito	Desvincular la cuenta del dispositivo
Pre-condiciones	Tener una cuenta iniciada
Flujo básico: 1. Pulsas el botón 2. Se elimina el token de las cookies.	

Caso de uso	Ver eventos
Actor	Usuario
Propósito	Ver los eventos disponibles y/o finalizados
Pre-condiciones	Tener sesión iniciada
Flujo básico: 1. Entrás en las pestañas de eventos 2. Se muestran los eventos 3. Entrás en el que te interese	

Caso de uso	Participar en eventos
Actor	Usuario
Propósito	Unirte a eventos disponibles
Pre-condiciones	Tener sesión iniciada
Flujo básico: 1. Pulsas el evento que quieres 2. Click en el botón de participar 3. Te redirige a la pestaña de participando	
Flujo alternativo: 1. El botón no se muestra 2. El evento está finalizado o no hay más aforo	

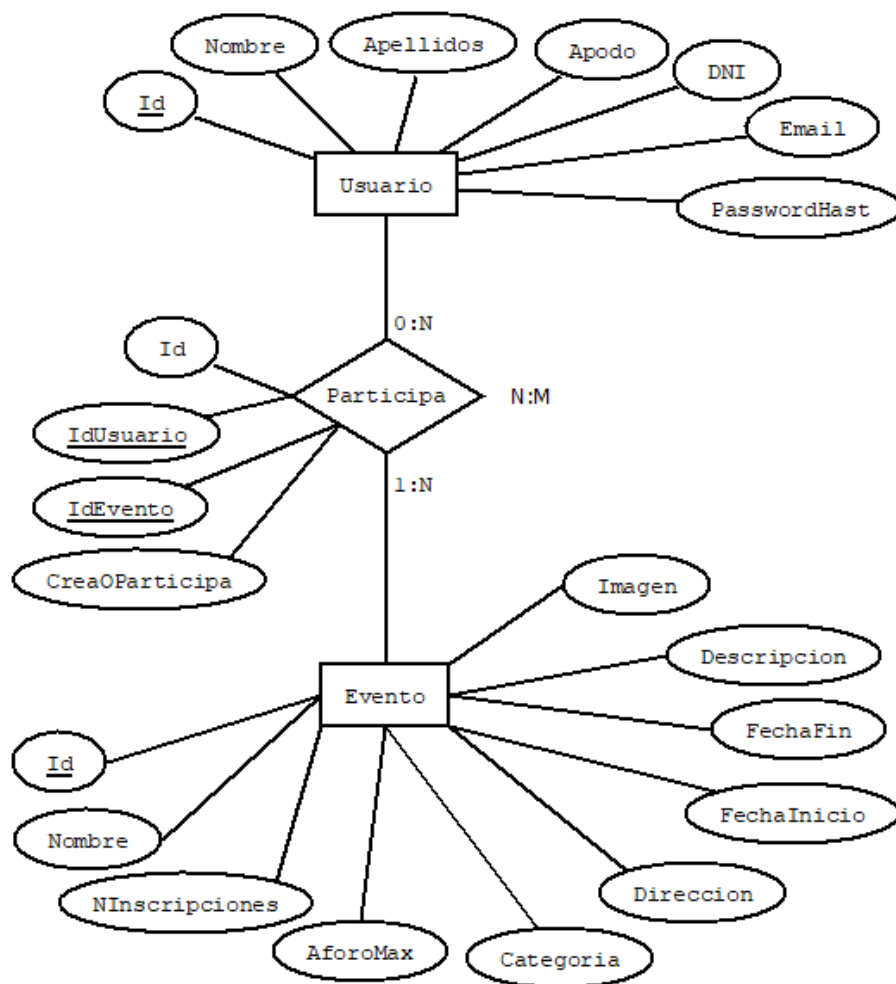
Caso de uso	Registrar eventos
Actor	Usuario
Propósito	Crear eventos disponibles para las personas con cuenta iniciada
Pre-condiciones	Tener sesión iniciada
Flujo básico: <ol style="list-style-type: none"> 1. Rellenas los datos 2. Datos correctos 3. Se envían a la API y de esta a la BBDD 4. Se inserta la persona creadora del evento en la tabla intermedia de la BBDD también desde la API 	
Fujo alterno: <ol style="list-style-type: none"> 1. Rellenas los datos mal 2. Avisa de los errores 	

Caso de uso	Auto generar entradas
Actor	Usuario
Propósito	Generar entradas de los eventos en los cuales participas
Pre-condiciones	Tener sesión iniciada
Flujo básico: <ol style="list-style-type: none"> 1. Estas participando en el evento 2. Pulsas botón 3. Se descarga un PDF con la entrada 	
Fujo alterno: <ol style="list-style-type: none"> 1. No estas participando en el evento 2. No muestra el botón 	

Diseño de la aplicación

BBDD

Diagrama:



La base de datos está diseñada en 2 tablas + 1 para la relación N:M

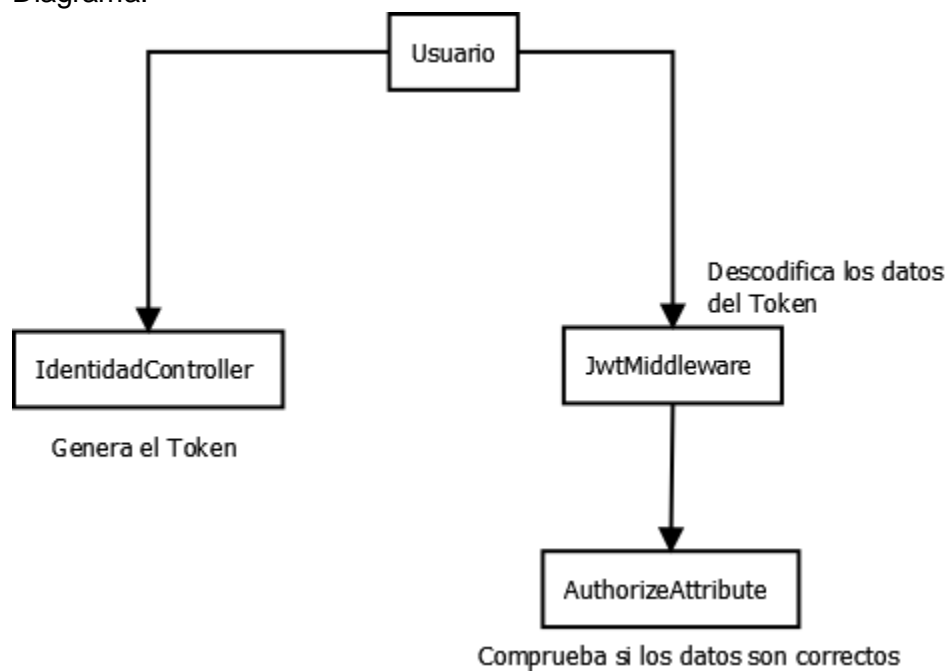
1. Usuario → Almacena los datos del usuario.
2. Participa → Contiene lo necesario para la relación y un campo para señalar si está participando o no a algún evento.
3. Evento → Almacena los datos de los eventos

API

Tiene 2 partes, la primera se encarga de gestionar JWT y la segunda se encarga de las peticiones comunes.

JWT

Diagrama:

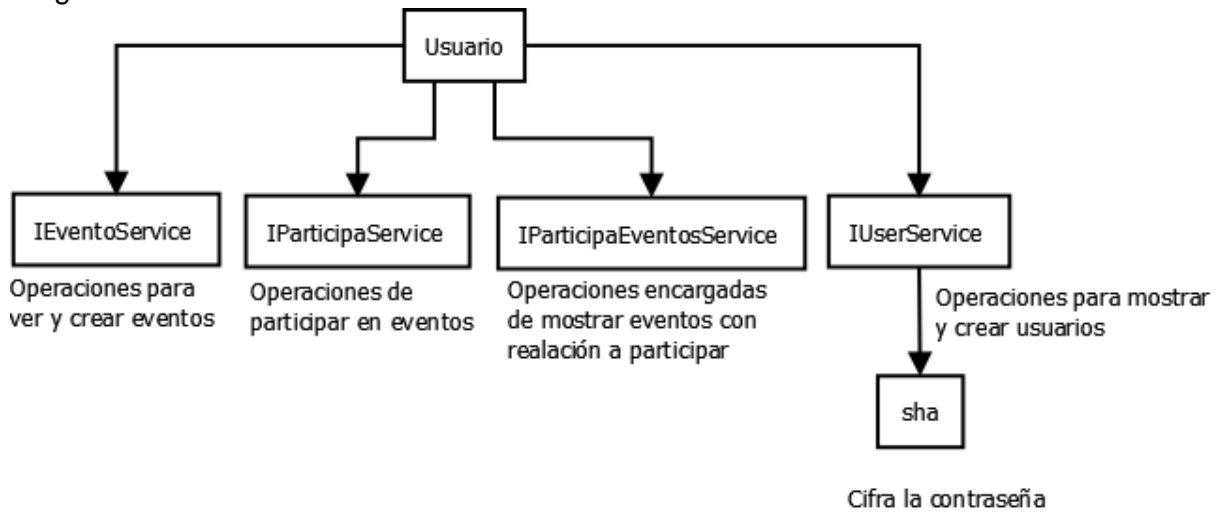


JWT está dividido en varios archivos:

- **IdenticadController** → Genera el Token del usuario.
- **JwtMiddleware** → Descodifica el token para poder utilizar sus datos.
- **AuthorizeAttribute** → Comprueba si el token es válido con los datos de **JwtMiddleware**.

Peticiones

Diagrama:



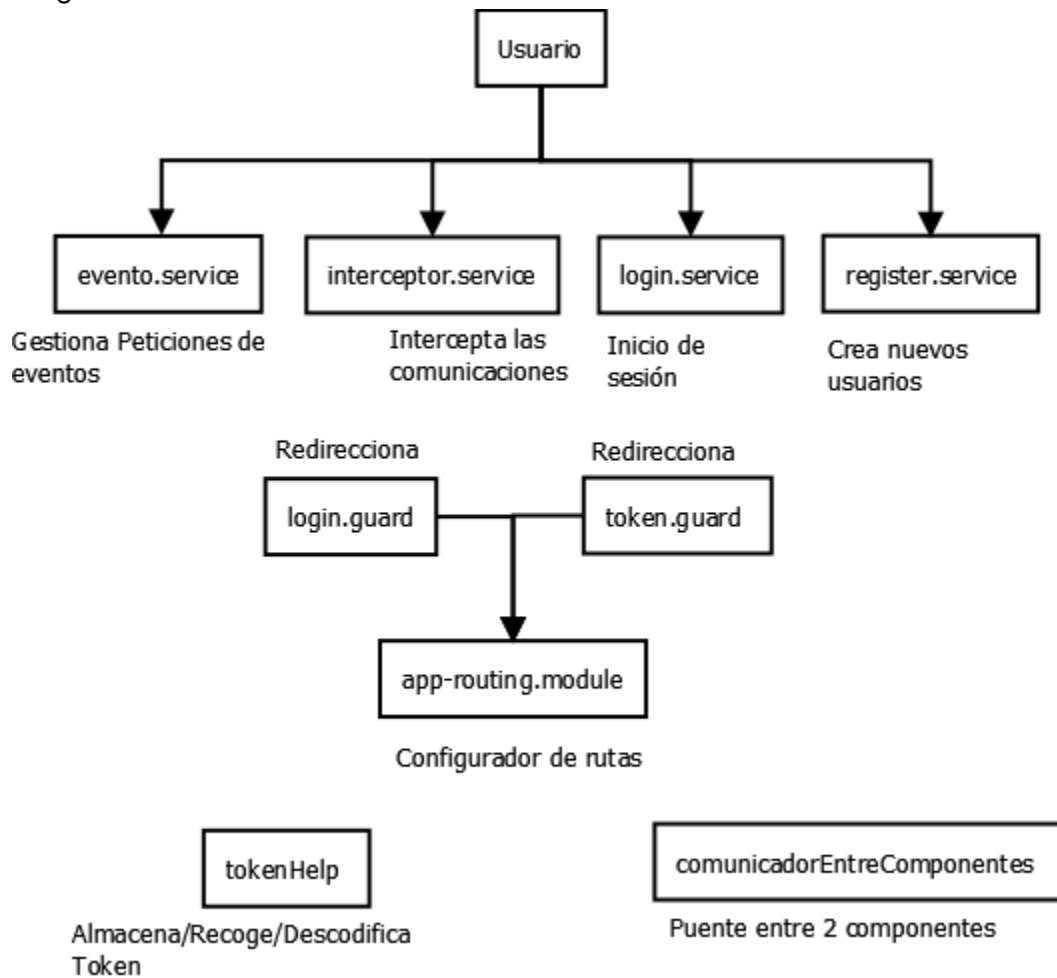
Las peticiones se encargan de enviar y recibir los datos necesarios para el correcto funcionamiento de la aplicación.

Estos son sus archivos:

- IEventoService → Interfaz con las operaciones de eventos simples.
- IParticipaService → Interfaz con la operación de participar en Eventos
- IParticipaEventosService → Interfaz con las operaciones de usuarios que participan en eventos.
- IUserService → Interfaz con las operaciones de gestión de usuarios.
- sha → Es el archivo encargado de cifrar las contraseñas antes de enviarlas a la BBDD.

Angular

Diagrama:



En este apartado muestro las funcionalidades de la página WEB:

Servicios

- **evento.service** → Servicio que gestiona las peticiones de los eventos.
- **interceptor.service** → Servicio que intercepta las comunicaciones.
- **login.service** → Servicio que se encarga de iniciar sesión.
- **register.service** → Servicio encargado de crear nuevos usuarios.

Guards

- token.guard → Servicio encargado de redireccionarte si has iniciado sesión.
- login.guard → Servicio encargado de redireccionarte si no has iniciado la sesión.

Routing

- app-routing.module → Es un servicio que configura las rutas de la página Web.

Helper

- tokenHelp → Es el encargado de almacenar y recoger el Token en las cookies. También es el encargado de descodificar el Token.
- comunicadorEntreComponentes → Sirve entre puente entre 2 componentes, puedes modificar campos de un componente desde otro que lo implemente.

Implementación

BBDD

```
CREATE DATABASE EventMaster;
```

Tabla Usuario:

```
USE EventMaster;

CREATE TABLE Usuario
(
  Id INTEGER IDENTITY(1,1),
  Nombre VARCHAR(50) NOT NULL,
  Apellidos VARCHAR(100) NOT NULL,
  Apodo VARCHAR(60) UNIQUE NOT NULL,
  DNI VARCHAR(9) NOT NULL,
  Email VARCHAR(150) UNIQUE NOT NULL,
  PasswordHash VARCHAR(max) NOT NULL,
  CONSTRAINT Pk_Usuario PRIMARY KEY (Id)
);
```

Esta tabla guarda los datos básicos de cada usuario, Nombre, Apellido, Apodo ...
Además, el campo PasswordHash está cifrado.
La contraseña está cifrada.

Tabla Evento:

```
USE EventMaster;

CREATE TABLE Evento
(
    Id INTEGER IDENTITY(1,1),
    Nombre VARCHAR(150) NOT NULL,
    NInscripciones INTEGER NOT NULL,
    AforoMax INTEGER NOT NULL,
    Categoria VARCHAR(100) NOT NULL,
    Direccion VARCHAR(60) NOT NULL,
    FechaInicio DATETIME NOT NULL,
    FechaFin DATETIME NOT NULL,
    Descripcion VARCHAR(max) NOT NULL,
    Imagen VARCHAR(200) NOT NULL,
    CONSTRAINT Pk_Evento PRIMARY KEY(Id)
);
```

En esta tabla hay varios campos muy importantes:

- Campos NInscripciones y AforoMax para que las personas no puedan unirse si no hay huecos disponibles.
- Categoría para que las personas encuentren de una forma más simple lo que quieren...
- FechaInicio y FechaFin para saber cuándo ha iniciado o finalizado un evento.
- Descripción es una explicación detallada de lo que necesitas para poder participar en el evento.

Tabla Participa

```
USE EventMaster;

CREATE TABLE Participa
(
    Id INTEGER IDENTITY(1,1) UNIQUE,
    IdUsuario INTEGER,
    IdEvento INTEGER,
    CreaOParticipa BIT NOT NULL,
    CONSTRAINT Pk_Participa PRIMARY KEY (IdUsuario,IdEvento),
    CONSTRAINT Fk_Usuario_Participa FOREIGN KEY (IdUsuario) REFERENCES Usuario(Id) ON DELETE CASCADE,
    CONSTRAINT Fk_Evento_Participa FOREIGN KEY (IdEvento) REFERENCES Evento(Id) ON DELETE CASCADE
);
```

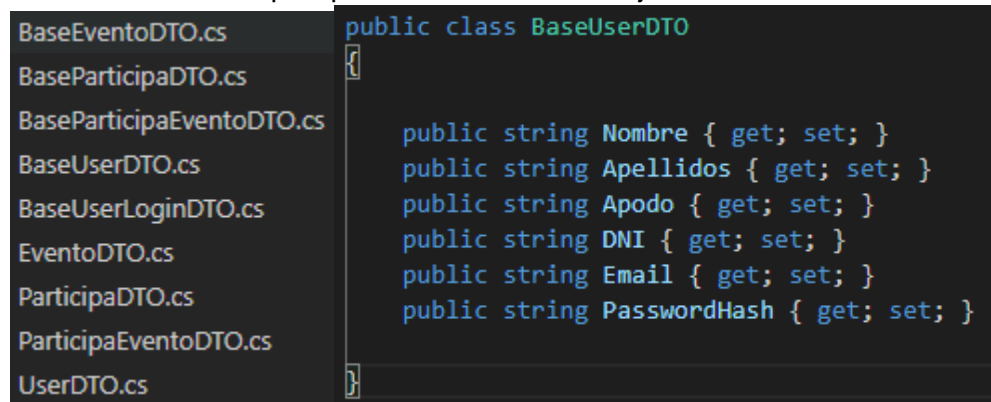
Es el enlace entre las dos tablas, ya que son ManyToMany.

Además, añado un campo bit (1 o 0) llamado que es igual que True o False, a este campo lo llamo CreaOParticipa. Si es 1 (True) significa que está creando un evento, si es 0 (False) significa que está participando en un evento.

API

Toda la API está documentada con Swagger en el propio proyecto.

Para entender con qué tipo de datos están trabajando usan modelos.



```
BaseEventoDTO.cs
BaseParticipaDTO.cs
BaseParticipaEventoDTO.cs
BaseUserDTO.cs
BaseUserLoginDTO.cs
EventoDTO.cs
ParticipaDTO.cs
ParticipaEventoDTO.cs
UserDTO.cs

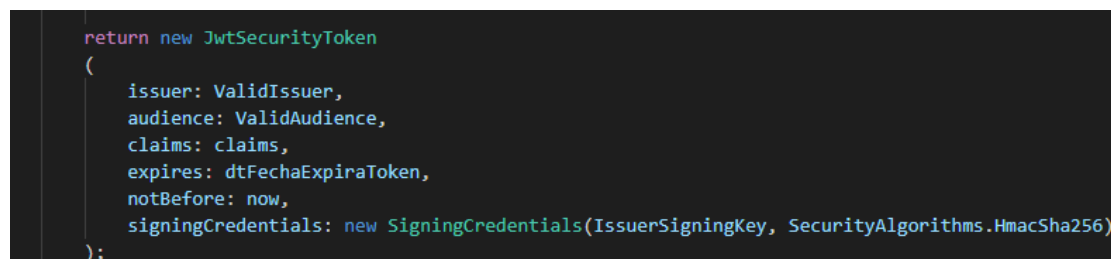
public class BaseUserDTO
{
    public string Nombre { get; set; }
    public string Apellidos { get; set; }
    public string Apodo { get; set; }
    public string DNI { get; set; }
    public string Email { get; set; }
    public string PasswordHash { get; set; }
```

JWT

JWT está dividido en 2 partes, la parte de generar token y la parte de comprobación de si ese token existe.

- Generar

Un usuario tiene que iniciar sesión, sus datos se envían al `IdentidadController` que se encargara de generar un token con un tiempo de expiración determinado.



```
return new JwtSecurityToken
(
    issuer: ValidIssuer,
    audience: ValidAudience,
    claims: claims,
    expires: dtFechaExpiraToken,
    notBefore: now,
    signingCredentials: new SigningCredentials(IssuerSigningKey, SecurityAlgorithms.HmacSha256)
);
```

- Comprobación

El usuario hace una petición a la API enviando el token, el archivo `JwtMiddleware` interceptará la petición si el token es null y la petición es para un componente con el campo "[Authorize]" entonces te devolverá un error, pero si el token es correcto, descodifica el token para poder utilizarlo.

Al terminar la revisión, si el controller al que le solicitas información tiene el campo "[Authorize]" será redirigido a `AuthorizeAttribute` que comprobará que los datos extraídos del token existen y entonces la petición se resolverá sin problemas.


```

[Authorize]
[HttpGet]
[Route("finalizados")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(EventoDTO))]
public ActionResult<EventoDTO> Get()
{
    return Ok(_EventoService.GetEventosFinalizados());
}

```

```

public void OnAuthorization(AuthorizationFilterContext context)
{
    UserDTO user = null;
    try
    {
        user = JsonConvert.DeserializeObject<UserDTO>(context.HttpContext.Items["X-User"].ToString());
    }
    catch (System.Exception)
    {
    }
    if (user == null)
    {
        context.Result = new JsonResult(new { message = "Unauthorized" }) { StatusCode = StatusCodes.Status401Unauthorized };
    }
}

```

Peticiones

Los controladores reciben las peticiones realizadas por los usuarios y estos hacen las comprobaciones necesarias y ejecutan los servicios.

Controladores:

- EventoController → Puede hacer 3 peticiones
 - Get de los datos por ID.
 - Get de los eventos finalizados.
 - Post de un nuevo evento y post en participar para indicar que has creado el evento.
- ParticipaController →
 - Post, el usuario participa en el evento y hace un Put de evento para modificar el número de personas inscritas.
- ParticipaEventosController →
 - Get de los eventos en los que participa el usuario.
 - Get de los eventos en los que no participa el usuario.
- UserController →
 - Get por el apodo del usuario.
 - Post Creando nuevo usuario.

Todos los controladores llaman a los servicios que son necesarios.

Angular

El usuario se mueve por los componentes de la aplicación, las rutas de los componentes están en app-routing.module.

Los guards redireccionan si no puedes estar en la página a la que intentas acceder.

```
const routes: Routes = [
  { path: '', component: HomeComponentComponent },
  { path: 'Login', component: LoginComponent, canActivate: [LoginGuard] },
  {
    path: 'Disponibles',
    component: EventosDisponiblesComponent,
    canActivate: [TokenGuard],
  },
  {
    path: 'Participando',
    component: EventosParticipandoComponent,
    canActivate: [TokenGuard],
  },
  { path: 'Register', component: RegistroComponent, canActivate: [LoginGuard] },
  { path: 'Evento/:id', component: EventoComponent, canActivate: [TokenGuard] },
  {
    path: 'EventoNuevo',
    component: EventoNuevoComponent,
    canActivate: [TokenGuard],
  },
  {
    path: 'eventosFinalizados',
    component: EventosFinalizadosComponent,
    canActivate: [TokenGuard],
  },
],
```

Hay un path marcado con comillas simples, esto indica que es la ruta raíz y te mostrara el componente HomeComponent.

Esta página se podrá visitar sin necesidad de tener ningún permiso en especial, cualquiera puede verla.

Componentes:

- evento.component → Para acceder a este componente tienes que entrar en un de los 3 siguientes componentes y pulsar el evento. Si entras en un evento que este disponible podrás unirte al evento.
- eventos-disponibles.component → Muestra todos los eventos a los cuales puedes apuntarte.
- eventos-finalizados.component → Muestra todos los eventos en los que han finalizado.
- evento-nuevo.component → Formulario para generar nuevos eventos. Cualquier persona con cuenta puede crear nuevos eventos.
- eventos-participando.component → Muestra todos los eventos en los cuales estas participando.
- footer.component → Pie de página de toda la Web.

- nav-bar.component → Cabecera de la Web.
- login.component → Formulario de inicio de sesión.
- registro.component → Registro de usuarios. Es un formulario con los datos del usuario, tiene validaciones para obligar al usuario a escribir los campos y algunas validaciones añadidas.

```
RegisterForm = this.fb.group({
  Nombre: ['', Validators.required],
  Apellidos: ['', Validators.required],
  Usuario: ['', Validators.required],
  DNI: ['', [Validators.required, Validators.minLength(9)]],
  Email: [
    '',
    [
      Validators.required,
      Validators.email,
      Validators.pattern('^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$'),
    ],
  ],
  Pass: ['', [Validators.required, Validators.minLength(6)]],
});
```

- home-component.component → Página principal, en esta vienen integrados 2 componentes con información sobre el resto de la página. Estos componentes son home-main-section.component y home-presentacion.component.

Pruebas

En las pruebas se realizó un video mostrando la página web y todas sus funcionalidades.

- Registro.
- Inicio de Sesión.
- Crear Evento.
- Unirse a eventos.
- Generar entrada en PDF.
- Mostrar Entrada.

Enlace a las pruebas [Video](#)

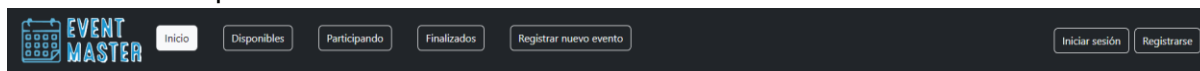
Ampliación y mejoras

- Calendario para filtrar por días.
- Perfil de usuario para poder dar estrellas de los eventos generados.
- Puntuaciones en los eventos finalizados.
- Historial de entradas que has comprado.
- Poder marcar eventos como interesado.
- Auto notificaciones por email.
- Poder editar eventos originados.
- Poder modificar usuarios.
- Roles.

Manual de usuario

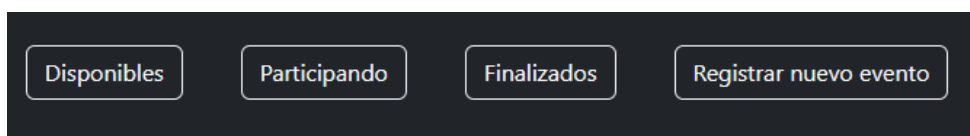
El usuario solo necesita un navegador para utilizar la aplicación. Puede usarla tanto en PC o en dispositivos móviles.

A la hora de emplearla es bastante fácil.



La cabecera nos permite navegar a toda la página de una forma intuitiva. En la página de inicio se muestra una breve explicación de lo que puede hacer.

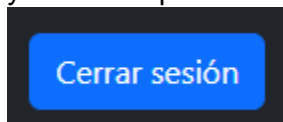
Si no has iniciado sesión no podrás entrar en ninguno de estos apartados.



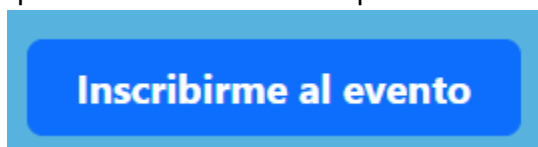
Puedes iniciarte sesión en su respectivo botón, también puedes registrarte si no tienes cuenta.



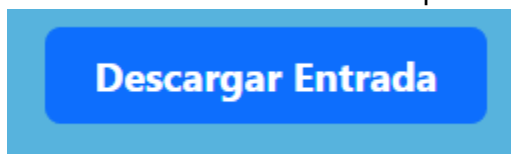
Una vez que has iniciado sesión puedes entrar en los botones que tenías bloqueados antes y también aparece un nuevo botón.



Si entras en “Disponibles” se mostrarán los eventos en los cuales puedes participar. Una vez que te unes al evento desaparece de “Disponibles” y aparecerá en “Participando”.



Al entrar en el evento de nuevo podrás descargar tu entrada.



Se descarga un documento PDF con la entrada.

```
Evento --> Partido de Tenis
Id Entrada --> 2
Categoría --> Deporte
Dirección --> Avenida de los Angeles, S/N, 26500 "La Planilla"
Fecha del inicio del evento --> 25/6/2022, 17:15:00
Fecha del final del evento --> 25/6/2022, 19:20:00
```



Una vez que el evento finalice podrás verlo desde “Finalizados” pero no podrás descargar la entrada.

Puedes crear nuevos eventos desde “Registrar nuevo evento”.

Si tienes algún problema puedes contarme con el botón de la página de inicio.

Manual de instalación

El usuario no tiene que instalar nada, ya que es una aplicación Web.

Instalación servidor

Necesario para la API

Descargar .net 6 → <https://dotnet.microsoft.com/en-us/download/dotnet/6.0>

^ 6.0.5 Security patch

[Release notes](#) Latest release date May 10, 2022

Build apps - SDK

SDK 6.0.300

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	Arm64 x64 x86	Arm64 x64 x86
All	dotnet-install scripts	

Visual Studio support

Visual Studio 2022 (v17.2)

Visual Studio 2022 for Mac (v17.0 latest preview)

Included in

Visual Studio 17.2.0

Included runtimes

Instalador de Microsoft .NET SDK 6.0.300 (x64)

Microsoft .NET SDK 6.0.300

SDK de .NET

El SDK de .NET se usa para compilar, ejecutar y probar las aplicaciones .NET. Puede elegir entre varios lenguajes, editores y herramientas de desarrollo y aprovechar las ventajas de un amplio ecosistema de bibliotecas para compilar aplicaciones web, móviles, de escritorio, juegos e IoT. Esperamos que lo disfrute.

Si tiene previsto usar .NET 6.0 con Visual Studio, se requiere Visual Studio 2022 17.0 o una versión más reciente. [Obtenga más información.](#)

Al hacer clic en Instalar, acepta los términos siguientes.

[Declaración de privacidad](#)

[Recopilación de telemetría y cómo desactivarla](#)

[Información de licencias de .NET](#)

Instalar

Cerrar

Después vas a la carpeta del proyecto con una terminal y ejecutas →

```
MasterEvent\API>dotnet run
```

Si da un error de que no encuentra dotnet entonces sigue [estos pasos](#)

Necesario para Web

Descargar Node.js → <https://nodejs.org/es/download/>

The image shows the Node.js download page. The 'LTS' section is highlighted in green and labeled 'Recomendado para la mayoría'. Below it, there are three download links: 'Instalador Windows' (node-v16.15.1-x64.msi), 'Instalador macOS' (node-v16.15.1.pkg), and 'Código Fuente' (node-v16.15.1.tar.gz). Below these links is a table showing the architecture for each download option.

Download Option	32-bit	64-bit
Instalador Windows (.msi)	32-bit	64-bit
Binario Windows (.zip)	32-bit	64-bit
Instalador macOS (.pkg)	64-bit / ARM64	
Binario macOS (.tar.gz)	64-bit	ARM64
Binario Linux (x64)	64-bit	
Binario Linux (ARM)	ARMv7	ARMv8
Código Fuente	node-v16.15.1.tar.gz	

The image shows the Node.js Setup Wizard. The first screen displays the Node.js logo and the text 'The Setup Wizard will install Node.js on your computer.' Below this is a license agreement window with the text 'Node.js is licensed for use as follows: Copyright Node.js contributors. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: ...'. The 'I accept the terms in the License Agreement' checkbox is checked. The 'Next' button is highlighted. The second screen shows the installation options. The 'Node.js runtime' checkbox is checked. The 'corepack manager', 'npm package manager', 'Online documentation shortcuts', and 'Add to PATH' checkboxes are also checked. The 'Next' button is highlighted. The third screen shows the 'Install' button highlighted.

Después de instalar node.js ve con un terminal a la ruta Web del proyecto y ejecuta →

```
MasterEvent\WEB>npm install
```

En la misma ruta luego ejecutas la Web →

```
MasterEvent\WEB>ng serve
```


Necesario para la BBDD

Hay que instalar Docker en →

<https://docs.docker.com/desktop/windows/install/>



En la instalación pulsamos en todo a siguiente.

https://hub.docker.com/_/microsoft-mssql-server

How to use this Image

Start a mssql-server instance for SQL Server 2022, which is currently in public preview.

```
docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=yourStrong(!)Password" -p 1433:1433 -d mcr.microsoft.com/mssql/server:2022-latest
```

Escogemos la imagen que más nos convenga yo voy a coger la primera.

Ejecutamos una PowerShell con modo administrador

Y ponemos la línea que hemos copiado.

Antes de pulsar enter cambiamos la contraseña a la que queramos que tenga la BBDD yo voy a poner la que tengo en la API en el archivo appsettings.

```
docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=j+K>Ij1E>ng9CNV>" -p 1433:1433 -d mcr.microsoft.com/mssql/server:2022-latest
```

Y pulsamos enter

```
m022-latest: Pulling from mssql/server
15fd17ec1767: Pull complete
223d0d87c158: Pull complete
1bdde79cee40: Pull complete
8644101ecaf8: Pull complete
1e9d020b44f4: Pull complete
Digest: sha256:85495b6c68b58a81a31d37f7e1e1d36384a75baf28bd2b16c77faa5905838126
Status: Downloaded newer image for mcr.microsoft.com/mssql/server:2022-latest
15162b015fcb98e99222c0986d4e26ba2cbcae85b529be37ef08e4d4925bcca
PS C:\Windows\system32>
```

Cuando termine la instalación se ejecutará de forma automática el servidor SQL server.

Si no se ejecuta de manera automática se puede ejecutar el contenedor de manera manual en el cliente de Docker.

Para terminar, ejecutamos el script .sql dentro del proyecto en la carpeta BBDD.
El script lo ejecutamos con un gestor de BBDD. La dirección es localhost con usuario sa y contraseña la que has escrito en el paso anterior en el PowerShell.

Extra

Esto es toda la instalación, pero tengo que añadir que para ver la aplicación hay que entrar desde →

<http://localhost:4200/>

Para que la API funcione correctamente hay que entrar en → <https://localhost:7199/swagger> y aceptar el certificado (detecta que no es seguro porque no tiene certificado SSL firmado).

Si quieres abrir la aplicación en red local necesitas modificar estos 4 ficheros →

`{ } appsettings.json M` `{ } launchSettings.json M X` `TS environment.prod.ts M` `TS environment.ts M`

Cambiar localhost por tu IP actual →

```
{
  "ApiAuth": {
    "Issuer": "https://localhost:7199/",
    "Audience": "Event Master",
    "SecretKey": "82e1hjdсаajd32100d32daskd9320jdkad29as"
  }
}
```

```
"profiles": {
  "Api": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "launchUrl": "swagger",
    "applicationUrl": "https://localhost:7199;http://localhost:5114",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  }
},
```

```
export const environment = {
  production: true,
  API_URL: "https://localhost:7199/"
};
```

```
export const environment = {
  production: false,
  API_URL: "https://localhost:7199/"
};
```

y ejecutar angular con →

```
MasterEvent\WEB>ng serve --host=192.168.1.12
```

Cálculo de tiempo final

- Diseño y creación de la Base de datos: 2 Horas.

- Diseño y creación de la API: 58 Horas.
- Diseño y creación de WEB Angular: 40 Horas.
- Documentación: 13 Horas.
- Testing 8 Horas.
- Presentación 2 Horas

Conclusión

En líneas generales, he terminado la mayoría de los objetivos mencionados en los "[Objetivos](#)".

He creado un gestor de eventos funcional. Se puede pulir alguna parte de la aplicación, pero para una demo es suficientemente funcional.

Bibliografía

[Generar PDF desde NodeJS](#)

[Documentación PDF](#)

[Creación de un sistema de login en Angular](#)

[Ejemplo de realización de login con .net y angular](#)

[Segundo Ejemplo de login con .net y angular](#)

[Cifrar contraseñas para la base de datos](#)

[Protegiendo vistas con guards](#)

[Instalar BootStrap en angular](#)

[Estilos Página Web](#)

[Validar formularios Angular](#)

[Authorization Swagger](#)

Anexos

API = Es un intermediario, lo puedes utilizar mediante peticiones HTTP/HTTPS. Se encarga de enviar y recibir información de la BBDD.

Angular = Un framework para aplicaciones web desarrollado en TypeScript.

Swagger = Un conjunto de herramientas de software de código abierto para diseñar, construir, documentar, y utilizar servicios web RESTful (API).

Framework = un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto.

ManyToMany = Es un tipo de unión entre tablas, también conocido como N:M.

Docker = Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente.

HTTP/HTTPS= Es un protocolo de transferencia de hiper textos. Mediante un enlace manda una petición al servidor para devolver una respuesta al cliente.

Tipos de peticiones de API:

Get	Recogen información de la BBDD
Post	Envían información a la BBDD
Put	Modifican información de la BBDD
Delete	Elimina información de la BBDD