



Universidad Tecnológica Nacional
FRRO

Tecnologías de desarrollo de software IDE

Informe Trabajo Práctico Academia

AÑO: 2021 - Comisión 3K7

Alumnos

Nombre	Legajo
Fonzo, Franco	46882
Tano, German	46873
Del Coro, Lara	46978
Poratti, Candela	47072
Colarte, Federico	47536

Docentes

Porta, Ezequiel
Joaquín, Andrés

Índice

Índice	2
Introducción	3
Tecnologías utilizadas	4
Arquitectura	6
Análisis y Diseño	8
Fragmentos de código	14
Integración con Azure DevOps	15

Introducción

El sistema Academia permite registrar y administrar actividades académicas tales como inscripciones, registro de notas, generación de informe, creación de cursos, ABMC de diferentes entidades, entre otras cosas.

El sistema puede ser utilizado por cualquier persona que posea un usuario con el rol de alumno, docente o administrador (dirección académica).

El mismo se ejecuta en web a través de Azure App Services, o en escritorio. Los datos pueden ser obtenidos tanto por medio de una conexión local, usando el motor de base de datos SQL Server, como por conexión remota a Azure SQL Server.

El diseño está unificado en ambas plataformas y una vez que un usuario ingresa, sólo puede realizar acciones permitidas según su rol.

Los Usuarios poseen un usuario y contraseña, una Persona asociada, y pueden ser habilitados o inhabilitados.

Las Personas pueden tomar el rol de Administrador, Alumno, o Docente, y tienen datos en común tales como Legajo, Nombre, Apellido, Dirección entre otros. Las Materias pertenecen a un Plan y tienen cantidad de horas de cursado semanales y totales. Los Planes son de una Especialidad.

La relación entre una Personas del tipo Docente y un Cursos, se llama DocenteCurso, y se la conoce como Dictado. La relación entre una Persona del tipo Alumno y un Curso, se llama AlumnoInscripción y se conoce como Inscripción.

Tecnologías utilizadas

- IDE: **Visual Studio - Versión 2019**
- Framework: **.NET Framework 4.7.2**
- Motor de Base de Datos: **Microsoft SQL Server / Azure SQL Server + SQL Database**

Al comenzar el desarrollo utilizamos Microsoft SQL Server como motor de base de datos de la aplicación de escritorio. Una vez desarrollada la capa de presentación web, e integrado el proyecto con Azure DevOps y Azure App Service, optamos por utilizar Azure SQL Server con SQL Database para que los datos puedan ser consultados tanto por instancia local, como remota.

- Versionado de Código: **Github**

Github nos permitió llevar el versionado del proyecto de una manera organizada. Además posibilitó y facilitó el trabajo en equipo, de manera que pudimos desarrollar al mismo tiempo e implementar nuevas partes de la aplicación gracias a la creación de distintas ramas y el posterior merge a la rama principal sin correr riesgos de arruinar el trabajo ya realizado.

- Acceso a datos: **Entity Framework**

El acceso a la base de datos a través de Entity Framework permite un código mucho más corto y limpio. Esto se ve reflejado en el proyecto Data.Database, el cual contiene en cada archivo adapter.cs breves líneas de código que permiten traer a la aplicación los datos de cada entidad.

- Integración y Distribución web: **Azure Devops (CI/CD)**

La integración y distribución continua que brinda Azure DevOps para la aplicación web fue implementada como integración del primer trabajo práctico realizado por el grupo. Esta tecnología agiliza la integración y distribución de la aplicación, ya que al realizar un commit en la rama de producción del repositorio de GitHub, Azure DevOps se encarga de manera automática de compilar la aplicación a través de pipelines, y enviar la nueva versión del proyecto a su App Service para que cualquier persona pueda acceder a través de su URL.

- Acceso a aplicación web: **Azure App service**

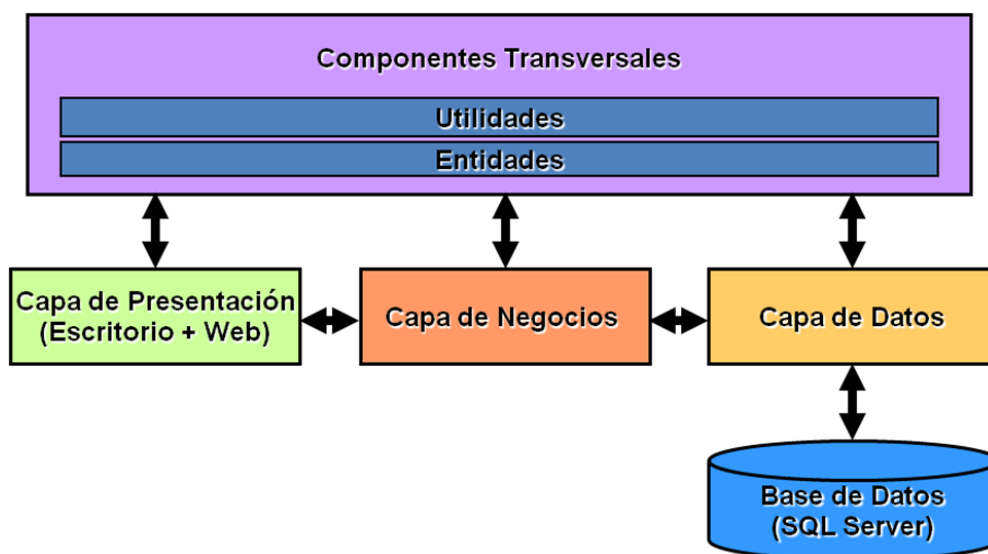
Azure App service, también implementada como integración del primer trabajo práctico, nos brindó la posibilidad de hospedar nuestra web ya creada para que la misma sea accesible desde cualquier navegador con acceso a internet.

Arquitectura

La solución implementa una arquitectura basada en capas.

Capas y proyectos de la solución:

Proyecto	Capa
Data.Database	Capa de Datos
Business.Logic	Capa de Negocios
UI.Web	Capa de Presentación: Web
UI.Desktop	Capa de Presentación: Escritorio
Business.Entities	Entidades
Util	Utilidades



Capa de Datos: El proyecto Data.Database centraliza la conexión a la base de datos utilizando el motor de base de datos Microsoft SQL Server, o Azure SQL Server. Para la consulta de los datos se utiliza Entity Framework.

Capa de Negocios: El proyecto Business.Logic es la capa intermediaria entre la capa de Presentación y la capa de Datos.

Capa de presentación

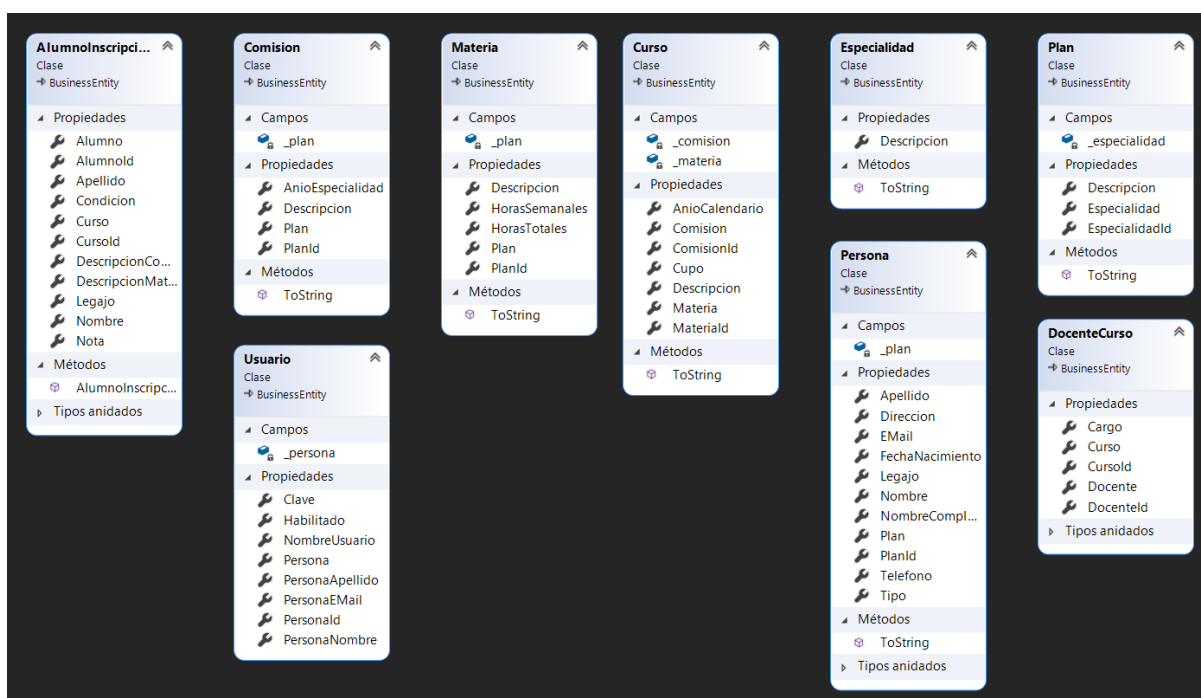
Web: El proyecto UI.Web utiliza WebForms con Master Page. El mismo contiene un Login que al iniciar sesión redirige al usuario a la página principal, con un menú de navegación.

Escritorio: El proyecto UI.Escritorio utiliza WinForms que inicia con un Login, que al iniciar sesión, redirige al usuario a la página principal con un menú.

Componentes Transversales

Entidades: El proyecto Business.Entities contiene la definición de cada entidad. Las mismas se implementan como Objetos.

Diagrama de navegación de clases



Análisis y Diseño

LOGIN

Escritorio:



A desktop application window titled "LOGIN" with a dark theme. The window has a teal sidebar on the left containing a graduation cap icon and the word "ACADEMIA". The main area has two input fields labeled "Usuario" and "Contraseña", a blue "Iniciar sesión" button, and a link "¿Has olvidado la contraseña?".

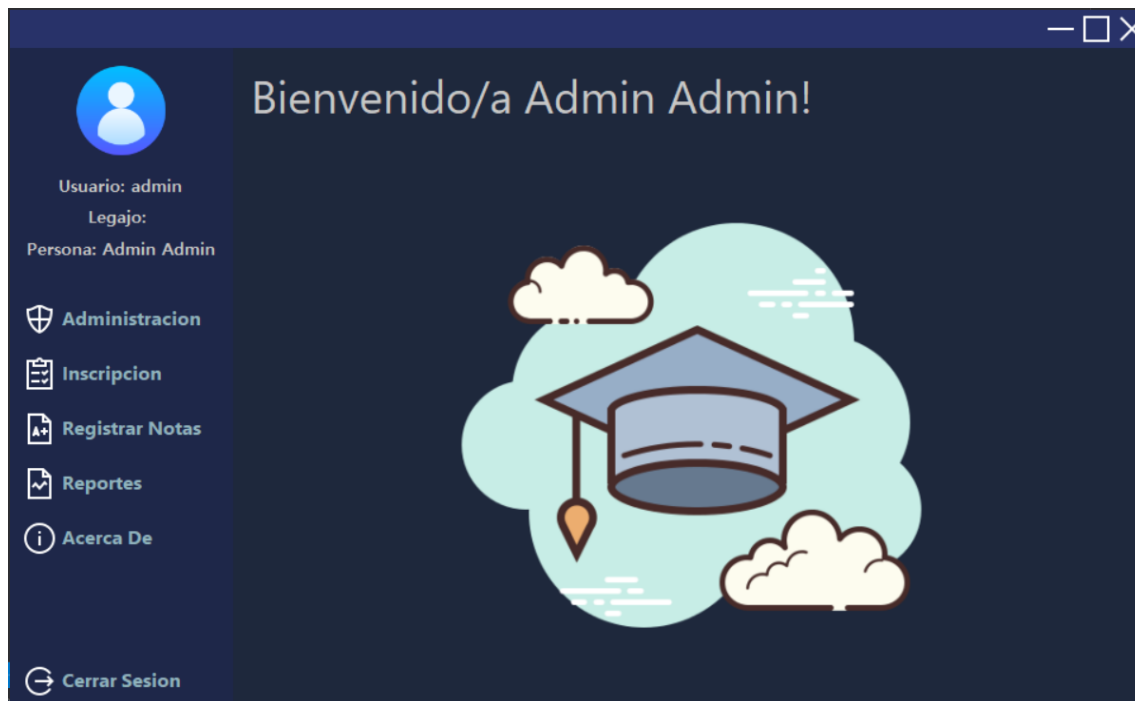
Web:



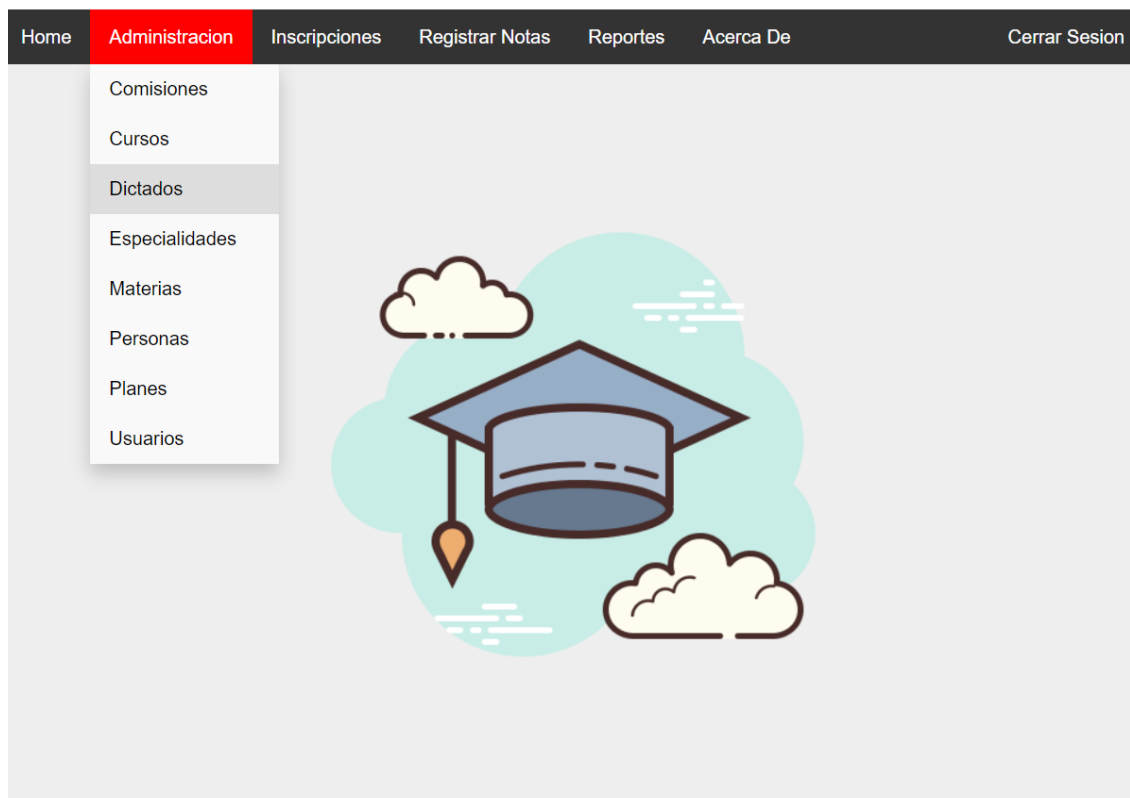
A web browser view of the login form. The header "Academia" is in a dark grey bar. The login form is a white box with a light grey border, containing the same "Usuario" and "Contraseña" fields, a teal "Iniciar Sesión" button, and the link "¿Has olvidado la contraseña?".

Menú de acceso

Escritorio: (vista de administrador)

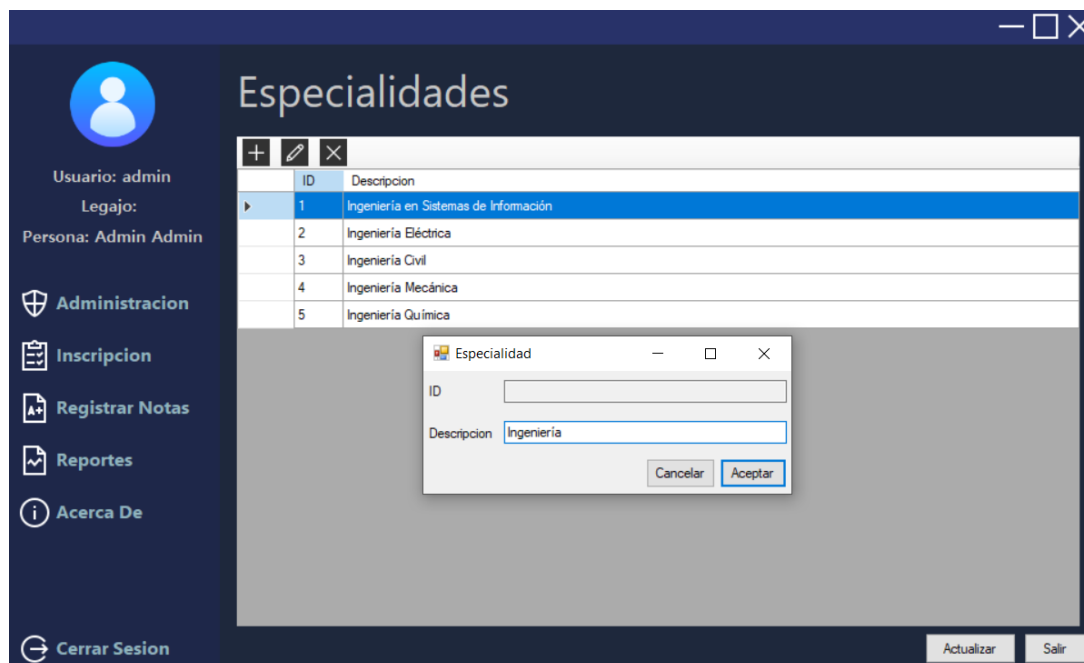


Web: (vista de administrador)

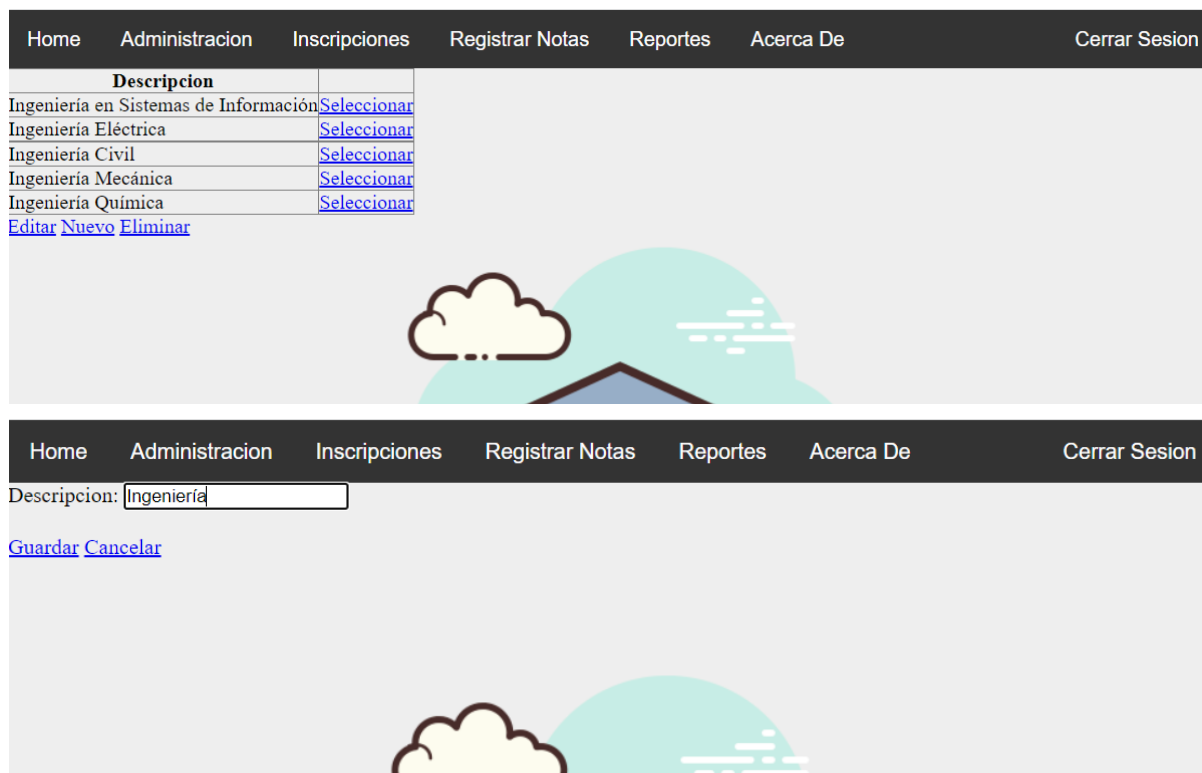


ABMC de entidades

Escritorio:



Web:



Generación de reportes

Escritorio:

Cursos



ID Curso: 2 Año Calendario: 2021 Cupo: 40

ID Materia: 2 Descripción: Software del Desarrollo IDE

Comisión: 3

ID	Alumno	Legajo
4	Franco Fonzo	46882
8	Federico Colarte	47536
7	Lara Del Coro	46978
6	German Tano	46873
9	Juana Garcia	47502
12	Sofia Rodriguez	46980

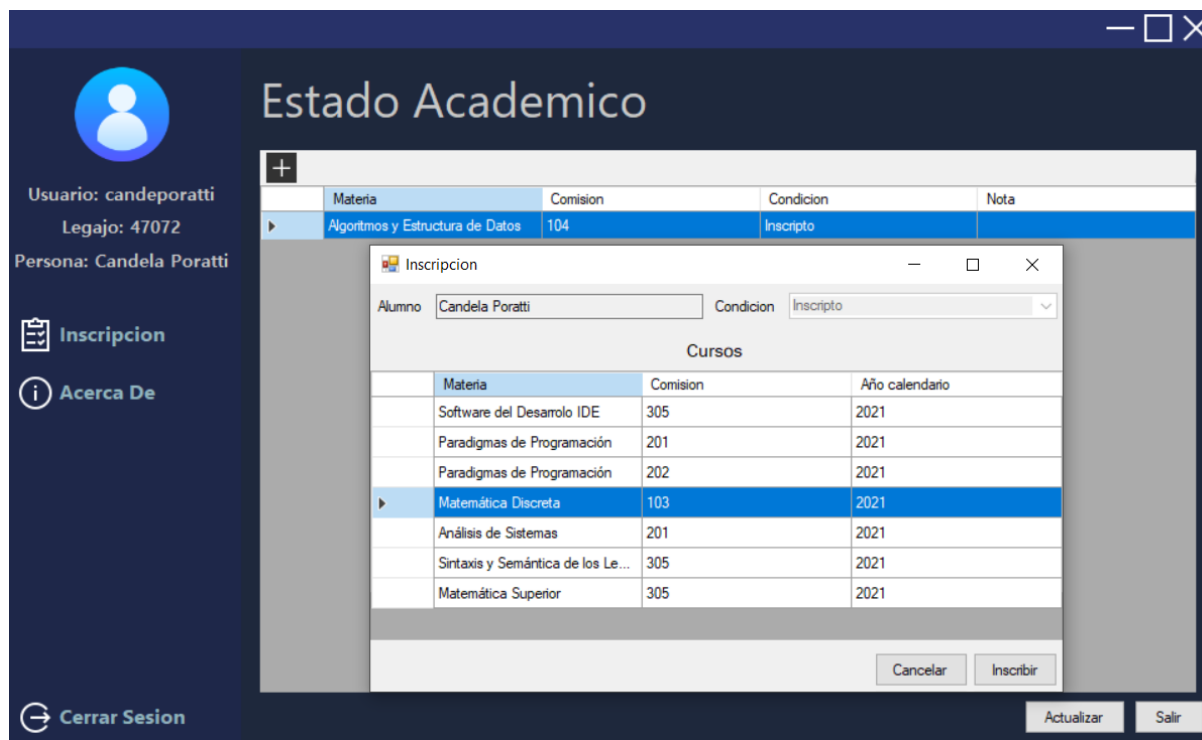
Planes



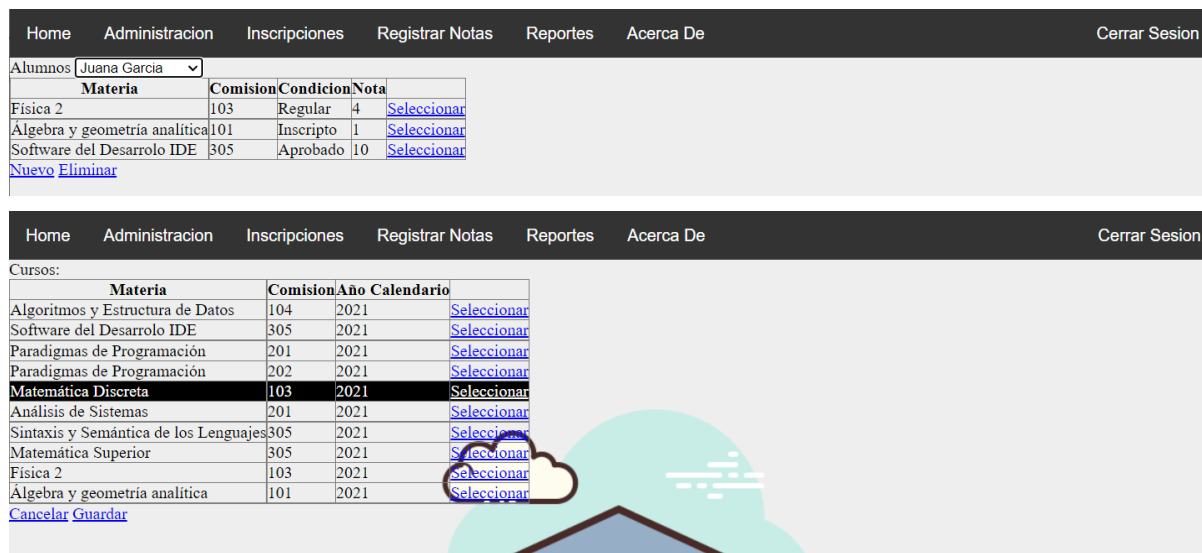
ID Plan	Descripción	ID Especialidad	Desc Esp
2	Plan de Estudio 2008	1	Ingeniería en Sistemas de Información

Inscripciones

Escritorio:

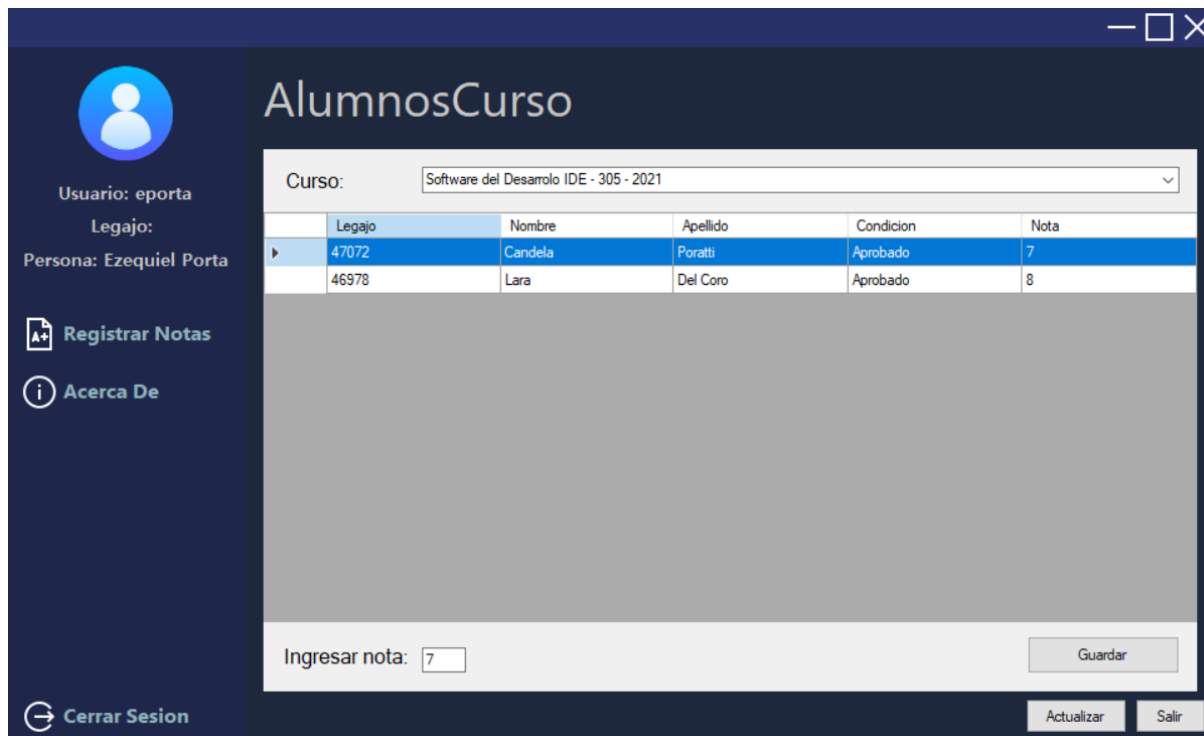


Web:



Registro de notas

Escritorio:



The desktop application window titled "AlumnosCurso" shows a sidebar with user information and navigation links. The main area displays a table of students for the selected course "Software del Desarrollo IDE - 305 - 2021".

Usuario: eporta
Legajo:
Persona: Ezequiel Porta

[Registrar Notas](#)
[Acerca De](#)

Curso: Software del Desarrollo IDE - 305 - 2021

Legajo	Nombre	Apellido	Condicion	Nota
47072	Candela	Poratti	Aprobado	7
46978	Lara	Del Coro	Aprobado	8

Ingresar nota: [Guardar](#)

[Cerrar Sesion](#) [Actualizar](#) [Salir](#)

Web:



The web application interface has a top navigation bar with links: Home, Registrar Notas, Acerca De, and Cerrar Sesion. The main content area shows the same course selection and student table as the desktop version.

Curso: Software del Desarrollo IDE - 305 - 2021

Legajo	Nombre	Apellido	Condicion	Nota	
47072	Candela	Poratti	Inscripto		Seleccionar
46978	Lara	Del Coro	Inscripto		Seleccionar

Ingresar nota: [Cancelar](#) [Guardar](#)



Fragmentos de código

Entity Framework:

Ejemplo de método GetAll() y GetOne() para Usuarios, aplicando Entity Framework:

```

1.     public List<Usuario> GetAll()
2.     {
3.         try
4.         {
5.             using (AcademiaContext context = new AcademiaContext())
6.             {
7.                 return context.Usuario.Include(u =>
8.                     u.Persona).ToList();
9.             }
10.        }
11.        catch (Exception ex)
12.        {
13.            throw new Exception("Error al recuperar los
14.                usuarios.", ex);
15.        }
16.
17.    public Usuario GetOne(int id)
18.    {
19.        try
20.        {
21.            using (var context = new AcademiaContext())
22.            {
23.                return context.Usuario.Include(u =>
24.                    u.Persona).FirstOrDefault(u => u.ID == id);
25.            }
26.        }
27.        catch (Exception ex)
28.        {
29.            throw new Exception("Error al recuperar el usuario:
30.                /nMensaje.", ex);
31.        }
32.    }

```

Integración con Azure DevOps

Creamos un proyecto de Azure DevOps en el que usamos el repositorio de Github en el cual fuimos desarrollando la aplicación a lo largo del año.

Para esto, creamos un pipeline de compilación de builds. Con un trigger, el pipeline se ejecuta cada vez que se realiza un commit en la rama Main del repositorio Github asociado.

Al terminar de ejecutarse el primer pipeline se crea un artefacto, que gracias al uso de la plantilla de asp.net, toma el proyecto web y lo envía a una pipeline de release. Una vez que se ejecuta esa segunda pipeline, Azure envía el artefacto, es decir, todo el proyecto compilado, a un App service para poder acceder a la aplicación.

El App service y la aplicación están conectadas a la base de datos cargada en Azure, por lo que los datos están unificados e integrados.

De esta manera obtenemos una integración y distribución continua del proyecto a través de los servicios de Azure.

[Link al proyecto Azure DevOps aquí](#)

[Link a la aplicación web aquí](#)