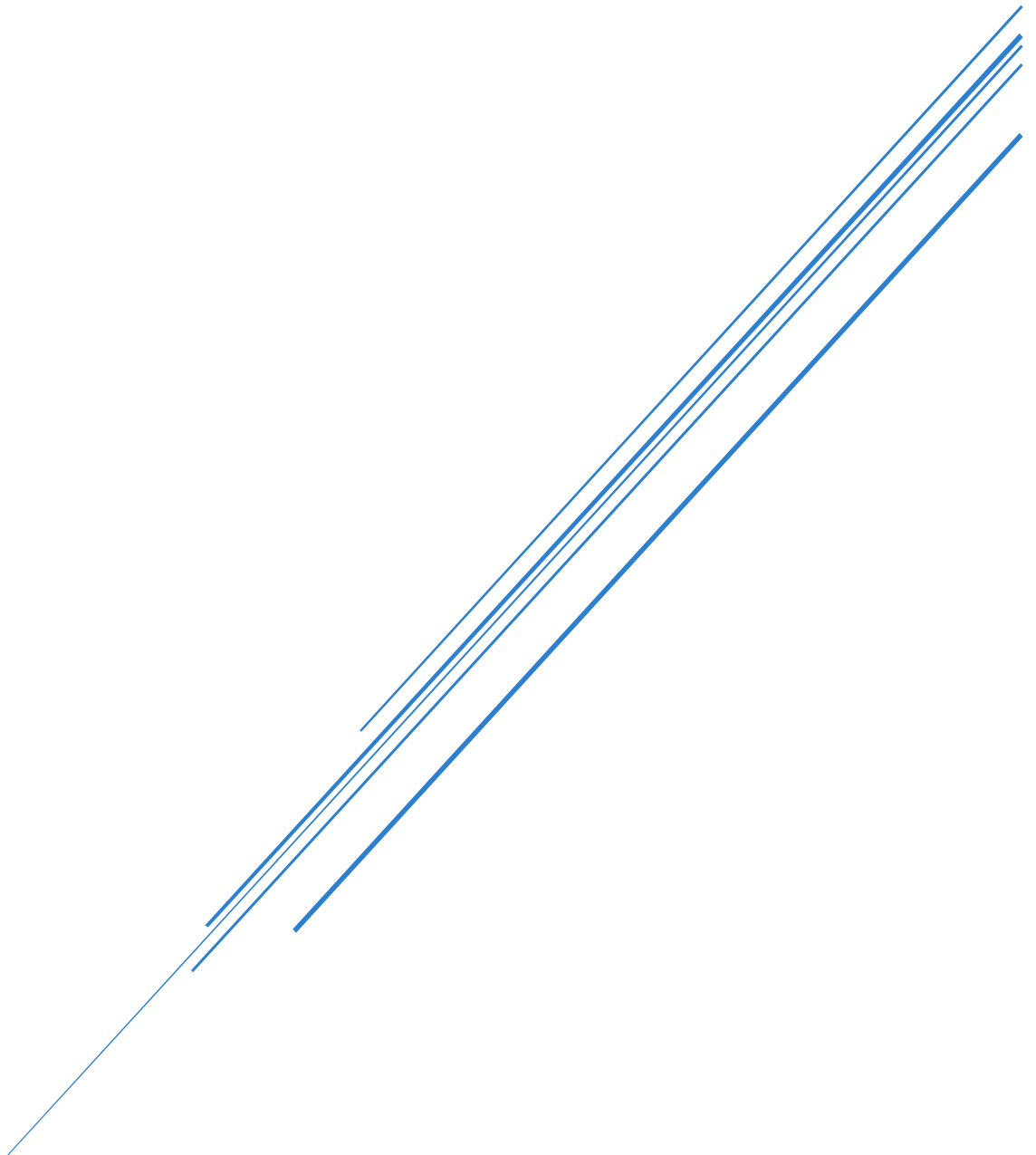


PROYECTECA

Brisa Suárez

<https://github.com/brisasp/ProjectEca.git>



I.E.S COMERCIO
2ºDAMV

ÍNDICE

INTRODUCCIÓN.....	2
OBJETIVOS	2
DISEÑO	2
API	4
SWAGGER.....	14
VISTAS.....	16
POSIBLES MEJORAS.....	34

INTRODUCCIÓN

Este proyecto tiene como finalidad desarrollar un sistema para la gestión del Aula Ateca del IES Comercio, buscando así sustituir el sistema actual que se usa.

Se basa en:

- API .net codefirst que gestiona toda la información
- BBDD SQL Server
- WPF para gestionar las solicitudes de reservas mediante un administrador, definir días no lectivos y modificar horarios
- Angular para la reserva Aula, interfaz destinada para los profesores, permite consultar horarios y realizar solicitudes de reserva

OBJETIVOS

Los objetivos de este proyecto son varios:

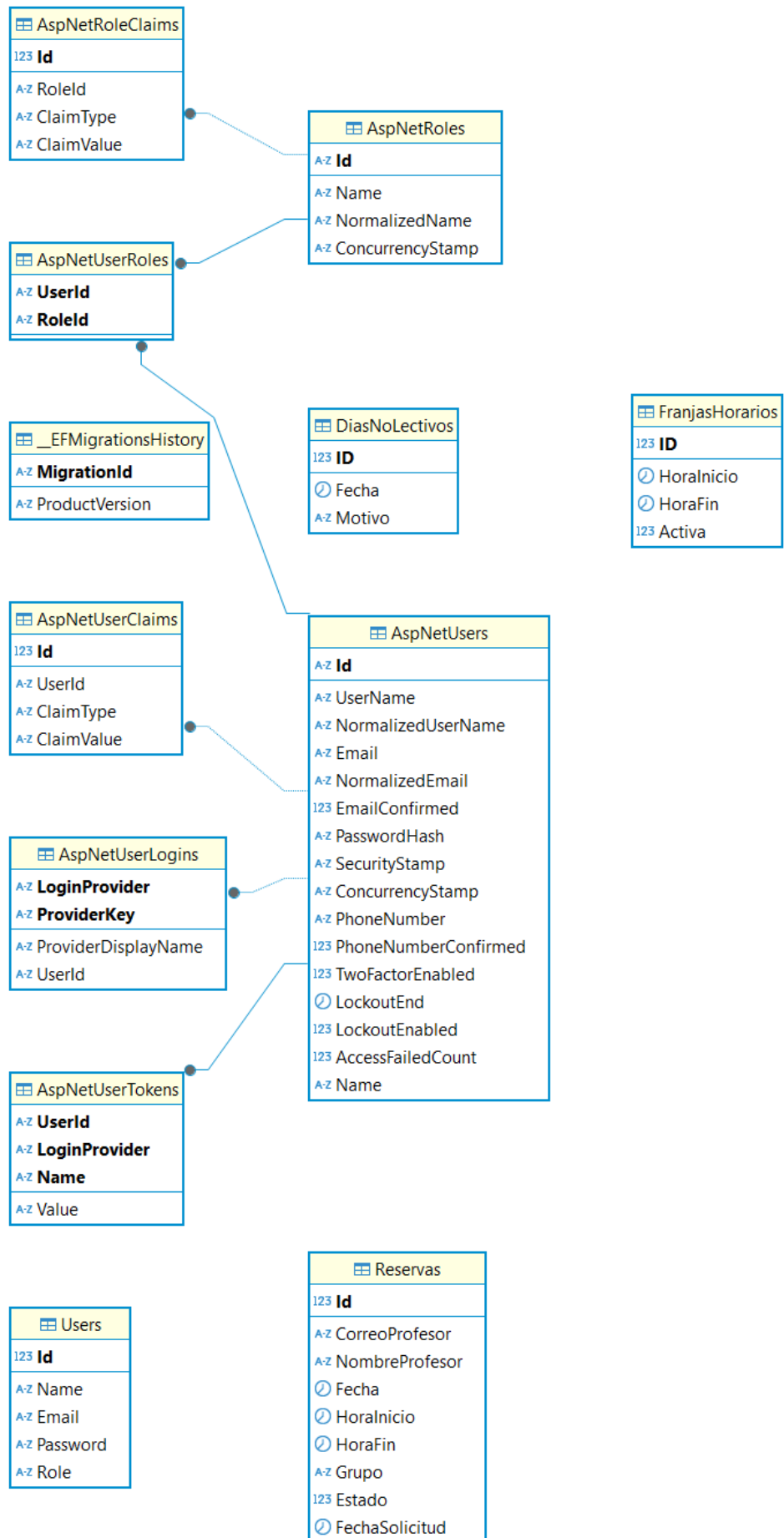
- Eliminar la gestión manual en Excel
- Reducir la carga administrativa
- Garantizar mayor precisión en la planificación de reservas
- Permitir que los administradores gestionen reservas a través de una aplicación de escritorio
- Evitar reservas en días no lectivos y sobre reservas ya existentes
- Garantizar seguridad y control de acceso a la API

DISEÑO

BBDD-

La base de datos del sistema está diseñada utilizando Entity Framework Core con enfoque Code First. Posteriormente, se generan las migraciones para crear y actualizar la estructura en SQL Server.

- Users: **ID** (int, PK), Name, Email, Password, Role
- FranjasHorarios; **ID** (int, PK), HoraInicio, HoraFin, Activa
- Reservas: **ID** (int, PK), CorreoProfesor, NombreProfesor, Fecha, HoraInicio, HoraFin, Grupo, Estado, FechaSolicitud.
- DiasNoLectivos: **ID** (int, PK), Fecha, Motivo



API

La API está desarrollada con ASP.NET Core y sigue un enfoque RESTful. Se encarga de gestionar reservas del Aula AtecA, incluyendo usuarios, roles, franjas horarias, días no lectivos...

Archivo launchSettings

```
1 {
2   "$schema": "https://json.schemastore.org/launchsettings.json",
3   "iisSettings": {
4     "windowsAuthentication": false,
5     "anonymousAuthentication": true,
6     "iisExpress": {
7       "applicationUrl": "http://localhost:65493",
8       "sslPort": 44390
9     }
10  },
11  "profiles": {
12    "http": {
13      "commandName": "Project",
14      "dotnetRunMessages": true,
15      "launchBrowser": true,
16      "launchUrl": "swagger",
17      "applicationUrl": "http://localhost:5072",
18      "environmentVariables": {
19        "ASPNETCORE_ENVIRONMENT": "Development"
20      }
21    },
22    "https": {
23      "commandName": "Project",
24      "dotnetRunMessages": true,
25      "launchBrowser": true,
26      "launchUrl": "swagger",
27      "applicationUrl": "https://localhost:7016;http://localhost:5072",
28      "environmentVariables": {
29        "ASPNETCORE_ENVIRONMENT": "Development"
30      }
31    },
32    "IIS Express": {
33      "commandName": "IISExpress",
34      "launchBrowser": true,
35      "launchUrl": "swagger",
36      "environmentVariables": {
37        "ASPNETCORE_ENVIRONMENT": "Development"
38      }
39    }
40  }
41 }
```

Este archivo se usa para definir configuraciones de inicio del proyecto, permitiendo que se inicie automáticamente en diferentes entornos y configuraciones.

Los diferentes perfiles configurados son estos

Nombre del perfil	URL
http	http://localhost:5072
https	https://localhost:7016

Archivo PasswordValidationAttribute

```

1 namespace DesignAPI.Attributes
2 {
3     using System.ComponentModel.DataAnnotations;
4     using System.Linq;
5     public class PasswordValidationAttribute : ValidationAttribute
6     {
7         public override bool IsValid(object value)
8         {
9             if (value is not string password)
10                 return false;
11             // Check password length (8-20 characters)
12             if (password.Length < 8 || password.Length > 20)
13                 return false;
14
15             // Check for at least one number
16             if (!password.Any(char.IsDigit))
17                 return false;
18
19             // Check for at least one lowercase letter
20             if (!password.Any(char.IsLower))
21                 return false;
22
23             // Check for at least one uppercase letter
24             if (!password.Any(char.IsUpper))
25                 return false;
26
27             // Check for at least one symbol
28             var symbols = @"!""#$%&'()*+,-./:;<=>@[\\]^_`{|}~";
29             if (!password.Any(c => symbols.Contains(c)))
30                 return false;
31
32             return true;
33         }
34         public override string FormatErrorMessage(string name)
35         {
36             return $"{name} must be 8-20 characters long and include at least one uppercase letter, one lowercase letter, one number, and one symbol."
37         }
38     }
39 }
```

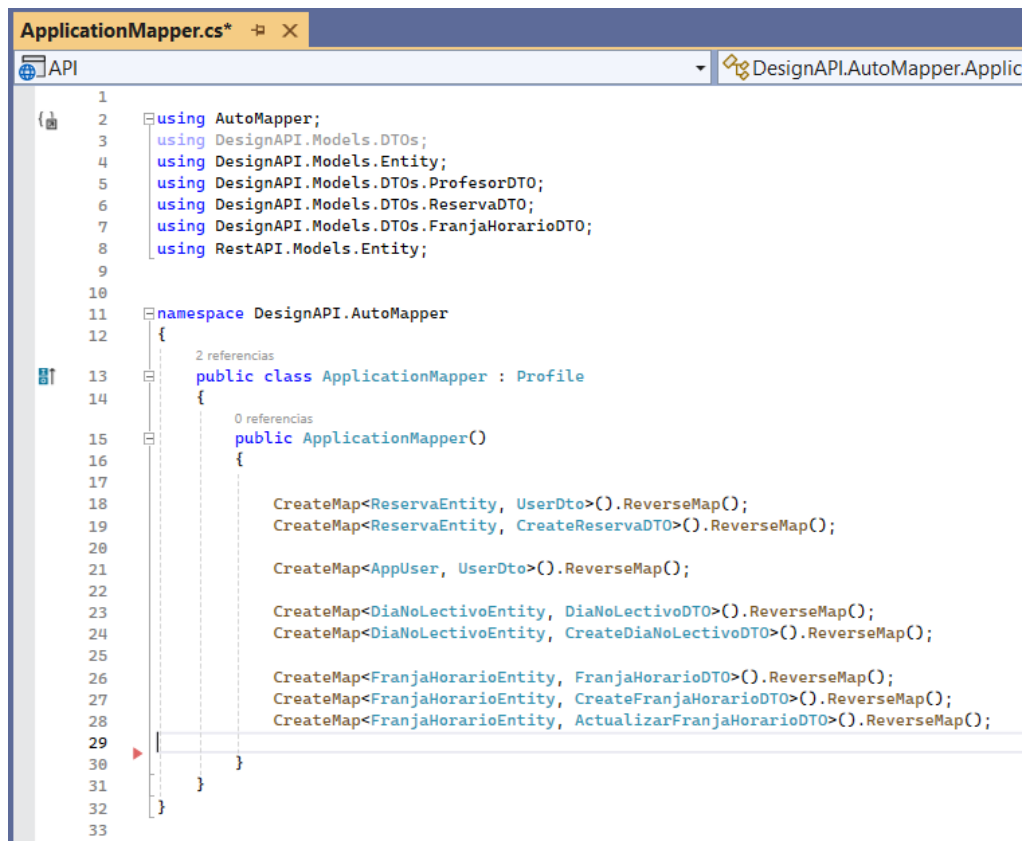
El sistema implementa una validación personalizada para las contraseñas de los usuarios mediante la clase PasswordValidationAttribute

Requisito	Condición
Longitud	Entre 8 y 20 caracteres
Número	Al menos un dígito (0-9)
Minúscula	Al menos una letra minúscula (a-z)
Mayúscula	Al menos una letra mayúscula (A-Z)
Símbolo especial	Al menos un carácter de la lista `!"#\$%&'()*+,-./:;<=>@[\\]^_`{ }~

Si alguno de estos requisitos no se cumple, el sistema devuelve el siguiente mensaje de error:

"{campo} must be 8-20 characters long and include at least one uppercase letter, one lowercase letter, one number, and one symbol."

Clase ApplicationMapper



```
1
2 using AutoMapper;
3 using DesignAPI.Models.DTOs;
4 using DesignAPI.Models.Entity;
5 using DesignAPI.Models.DTOs.ProfesorDTO;
6 using DesignAPI.Models.DTOs.ReservaDTO;
7 using DesignAPI.Models.DTOs.FranjaHorarioDTO;
8 using RestAPI.Models.Entity;
9
10
11 namespace DesignAPI.AutoMapper
12 {
13     public class ApplicationMapper : Profile
14     {
15         public ApplicationMapper()
16         {
17
18             CreateMap<ReservaEntity, UserDto>().ReverseMap();
19             CreateMap<ReservaEntity, CreateReservaDTO>().ReverseMap();
20
21             CreateMap<AppUser, UserDto>().ReverseMap();
22
23             CreateMap<DiaNoLectivoEntity, DiaNoLectivoDTO>().ReverseMap();
24             CreateMap<DiaNoLectivoEntity, CreateDiaNoLectivoDTO>().ReverseMap();
25
26             CreateMap<FranjaHorarioEntity, FranjaHorarioDTO>().ReverseMap();
27             CreateMap<FranjaHorarioEntity, CreateFranjaHorarioDTO>().ReverseMap();
28             CreateMap<FranjaHorarioEntity, ActualizarFranjaHorarioDTO>().ReverseMap();
29         }
30     }
31 }
32
33
```

AutoMapper sirve para transformar automáticamente objetos entre capas sin necesidad de escribir código repetitivo, facilitando así el manejo de objetos en la API

Entidad origen	DTO destino	Dirección
ReservaEntity	UserDto	Bidireccional
ReservaEntity	CreateReservaDTO	Bidireccional
AppUser	UserDto	Bidireccional
DiaNoLectivoEntity	DiaNoLectivoDTO	Bidireccional
DiaNoLectivoEntity	CreateDiaNoLectivoDTO	Bidireccional
FranjaHorarioEntity	FranjaHorarioDTO	Bidireccional
FranjaHorarioEntity	CreateFranjaHorarioDTO	Bidireccional
FranjaHorarioEntity	ActualizarFranjaHorarioDTO	Bidireccional

Se definen mapeos para reservas, usuarios, DiasNoLectivos y FranjaHorario, asegurando que los datos se transfieran correctamente entre la base de datos y las respuestas de la API.

EMAILSERVICE + EMAILSETTINGS

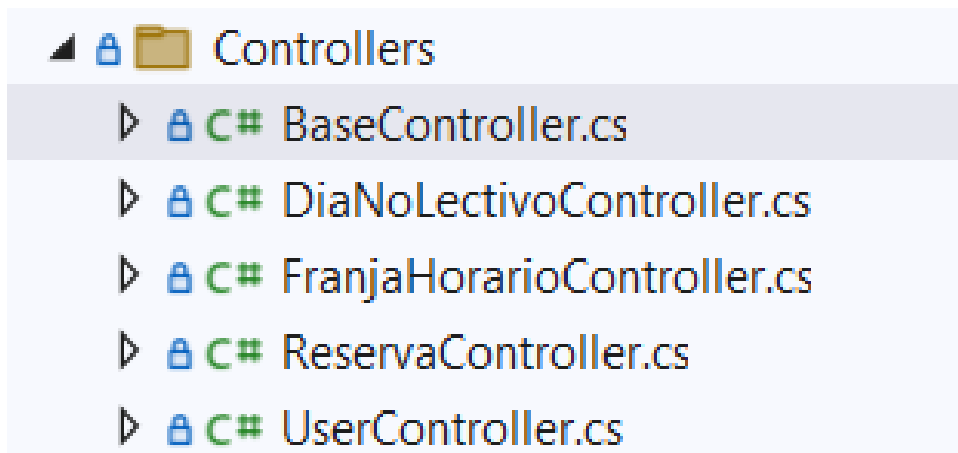
```
EmailService.cs*  + X
API
4
5
6 6 referencias
7 public class EmailService
8 {
9     private readonly IConfiguration _config;
10
11     0 referencias
12     public EmailService(IConfiguration config)
13     {
14         _config = config;
15     }
16
17     2 referencias
18     public async Task EnviarCorreo(string destinatario, string asunto, string cuerpo)
19     {
20         try
21         {
22             var from = _config["EmailSettings:From"];
23             var password = _config["EmailSettings:Password"];
24             var host = _config["EmailSettings:SmtpServer"];
25             var port = int.Parse(_config["EmailSettings:Port"]);
26
27             var mensaje = new MailMessage
28             {
29                 From = new MailAddress(from),
30                 Subject = asunto,
31                 Body = cuerpo,
32                 IsBodyHtml = true
33             };
34             mensaje.To.Add(destinatario);
35
36             var smtp = new SmtpClient(host, port)
37             {
38                 Credentials = new NetworkCredential(from, password),
39                 EnableSsl = true
40             };
41
42             await smtp.SendMailAsync(mensaje);
43         }
44         catch (Exception ex)
45         {
46             Console.WriteLine($"❌ Error al enviar el correo: {ex.Message}");
47             throw;
48         }
49     }
50 }
```

```
EmailSettings.cs  + X
API
1 using System.Net.Mail;
2 using System.Net;
3 namespace API.Configuration;
4
5 1 referencia
6 public class EmailSettings
7 {
8     0 referencias
9     public string Correo { get; set; }
10
11     0 referencias
12     public string Password { get; set; }
13 }
```

La API incluye un servicio personalizado para el envío de correos electrónicos mediante SMTP, con el objetivo de notificar al administrador sobre nuevas solicitudes de reserva o rechazos de las mismas.

Se invoca tras crear una reserva (POST /api/reserva) o tras rechazarla en el método Update del BaseController.

CONTROLLERS



BaseController: Clase genérica utilizada como base para el resto de controladores.

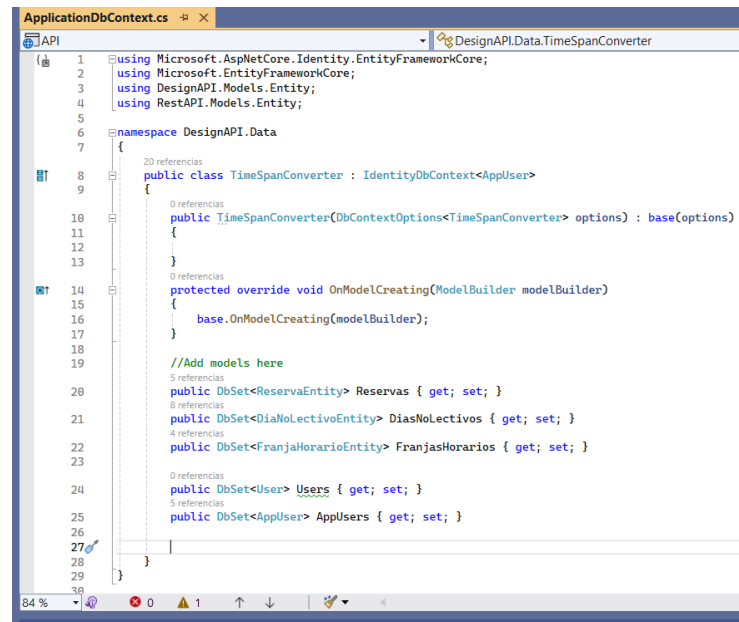
DiaNoLectivoController: Gestiona los días en los que no se permiten reservas, permite al administrador crear, consultar y eliminar días no lectivos y hereda de BaseController

FranjaHorarioController: permite definir las franjas horarias disponibles para las reservas, se puede crear, editar y consultar dichas franjas, también hereda de BaseController

ReservaController: Hereda de BaseController, devuelve reservas con estado Pendiente, devuelve reservas del usuario autenticado por el correo, crea nuevas reservas validando que se cumplan los requisitos como que no sea un día no lectivo ni que exista duplicidad de reservas, además se ocupa de también de enviar la notificación por correo

UserController: Maneja el inicio de sesión y valida la contraseña

APPLICATIONDBCONTEXT



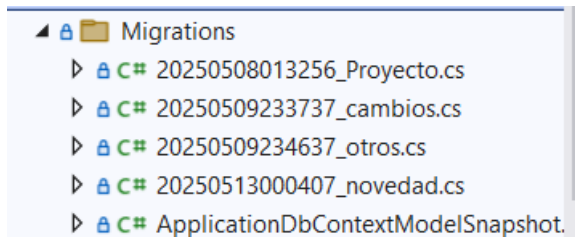
Es el contexto principal de base de datos de la API. Este contexto es responsable de exponer las tablas de la base de datos mediante propiedades DbSet<>

- Reservas
Representa la tabla que almacena las solicitudes de reserva realizadas por los usuarios.
- Días No Lectivos
Contiene los días marcados como no lectivos, sobre los cuales no se pueden realizar reservas.
- Franjas Horarias
Define los intervalos horarios disponibles para reservar el Aula AtecA.

Al heredar de IdentityDbContext<AppUser>, el contexto incluye automáticamente todas las tablas necesarias para la autenticación y autorización de usuarios (como AspNetUsers, AspNetRoles, AspNetUserRoles, etc.), permitiendo así el uso de roles y login personalizado.

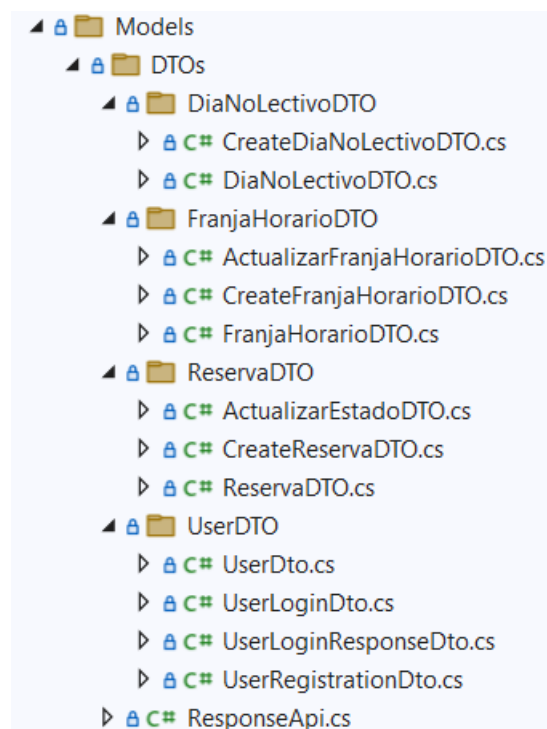
MIGRATIONS

La gestión del esquema de la base de datos en este proyecto se realiza mediante migraciones de Entity Framework Core, siguiendo un enfoque Code First.



Son todas las evoluciones de la base de datos a lo largo del desarrollo de este proyecto

DTOs



Esto sirve para estructurar mejor la información que se envía y recibe desde la API, generando así seguridad mayor flexibilidad y facilitando el mantenimiento del código

- **DiaNoLectivoDTO**
Contiene objetos para gestionar días no lectivos en el sistema:
CreateDiaNoLectivoDto, DiaNoLectivoDto
- **FranjaHorarioDTO**
Incluye objetos relacionados con la gestión de franjas horarias:
CreateFranjaHorarioDTO, ActualizarFranjaHorarioDTO, FranjaHorarioDto

- **ReservaDTO**

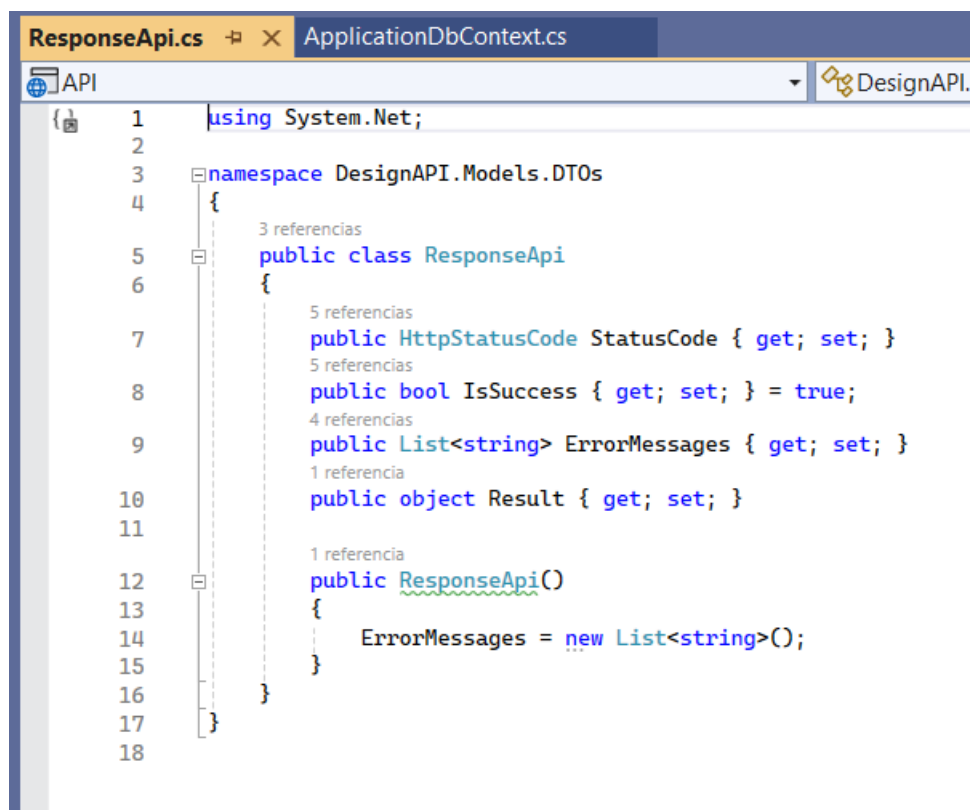
Contiene objetos necesarios para la gestión de reservas:
CreateReservaDTO, ActualizarEstadoDTO, ReservaDTO

- **UserDTO**

Modela los datos de los usuarios en operaciones de autenticación y registro:

UserDTO, UserLoginDto, UserLoginResponseDto, UserRegistrationDto

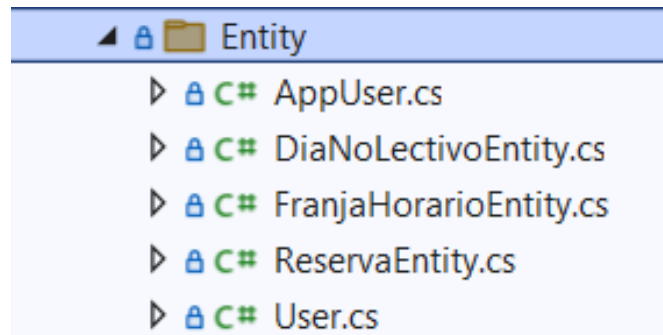
RESPONSEAPI



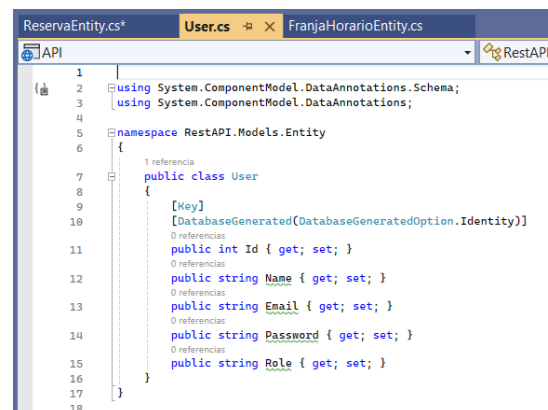
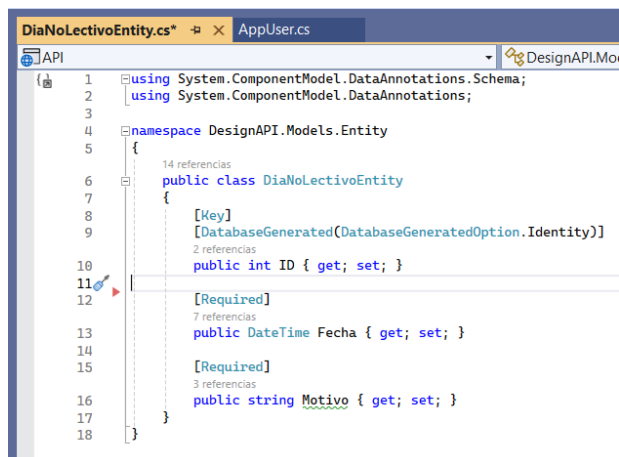
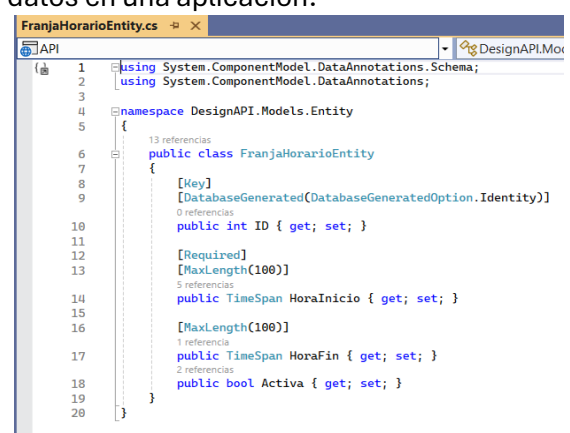
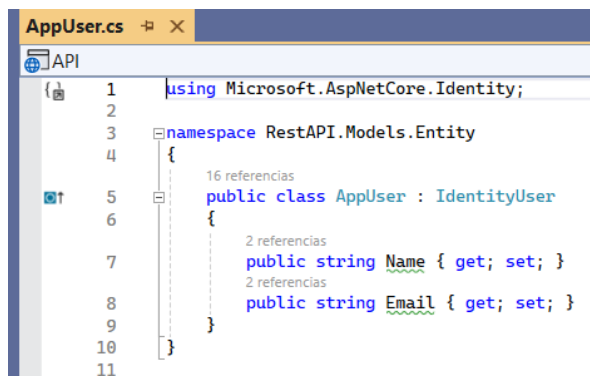
```
1 using System.Net;
2
3 namespace DesignAPI.Models.DTOs
4 {
5     3 referencias
6     public class ResponseApi
7     {
8         5 referencias
9         public HttpStatusCode StatusCode { get; set; }
10        5 referencias
11        public bool IsSuccess { get; set; } = true;
12        4 referencias
13        public List<string> ErrorMessage { get; set; }
14        1 referencia
15        public object Result { get; set; }
16
17        1 referencia
18        public ResponseApi()
19        {
20            ErrorMessage = new List<string>();
21        }
22    }
23 }
```

Archivo que define una estructura estándar para devolver respuestas desde los controladores, incluyendo el estado de la operación, mensajes y contenido opcional.

ENTITY

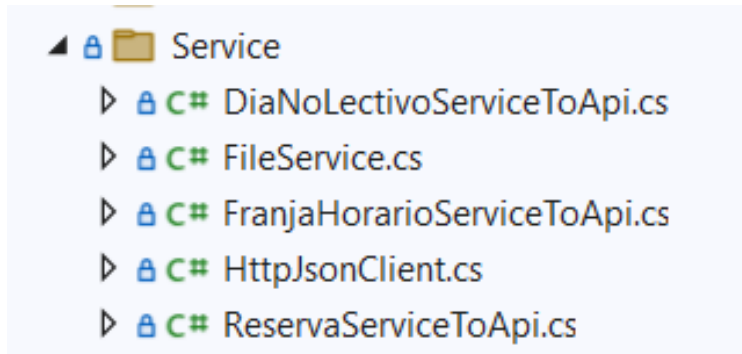


Estas clases sirven para representar la estructura de un alumno dentro de la base de datos en una aplicación.



Esta clase representa la entidad DiaNoLectivo en la base de datos. Define los atributos de un Dia no lectivo y sus restricciones, asegurando la integridad de los datos

WPF



DiaNoLectivoServiceToApi:

Servicio encargado de la comunicación con la API para la gestión de los días no lectivos. Permite consultar, crear, modificar y eliminar días no lectivos desde la aplicación cliente.

Obtiene la lista de días no lectivos, envía nuevas fechas no lectivas a la API y actualiza o borra días no lectivos ya existentes.

FileService:

Servicio para gestión de archivos, encargado de operaciones relacionadas con carga, descarga y manejo de archivos dentro de la aplicación.

Valida formatos y tamaños de archivo

FranjaHorarioServiceToApi:

Servicio que comunica con la API para la gestión de franjas horarias. Permite manejar la disponibilidad y restricciones horarias en la aplicación.

Consulta franjas horarias activas, crea, modifica y elimina franjas horarias

HttpJsonClient:

Envía peticiones a la API y recibe respuestas JSON

ReservaServiceToApi:

Servicio para la gestión de reservas mediante comunicación con la API. Facilita operaciones sobre reservas como crear, consultar y modificar.

Obtiene reservas por usuario o estado, crea nuevas reservas, modifica o cancela reservas existentes.

SWAGGER

DiaNoLectivo			^
GET	/api/DiaNoLectivo/{id}		✓ 🔒
PUT	/api/DiaNoLectivo/{id}		✓ 🔒
GET	/api/DiaNoLectivo		✓ 🔒
POST	/api/DiaNoLectivo		✓ 🔒
FranjaHorario			^
GET	/api/FranjaHorario/activos		✓ 🔒
GET	/api/FranjaHorario/disponibles		✓ 🔒
GET	/api/FranjaHorario/{id}		✓ 🔒
PUT	/api/FranjaHorario/{id}		✓ 🔒
GET	/api/FranjaHorario		✓ 🔒
POST	/api/FranjaHorario		✓ 🔒
Reserva			^
GET	/api/Reserva/pendientes		✓ 🔒
GET	/api/Reserva/mis-reservas		✓ 🔒
POST	/api/Reserva		✓ 🔒
GET	/api/Reserva		✓ 🔒
GET	/api/Reserva/{id}		✓ 🔒
PUT	/api/Reserva/{id}		✓ 🔒
User			^
POST	/api/users/register		✓ 🔒
POST	/api/users/login		✓ 🔒

DiaNoLectivoController

Método	Ruta	Descripción
GET	/api/DiaNoLectivo/{id}	Obtener un día no lectivo por ID.
GET	/api/DiaNoLectivo	Listar todos los días no lectivos.
POST	/api/DiaNoLectivo	Crear un nuevo día no lectivo.
PUT	/api/DiaNoLectivo/{id}	Actualizar un día no lectivo existente.

FranjaHorarioController

Método	Ruta	Descripción
GET	/api/FranjaHorario/activos	Listar franjas horarias marcadas como activas.
GET	/api/FranjaHorario/disponibles	Listar franjas disponibles para reserva.
GET	/api/FranjaHorario/{id}	Obtener una franja horaria por su ID.
GET	/api/FranjaHorario	Listar todas las franjas horarias.
POST	/api/FranjaHorario	Crear una nueva franja horaria.
PUT	/api/FranjaHorario/{id}	Actualizar una franja horaria existente.

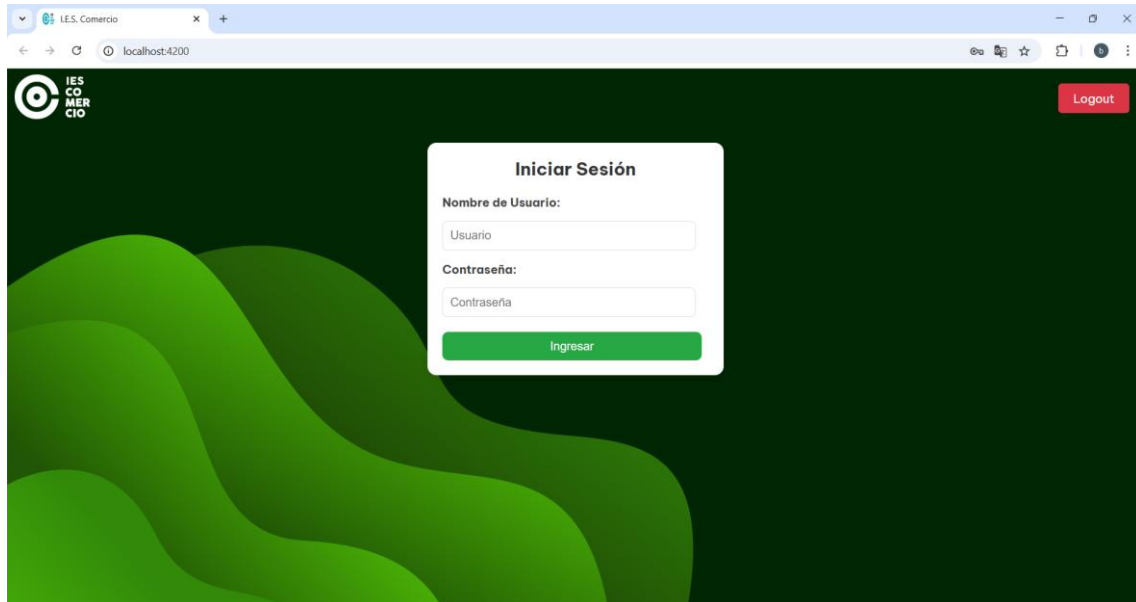
ReservaController

Método	Ruta	Descripción
GET	/api/Reserva/pendientes	Obtener todas las reservas con estado pendiente.
GET	/api/Reserva/mis-reservas	Obtener las reservas del usuario autenticado.
GET	/api/Reserva/{id}	Obtener los detalles de una reserva por su ID.
GET	/api/Reserva	Listar todas las reservas.
POST	/api/Reserva	Crear una nueva reserva.
PUT	/api/Reserva/{id}	Actualizar una reserva existente (por ejemplo, estado).

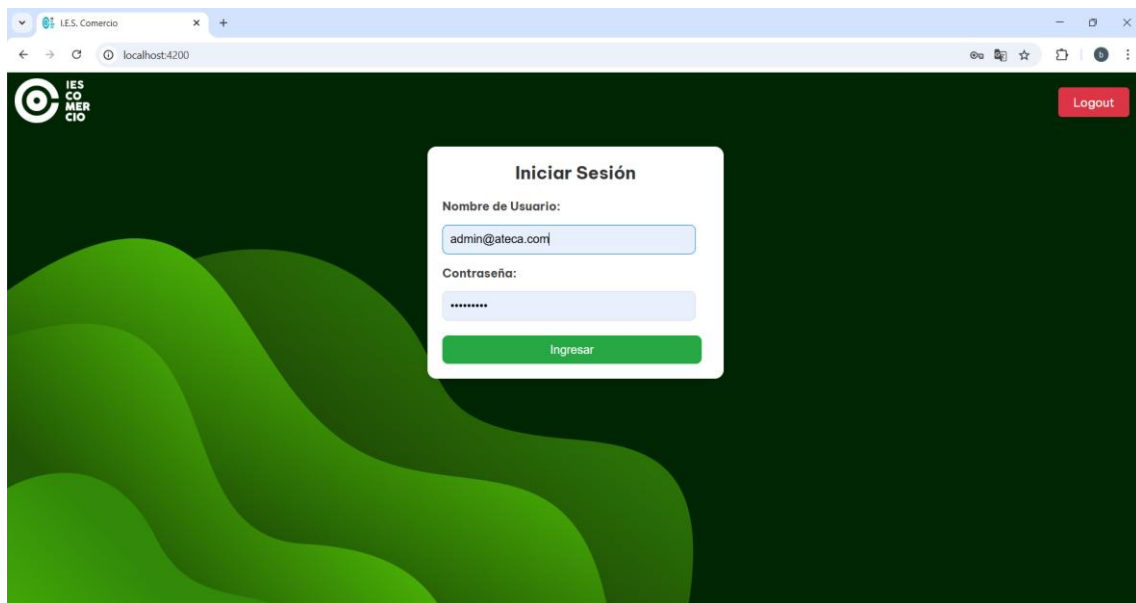
VISTAS

La aplicación de escritorio y la web (Angular) tienen el mismo cometido

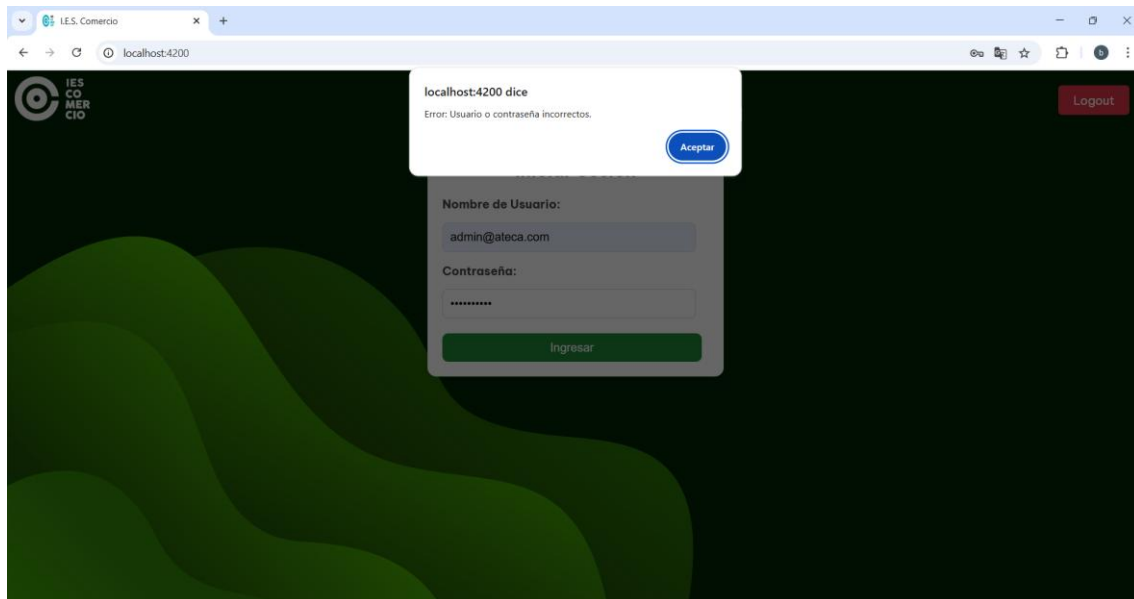
Al iniciar, se muestra un Login, no tenemos acceso a registrar porque ya tenemos un usuario en el seed que está registrado



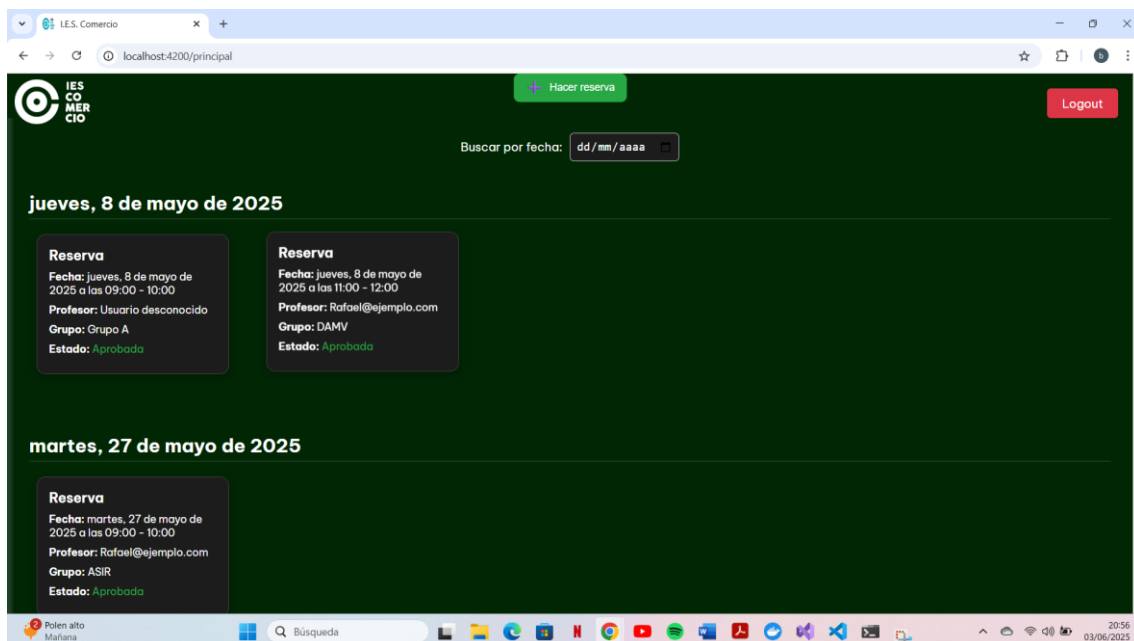
Y ponemos la contraseña y los datos que tenemos



Es sencillo e intuitivo con únicamente nombre de usuario y contraseña y un botón de ingresar, si los datos son válidos, nos lleva a la vista principal. Si son erróneos se muestra un mensaje de error.



Cuando metemos los datos correctamente nos lleva única y principal página

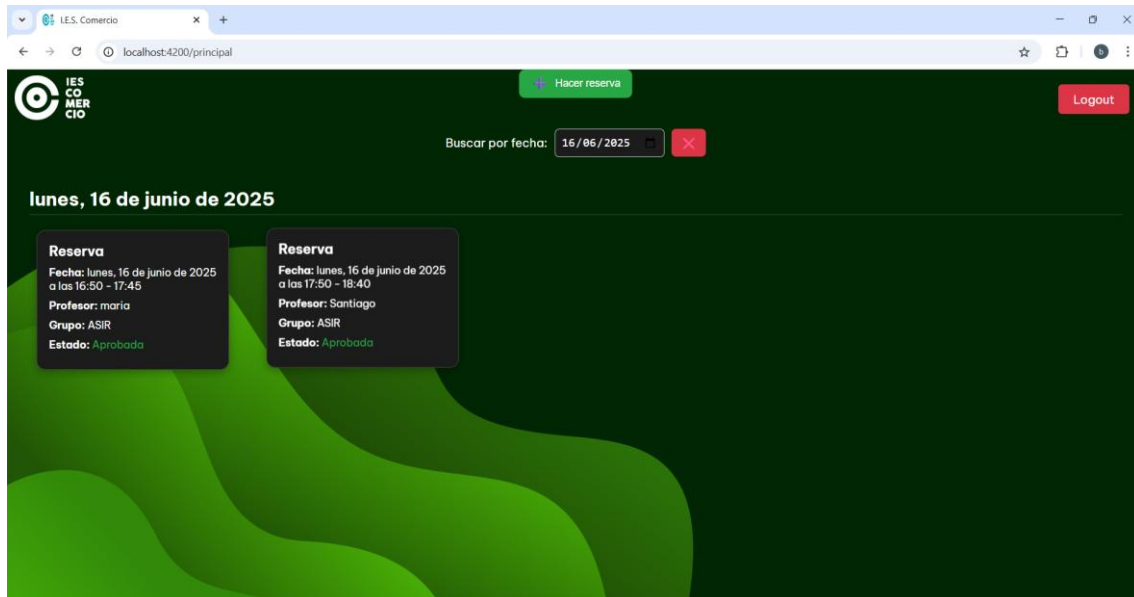


Aquí podemos ver alguna prueba de ejemplo cuando, por ejemplo, no habíamos configurado las franjas horarias, eso ya está hecho y ya no admitiría reservas fuera de las franjas horarias que hay.

Es el encargado de mostrar la lista de reservas agrupadas por fecha, con filtros por fecha y funcionalidad para mostrar el formulario de creación de nuevas reservas.

También hay presente un campo para buscar el día que se quiere reservar y ver como esta el día, un botón para hacer una nueva reserva y el logout

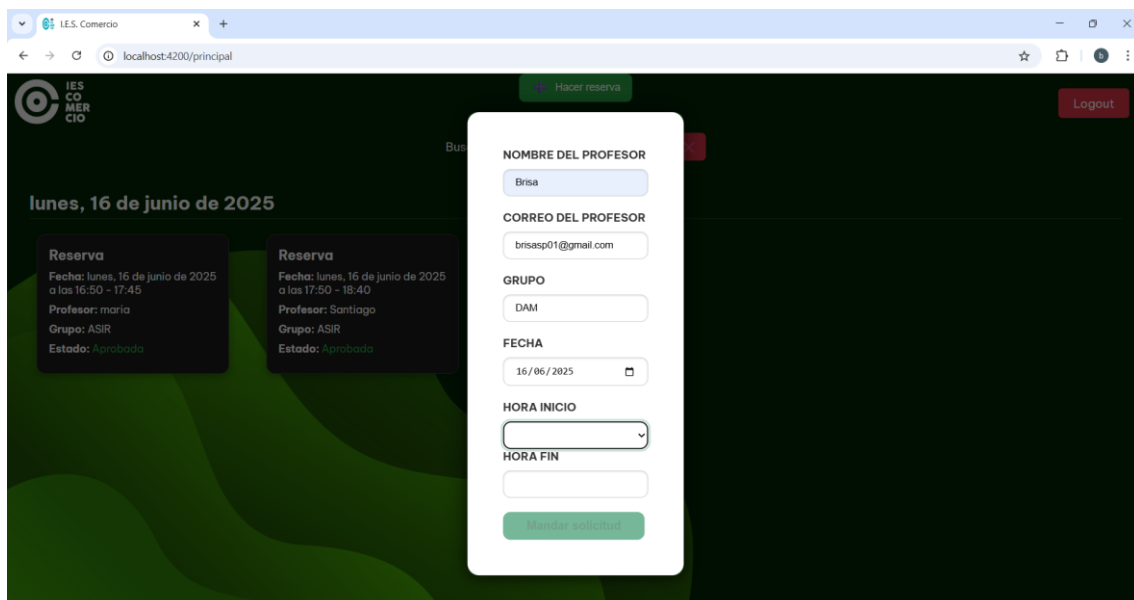
Aquí vemos como el filtro ejecuta la búsqueda de reservas de ese día



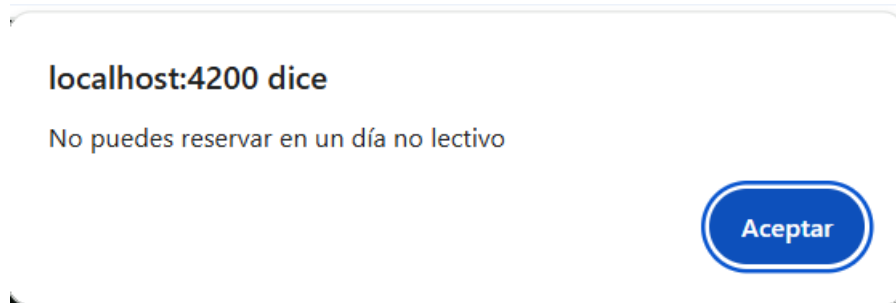
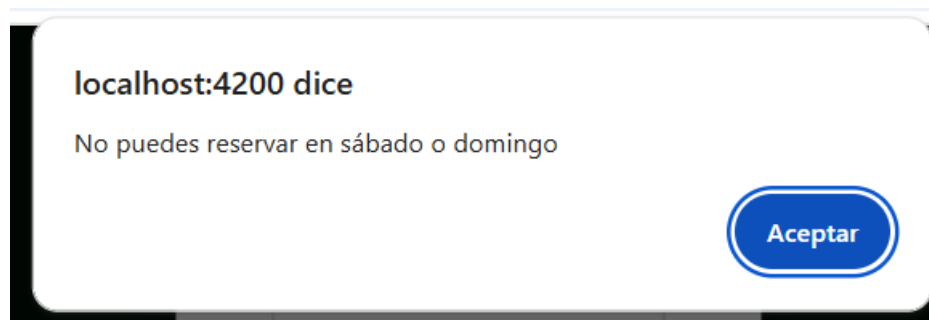
Ahora vamos a crear una nueva reserva

Ese mismo día, por ejemplo, para comprobar la duplicidad de las reservas.

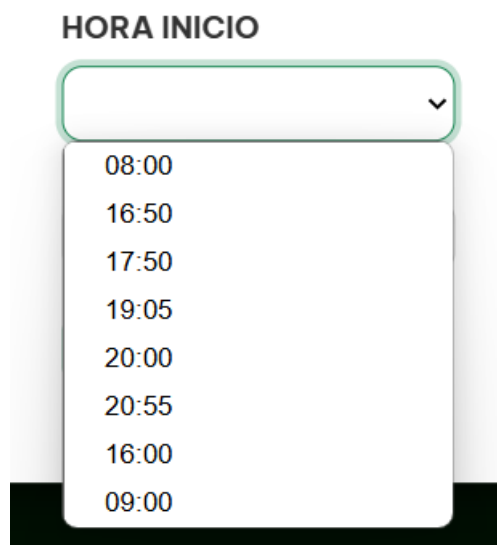
Como podemos observar tras seleccionar el día, tenemos que rellenar todos los campos del formulario sino la solicitud no se manda



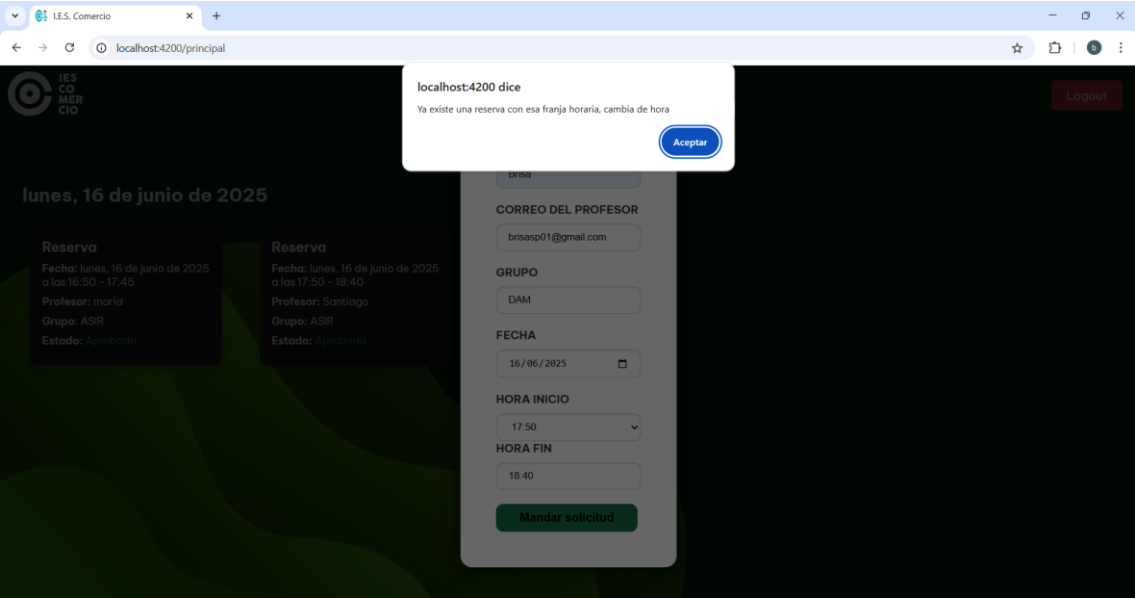
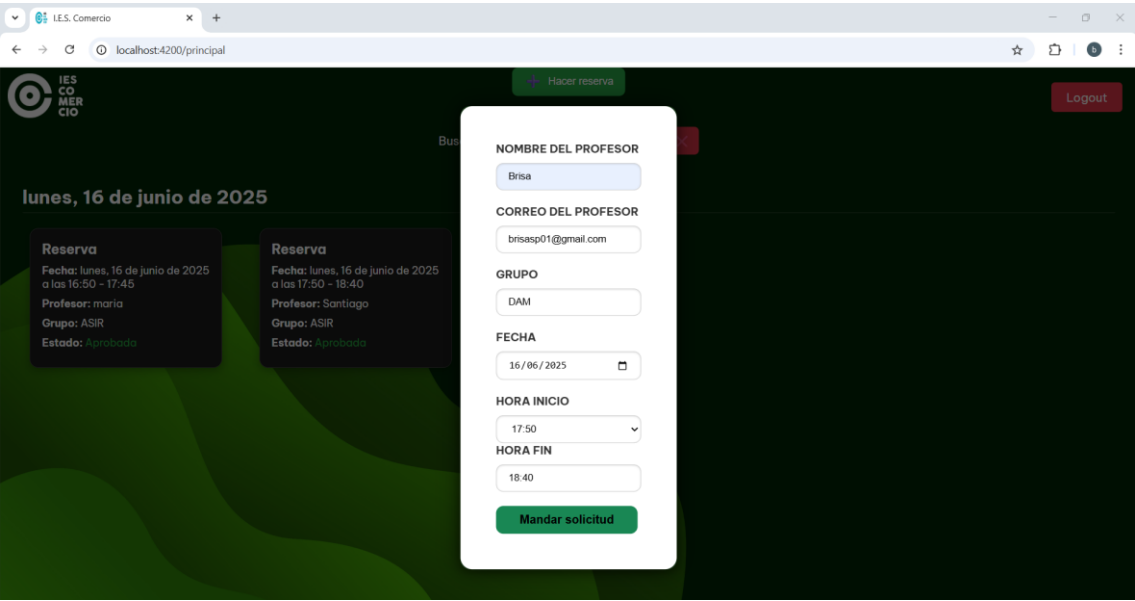
Respecto al campo fecha también tenemos una advertencia tanto si se intenta reservar fin de semana o días no lectivos



Las franjas horarias van directamente conectadas con la API, entonces si por ejemplo se modifica una automáticamente se cambiaria en Angular, y cada hora de inicio va con una hora determinada de fin.



Si seleccionamos una franja horaria ya aprobada, nos salta este mensaje



Para que se cambie la hora, porque no se pueden reservar el mismo día a la misma franja horaria, procedemos a cambiar la franja horaria entonces

The screenshot shows a web browser window with the URL `localhost:4200/principal`. The page has a dark green background with a logo in the top left and a 'Logout' button in the top right. A modal form is centered on the screen. The form contains the following fields:

- NOMBRE DEL PROFESOR:** A text input with the value 'Brisa'.
- CORREO DEL PROFESOR:** A text input with the value 'brisasp01@gmail.com'.
- GRUPO:** A text input with the value 'DAM'.
- FECHA:** A date picker showing '16/06/2025'.
- HORA INICIO:** A dropdown menu showing '19:05'.
- HORA FIN:** A text input with the value '19:55'.
- Mandar solicitud:** A green button at the bottom of the form.

In the background, there are two reservation cards for 'lunes, 16 de junio de 2025'. Each card shows a time slot (16:50-17:45 and 17:50-18:40), the professor's name (Santiago), the group (ASIR), and the status (Aprobado).

This screenshot shows the same reservation form as the previous one, but with a success message overlay. The message is a white box with a green checkmark icon and the text: 'localhost:4200 dice' and 'Tu solicitud se ha mandado correctamente.' Below the message is a blue 'Aceptar' button. The reservation form is still visible in the background, showing the same data as before.

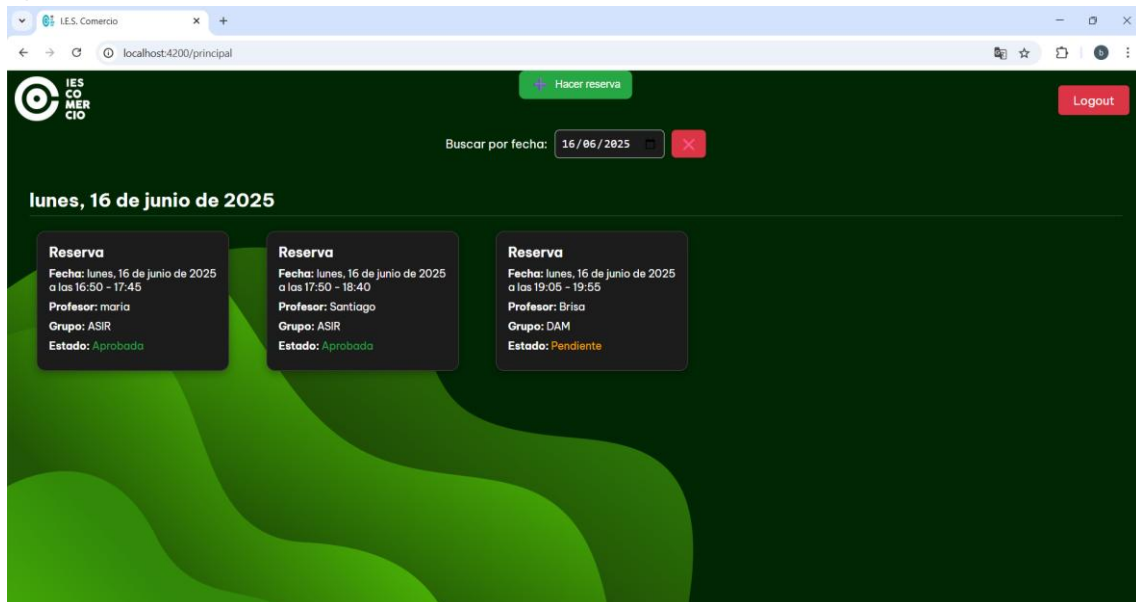
Automáticamente se genera un mensaje de nueva reserva al correo del administrador

The screenshot shows a Gmail inbox interface. The search bar at the top contains 'Buscar correo'. The inbox list shows several emails, including one from 'Promociones' and one from 'Social'. The email 'Nueva solicitud de reserva' is highlighted, showing the subject 'Nueva solicitud de reserva - Se ha realizado una nueva reserva Profesor: Brisa (brisasp01@gmail.com) Fecha: 2025-06-16 Hor...' and the time '21:07'.

Concretamente llega ese mensaje:

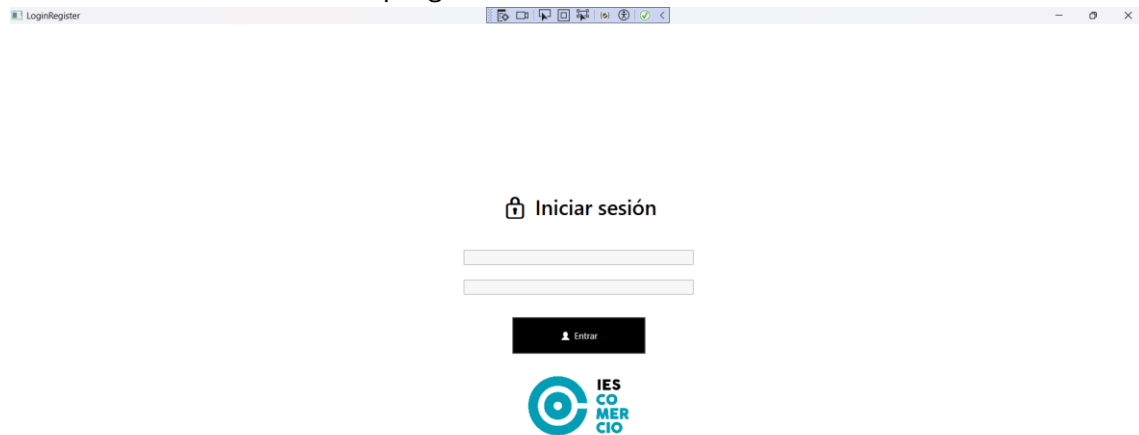


Y la reserva se nos carga ahí como pendiente claro, hasta que el administrador proceda a aprobarla

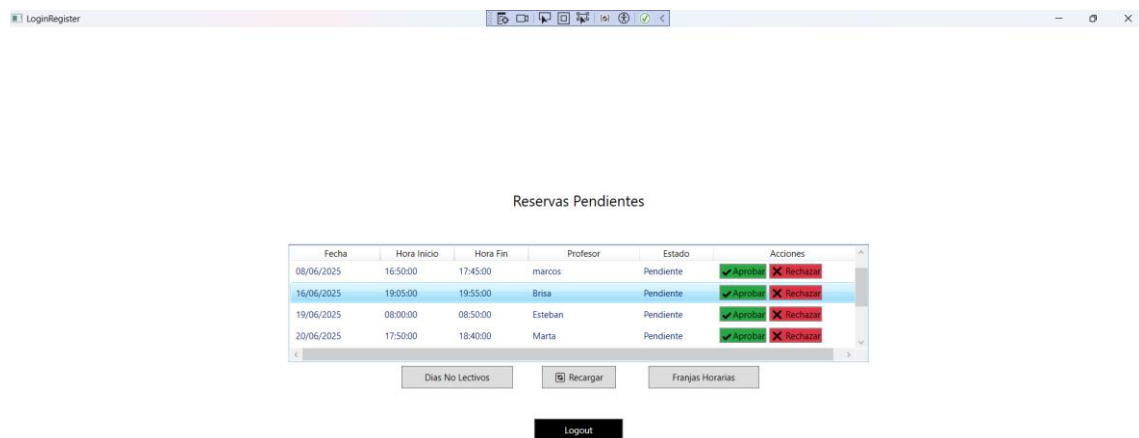


Y si le damos al botón Logout se cierra la sesión.

Ahora el administrador tiene que gestionar las reservas

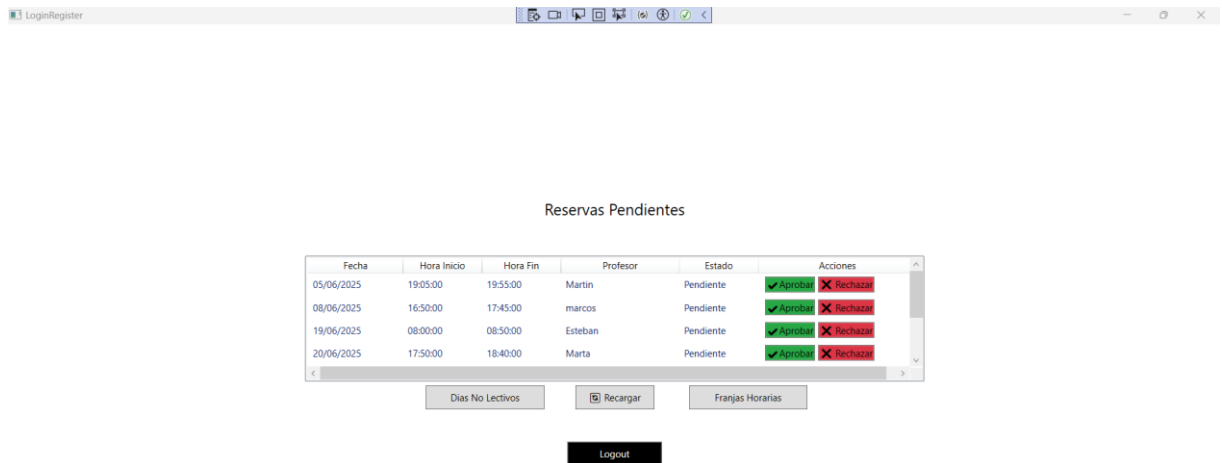


Una vez hecho el login nos sale la pagina principal donde sales las reservas por confirmar o rechazar, ahí tenemos la reserva que acabamos de hacer



Vamos a proceder a rechazarla:

Y ya no nos sale



Pero llega la notificación al correo del profesor que quería reservar ese día en esa franja horaria:



Se ve como que es para mi porque en este caso coincide el correo del admin con el mio personal

Reserva rechazada

Recibidos x



brisasp01@gmail.com

para mí ▼

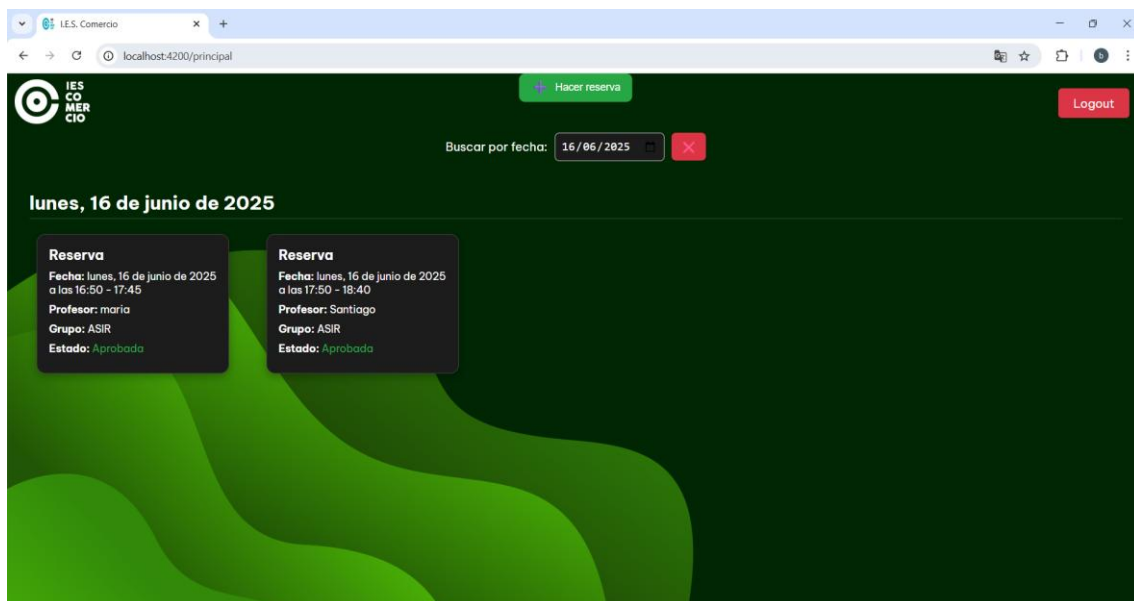
Tu reserva ha sido rechazada

Fecha: 2025-06-16

Hora: 19:05:00 - 19:55:00

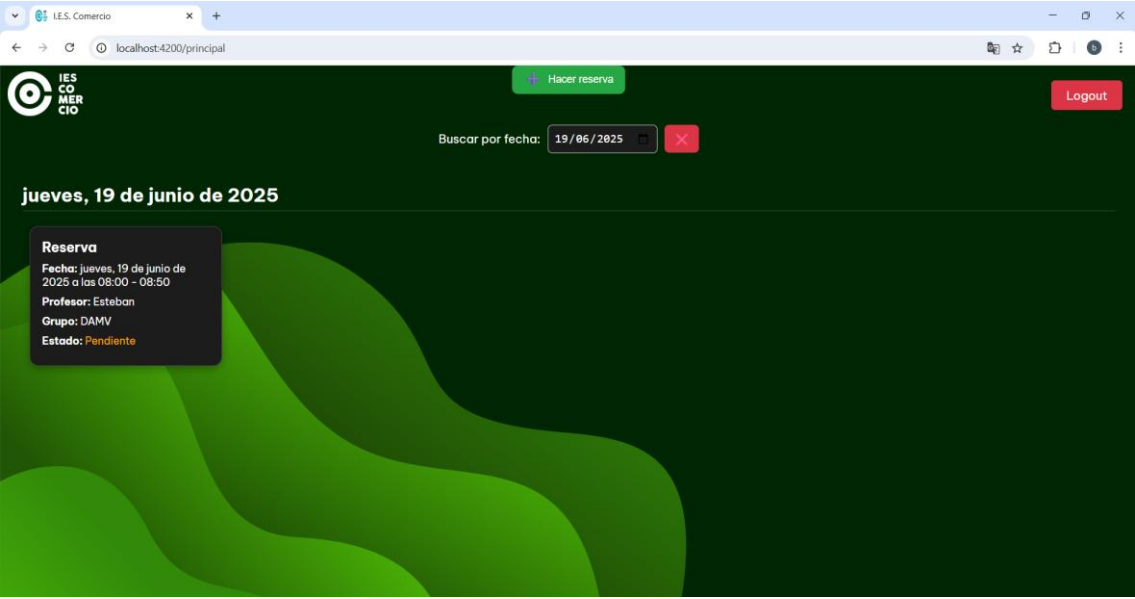
Grupo: DAM

Motivo: Contacta con el administrador para más información.



Y se borrara la petición pendiente de Angular

Ahora vamos a hacer lo mismo, pero aceptando la propuesta:



Cogemos esa mismamente

Reservas Pendientes

Fecha	Hora Inicio	Hora Fin	Profesor	Estado	Acciones	
05/06/2025	19:05:00	19:55:00	Martin	Pendiente	<input checked="" type="checkbox"/> Aprobar	<input type="checkbox"/> Rechazar
08/06/2025	16:50:00	17:45:00	marcos	Pendiente	<input checked="" type="checkbox"/> Aprobar	<input type="checkbox"/> Rechazar
19/06/2025	08:00:00	08:50:00	Esteban	Pendiente	<input checked="" type="checkbox"/> Aprobar	<input type="checkbox"/> Rechazar
20/06/2025	17:50:00	18:40:00	Marta	Pendiente	<input checked="" type="checkbox"/> Aprobar	<input type="checkbox"/> Rechazar

Dias No Lectivos

Recargar

Franjas Horarias

Logout

Reservas Pendientes

Fecha	Hora Inicio	Hora Fin	Profesor	Estado	Acciones	
05/06/2025	19:05:00	19:55:00	Martin	Pendiente	<input checked="" type="checkbox"/> Aprobar	<input type="checkbox"/> Rechazar
08/06/2025	16:50:00	17:45:00	marcos	Pendiente	<input checked="" type="checkbox"/> Aprobar	<input type="checkbox"/> Rechazar
19/06/2025	08:00:00	08:50:00	Esteban	Pendiente	<input checked="" type="checkbox"/> Aprobar	<input type="checkbox"/> Rechazar
20/06/2025	17:50:00	18:40:00	Marta	Pendiente	<input checked="" type="checkbox"/> Aprobar	<input type="checkbox"/> Rechazar

Dias No Lectivos

Recargar

Franjas Horarias

Logout

Reservas Pendientes

Fecha	Hora Inicio	Hora Fin	Profesor	Estado	Acciones
05/06/2025	19:05:00	19:55:00	Martin	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
08/06/2025	16:50:00	17:45:00		Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
19/06/2025	08:00:00	08:50:00		Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
20/06/2025	17:50:00	18:40:00		Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar

Dias No Lectivos

Frangas Horarias

Logout

Y se actualiza automáticamente en Angular

Y ya estaría

En WPF también encontramos estos campos/botones

Reservas Pendientes

Fecha	Hora Inicio	Hora Fin	Profesor	Estado	Acciones
05/06/2025	19:05:00	19:55:00	Martin	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
20/06/2025	17:50:00	18:40:00	Marta	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
20/06/2025	16:50:00	17:45:00	Marta	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
23/06/2025	08:00:00	08:50:00	Brisa	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar

Dias No Lectivos

Recargar

Frangas Horarias

Logout

Tenemos el campo de días no lectivos, que son días que no se pueden seleccionar porque no hay clase, evidentemente, siguiendo el calendario escolar

Agregar Día No Lectivo

ID	Fecha	Motivo
4	23/12/2025	Vacaciones de Navidad
5	24/12/2025	Vacaciones de Navidad
6	25/12/2025	Navidad
7	26/12/2025	Vacaciones de Navidad
8	27/12/2025	Vacaciones de Navidad
9	28/12/2025	Vacaciones de Navidad
10	29/12/2025	Vacaciones de Navidad
11	30/12/2025	Vacaciones de Navidad

< Anterior
Siguiente >
Logout

Dia No Lectivo

Fecha

03/06/2025

15

junio de 2025

L	M	X	J	V	S	D
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Home

Dia No Lectivo


Fecha

03/06/2025

15

Motivo

Añadir Dia No Lectivo



Ahí añadimos un día No lectivo y el Motivo que se quiera por ejemplo

Agregar Día No Lectivo

ID	Fecha	Motivo
4	23/12/2025	Vacaciones de Navidad
5	24/12/2025	Vacaciones de Navidad
6	25/12/2025	Navidad
7	26/12/2025	Vacaciones de Navidad
8	27/12/2025	Vacaciones de Navidad
9	28/12/2025	Vacaciones de Navidad
10	29/12/2025	Vacaciones de Navidad
11	30/12/2025	Vacaciones de Navidad



< Anterior

Siguiente >

Logout

Día no lectivo añadido con éxito.

Aceptar

Agregar Día No Lectivo

ID	Fecha	Motivo
28	03/04/2026	Viernes Santo
29	06/04/2026	Lunes de Pascua
30	09/06/2026	Día de La Rioja
31	22/06/2026	Fin de clases
32	26/06/2026	Fin del curso
1011	15/06/2025	Día de pruebas de Brisa
1012	14/06/2025	Día de pruebas de Brisa 2
1013	04/06/2025	Ejemplo



< Anterior

Siguiente >

Logout

Nos vamos al Home después dándole al botón de la casita ubicado en el fondo izquierdo

Reservas Pendientes

Fecha	Hora Inicio	Hora Fin	Profesor	Estado	Acciones
05/06/2025	19:05:00	19:55:00	Martin	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
20/06/2025	17:50:00	18:40:00	Marta	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
20/06/2025	16:50:00	17:45:00	Marta	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar
23/06/2025	08:00:00	08:50:00	Brisa	Pendiente	<input checked="" type="checkbox"/> Aprobar <input checked="" type="checkbox"/> Rechazar

Días No Lectivos

☒ Recargar

Franjas Horarias

Logout

Y vemos un botón de recargar para ver las nuevas reservas pendientes al lado, después también tenemos un botón de franjas horarias

+ Agregar Franja Horaria

Guardar cambios

ID	Horainicio	HoraFin	Activa
6	08:00:00	08:50:00	<input checked="" type="checkbox"/>
13	09:00:00	09:50:00	<input checked="" type="checkbox"/>
12	16:00:00	16:45:00	<input checked="" type="checkbox"/>
7	16:50:00	17:45:00	<input checked="" type="checkbox"/>
8	17:50:00	18:40:00	<input checked="" type="checkbox"/>
9	19:05:00	19:55:00	<input checked="" type="checkbox"/>
10	20:00:00	20:50:00	<input checked="" type="checkbox"/>
11	20:55:00	21:45:00	<input checked="" type="checkbox"/>

< Anterior

Siguiente >

Logout

Donde podemos añadir nuevas Franjas horarias

Dia No

Franja Horaria

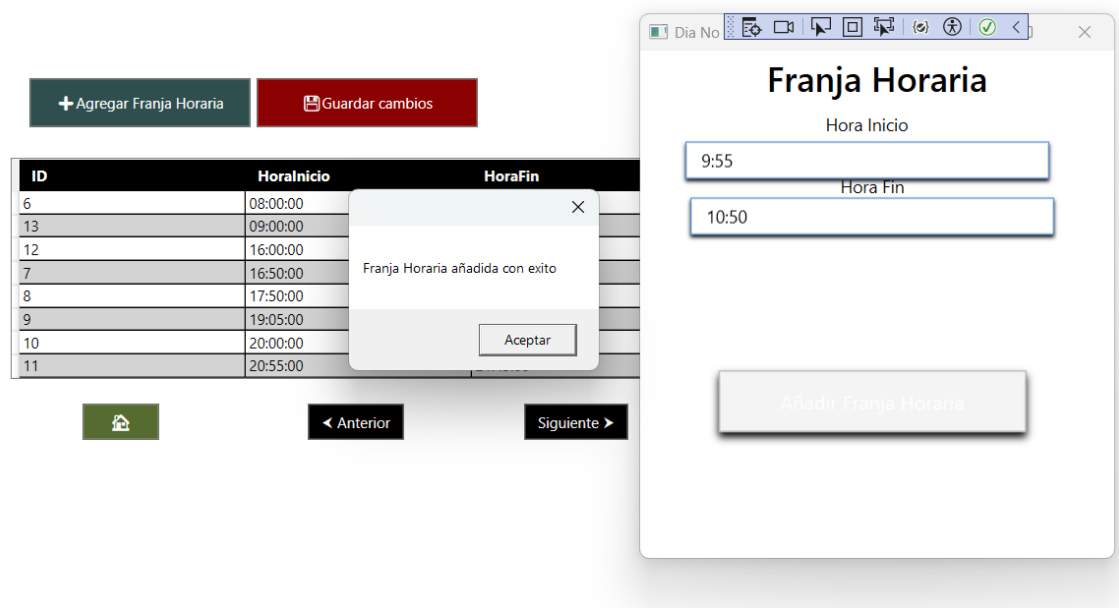
Hora Inicio

9:55

Hora Fin

10:50

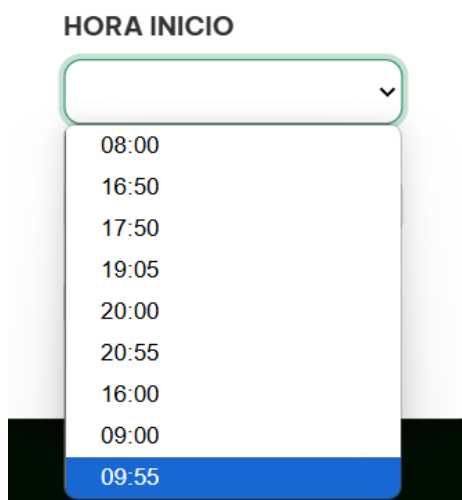
Añadir Franja Horaria



+ Agregar Franja Horaria		Guardar cambios	
ID	Horalnicio	HoraFin	Activa
6	08:00:00	08:50:00	<input checked="" type="checkbox"/>
13	09:00:00	09:50:00	<input checked="" type="checkbox"/>
14	09:55:00	10:50:00	<input checked="" type="checkbox"/>
12	16:00:00	16:45:00	<input checked="" type="checkbox"/>
7	16:50:00	17:45:00	<input checked="" type="checkbox"/>
8	17:50:00	18:40:00	<input checked="" type="checkbox"/>
9	19:05:00	19:55:00	<input checked="" type="checkbox"/>
10	20:00:00	20:50:00	<input checked="" type="checkbox"/>


Y ahí nos sale la nueva franja horaria, están ordenadas en WPF

Ahí tenemos como se cargan en Angular



Y ahora vamos a proceder a modificarla


LoginRegister



+ Agregar Franja Horaria

Guardar cambios

ID	Horainicio	HoraFin	Activa
6	08:00:00	08:50:00	<input checked="" type="checkbox"/>
13	09:00:00	09:50:00	<input checked="" type="checkbox"/>
14	09:58:00	10:58:00	<input checked="" type="checkbox"/>
12	16:00:00	16:45:00	<input checked="" type="checkbox"/>
7	16:50:00	17:45:00	<input checked="" type="checkbox"/>
8	17:50:00	18:40:00	<input checked="" type="checkbox"/>
9	19:05:00	19:55:00	<input checked="" type="checkbox"/>
10	20:00:00	20:50:00	<input checked="" type="checkbox"/>




< Anterior

Siguiente >

Logout


LoginRegister



+ Agregar Franja Horaria

Guardar cambios

ID	Horainicio	HoraFin	Activa
6	08:00:00	08:50:00	<input checked="" type="checkbox"/>
13	09:00:00	09:50:00	<input checked="" type="checkbox"/>
14	09:58:00	10:58:00	<input checked="" type="checkbox"/>
12	16:00:00	16:45:00	<input checked="" type="checkbox"/>
7	16:50:00	17:45:00	<input checked="" type="checkbox"/>
8	17:50:00	18:40:00	<input checked="" type="checkbox"/>
9	19:05:00	19:55:00	<input checked="" type="checkbox"/>
10	20:00:00	20:50:00	<input checked="" type="checkbox"/>



< Anterior

Siguiente >

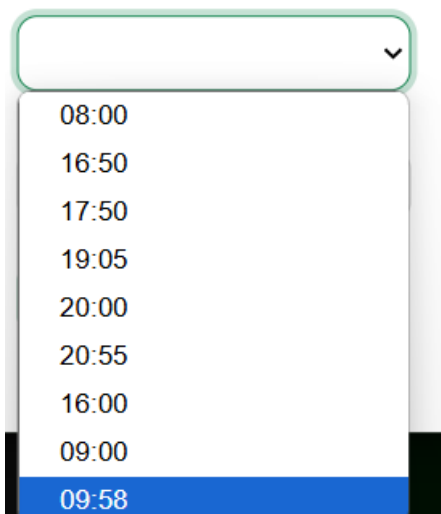
Logout

Cambios guardados correctamente.

Aceptar

Y ahí tenemos la hora en Angular modificada

HORA INICIO

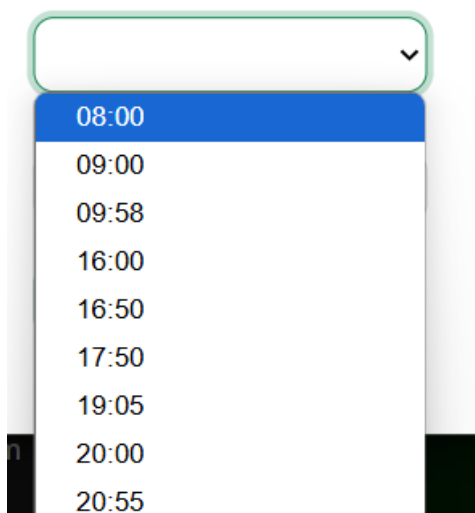


A dropdown menu titled "HORA INICIO" is shown. The menu is open, displaying a list of times. The time "09:58" is selected and highlighted in blue. The other times in the list are: 08:00, 16:50, 17:50, 19:05, 20:00, 20:55, 16:00, and 09:00.

HORA INICIO
08:00
16:50
17:50
19:05
20:00
20:55
16:00
09:00
09:58

Tras unos ajustes de última hora ya salen ordenadas

HORA INICIO



A dropdown menu titled "HORA INICIO" is shown. The menu is open, displaying a list of times. The time "08:00" is selected and highlighted in blue. The other times in the list are: 09:00, 09:58, 16:00, 16:50, 17:50, 19:05, 20:00, and 20:55.

HORA INICIO
08:00
09:00
09:58
16:00
16:50
17:50
19:05
20:00
20:55

Y por último el botón de LogOut, que al igual que en angular nos cierra la sesión

MEJORAS

Visualmente mejorar el calendario angular(Únicamente el front, dado que el back hace su funcionalidad)

Mejorar la interfaz de WPF visualmente(únicamente el front, dado que el back hace su funcionalidad)

Generar el Login con Google Autentication

Poder ver reservas del usuario autenticado únicamente también

Que se actualice automáticamente las reservas en vez de usar botón de recargar