

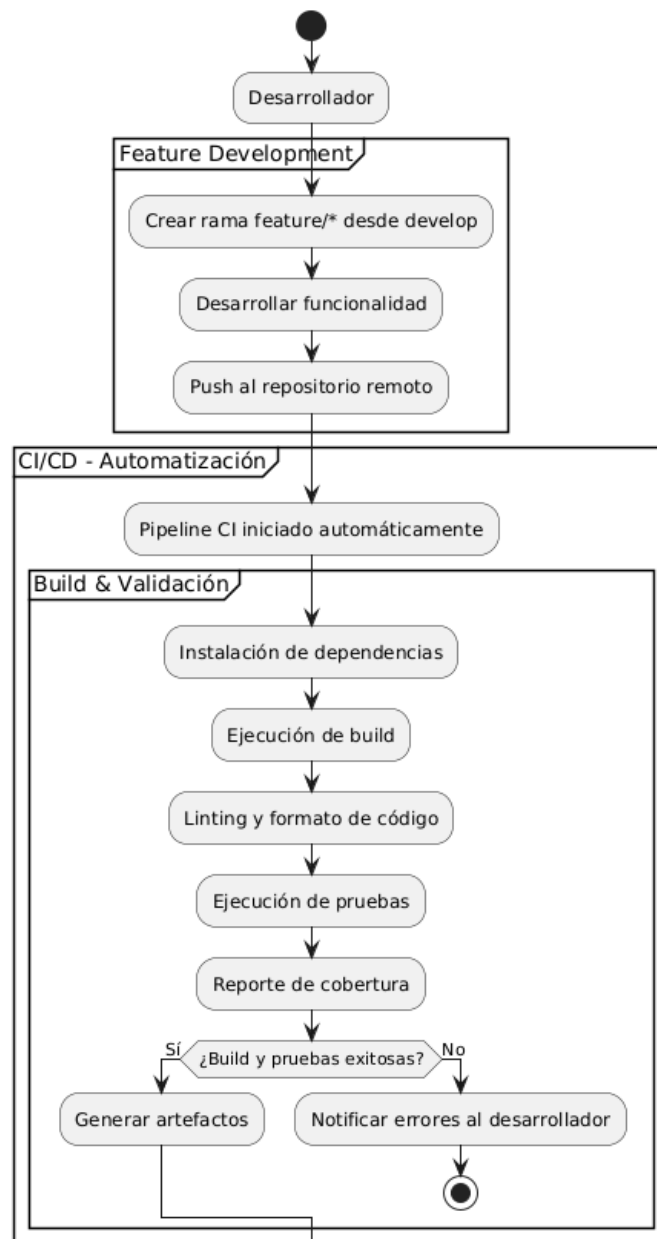
### Proyecto Primer Bimestre

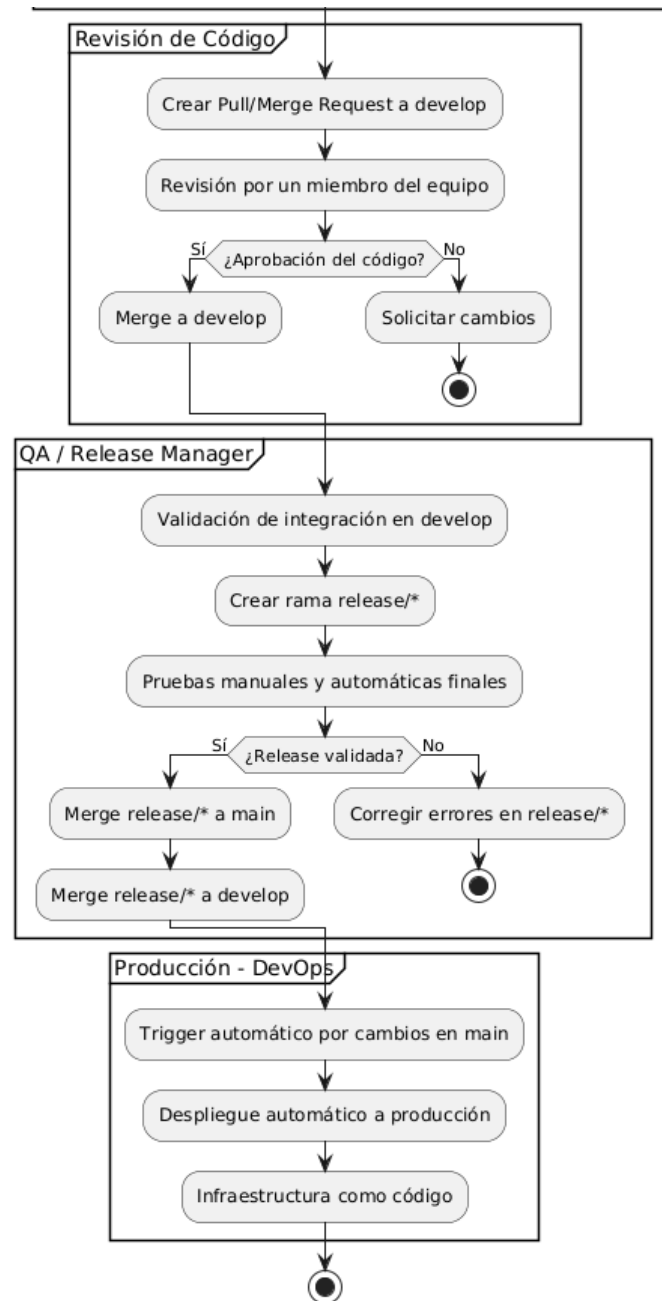
**Integrante:** Jonatan Claudio

**Proyecto:** Sistema Simple de Registro de Notas

**Actividades del Proyecto:**

- Estrategia de ramification (branching strategy)





El flujo de trabajo descrito en el diagrama sigue una estrategia de desarrollo colaborativa basada en ramas, comúnmente conocida como Git Flow. Todo comienza cuando un desarrollador crea una nueva rama llamada `feature/*` a partir de la rama principal de desarrollo (`develop`). En esa rama trabaja en una nueva funcionalidad del sistema, como por ejemplo el módulo de registro o consulta de notas. Una vez que ha terminado el desarrollo, realiza un push al repositorio remoto, lo que activa automáticamente un pipeline de integración continua (CI).

Este pipeline automatizado instala las dependencias, ejecuta el proceso de construcción (build), aplica herramientas de formato y análisis de código (como linters) y corre pruebas automatizadas para validar que la funcionalidad no rompa nada. También genera un reporte de cobertura de pruebas. Si alguna etapa falla, se notifica al desarrollador para que realice correcciones. Si todo va bien, se generan los artefactos del sistema.

Luego, el desarrollador crea un Pull Request o Merge Request hacia la rama develop, donde otro miembro del equipo revisa el código. Si el código es aprobado, se integra en la rama develop. En caso contrario, se solicitan cambios y el flujo regresa al desarrollador para corregirlos.

Una vez que hay una versión estable lista para publicación, el equipo de QA o el release manager crea una rama release/\* desde develop. En esa rama se realizan pruebas manuales y automáticas finales. Si la versión es validada, se hace merge tanto a la rama main (que representa la producción) como de nuevo a develop para asegurar que todo quede sincronizado. Si la versión no es válida, se corrigen errores en la rama release.

Finalmente, cuando hay cambios en la rama main, se puede activar automáticamente el despliegue a producción si se tiene configurada la infraestructura de DevOps. Esto permite que el sistema esté disponible con la última versión sin intervención manual, utilizando tecnologías como AWS, contenedores o infraestructura como código.

- **Plan de Mantenimiento**

Aunque el sistema actual cumple su propósito básico, existen muchas oportunidades de mejora para hacerlo más profesional, escalable y mantenible.

Primero, es fundamental reemplazar el uso de archivos .txt por una base de datos, como SQLserver, o PosgresSQ. Aunque los archivos de texto son simples, tienen muchas limitaciones: no permiten búsquedas eficientes, no garantizan integridad referencial y son vulnerables a corrupción o pérdida de datos. Con una base de datos, cada profesor y estudiante se registraría con un ID único, se podrían usar índices para búsqueda rápida y relaciones entre tablas para mantener coherencia entre notas y profesores.

Segundo, la lógica de negocio está actualmente acoplada a la interfaz gráfica, lo cual complica la prueba unitaria y la reutilización del código. Se recomienda aplicar el patrón MVC (Modelo-Vista-Controlador) o al menos separar claramente la lógica del modelo (registro de profesores y estudiantes) en clases dedicadas, para que la GUI (interfaz) solo se encargue de mostrar y recoger datos.

Tercero, se deben realizar validaciones más estrictas. Por ejemplo:

- Limitar las notas al rango 0–20.
- Verificar que la cédula tenga el formato correcto.

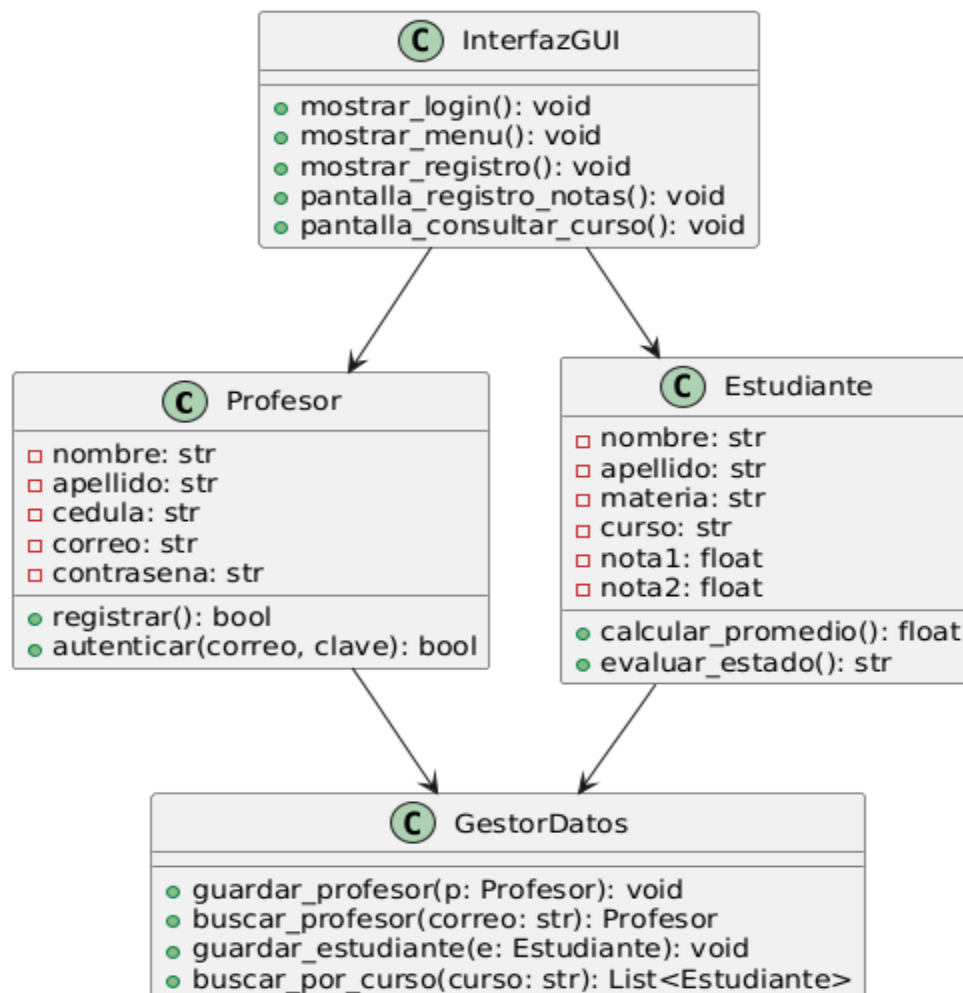
- Verificar que no se registre un profesor con el mismo correo ya existente.
- Asegurar que no se registren estudiantes con campos incompletos o duplicados.

Cuarto, sería recomendable mejorar la experiencia del usuario:

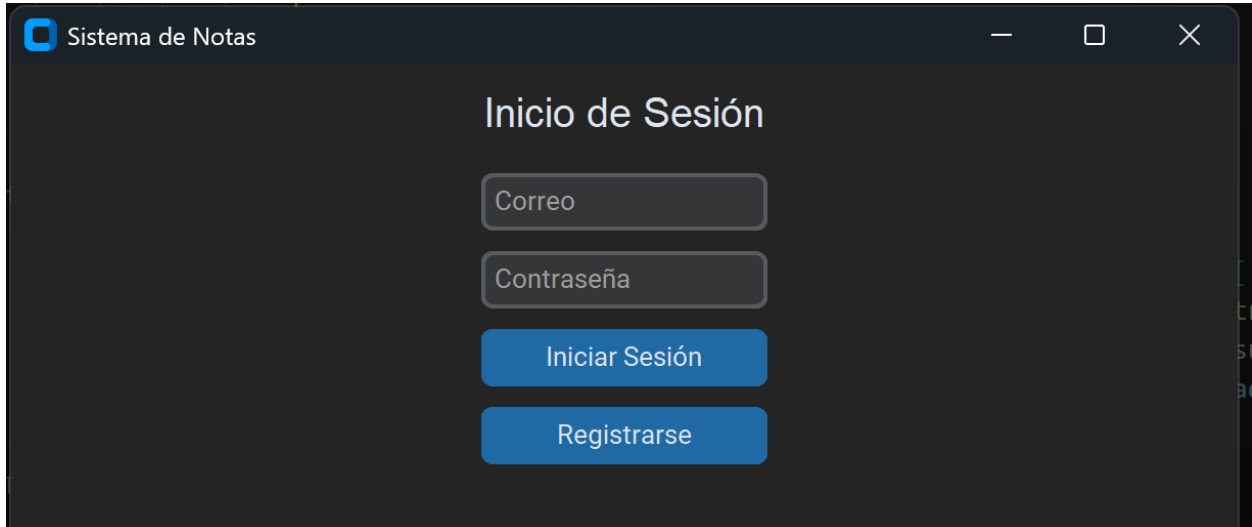
- Añadir mensajes de confirmación visuales con íconos.
- Incluir colores o iconos más atractivos y profesionales.
- Guardar la sesión del profesor para que no tenga que iniciar sesión nuevamente si cierra y abre la app (por ejemplo, usando una cookie local o un archivo .session).

Finalmente, se pueden incluir funcionalidades evolutivas, como exportar datos, agregar reportes por curso, enviar correos automáticos con resultados, y acceso web a través de Flask o Django en el futuro.

### • Prototipo

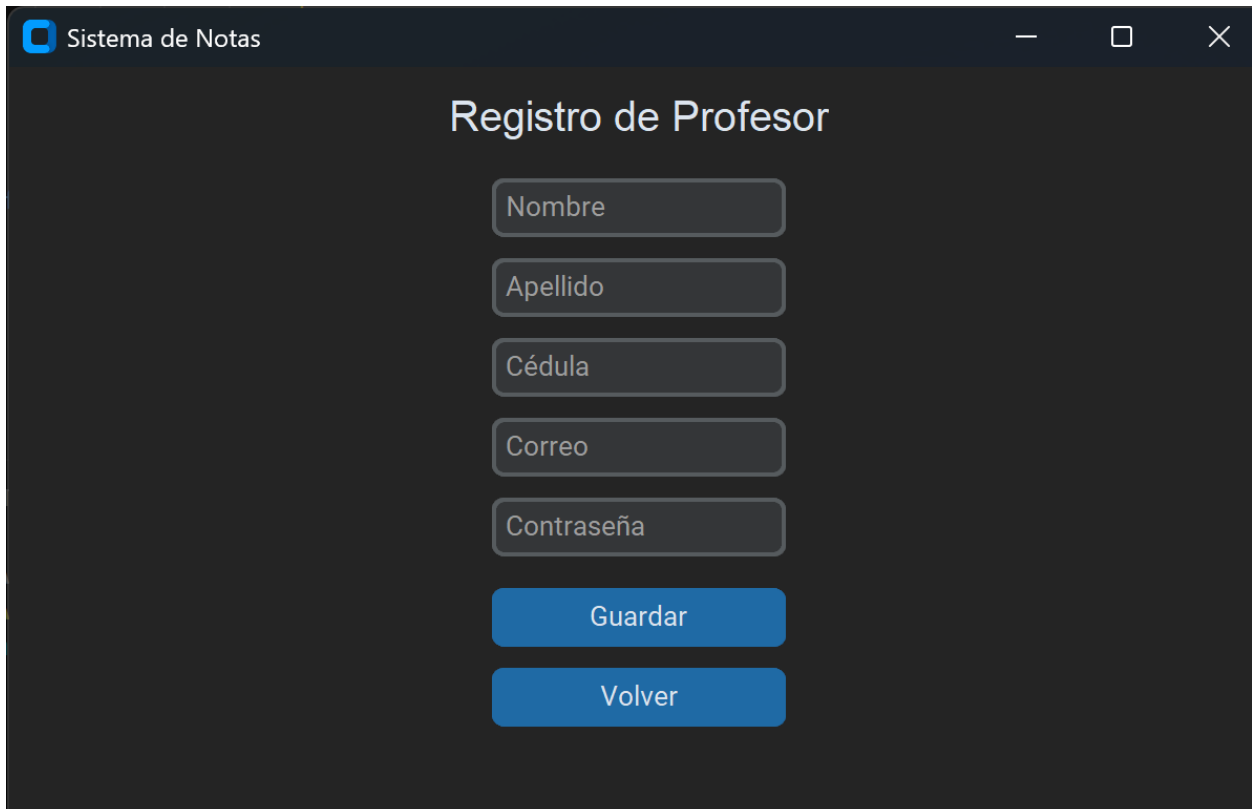


Pantalla de Login.



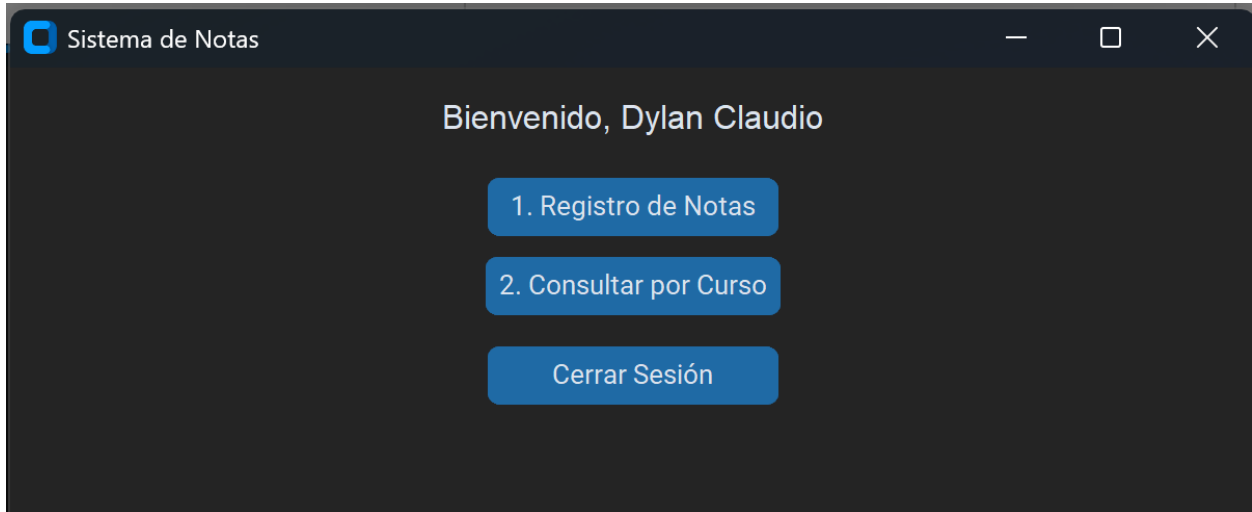
The screenshot shows a web application window titled 'Sistema de Notas'. The main heading is 'Inicio de Sesión'. Below the heading, there are four input fields: 'Correo' (Email), 'Contraseña' (Password), 'Iniciar Sesión' (Login), and 'Registrarse' (Register). The 'Iniciar Sesión' and 'Registrarse' buttons are blue, while the input fields are light gray.

Pantalla de Registro



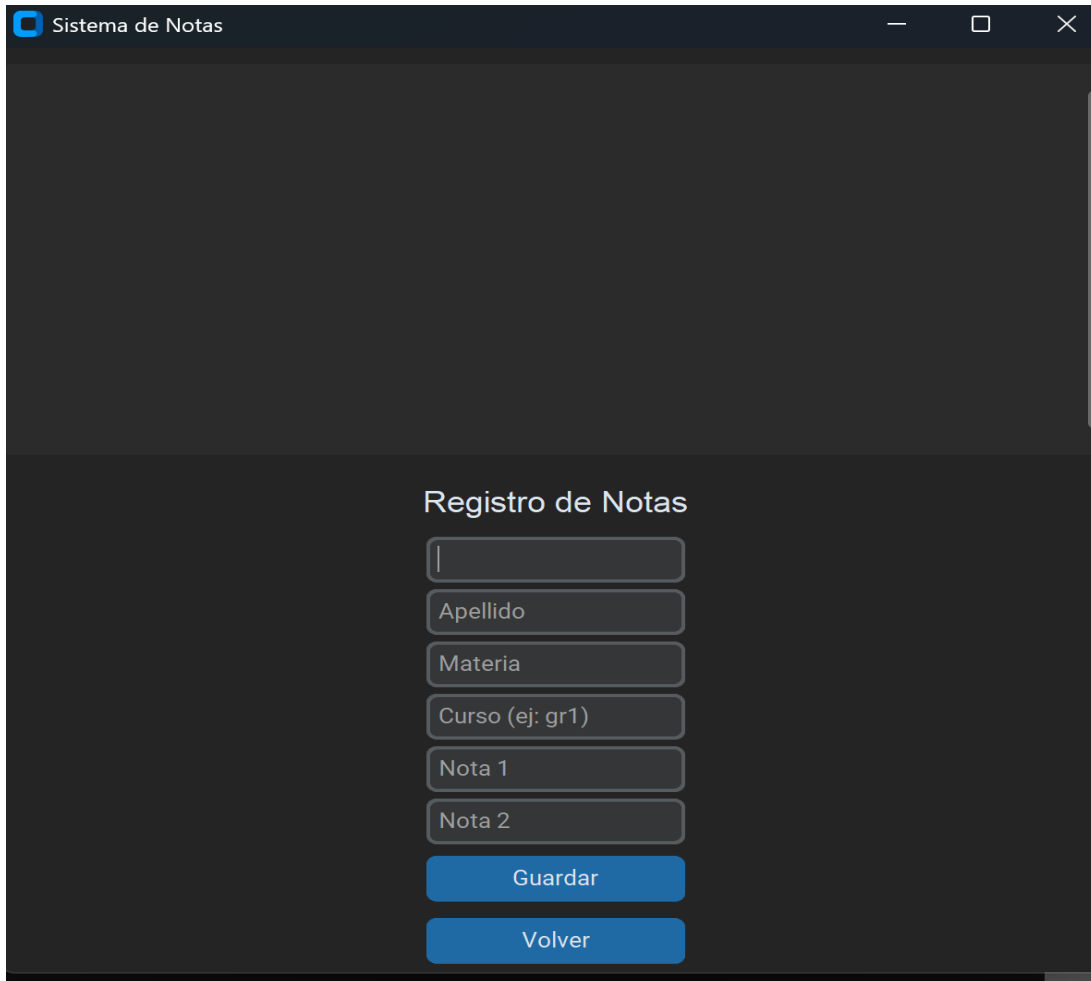
The screenshot shows a web application window titled 'Sistema de Notas'. The main heading is 'Registro de Profesor'. Below the heading, there are five input fields: 'Nombre' (Name), 'Apellido' (Last Name), 'Cédula' (ID Card), 'Correo' (Email), and 'Contraseña' (Password). At the bottom, there are two buttons: 'Guardar' (Save) and 'Volver' (Back). The 'Guardar' and 'Volver' buttons are blue, while the input fields are light gray.

Pantalla de menu principal



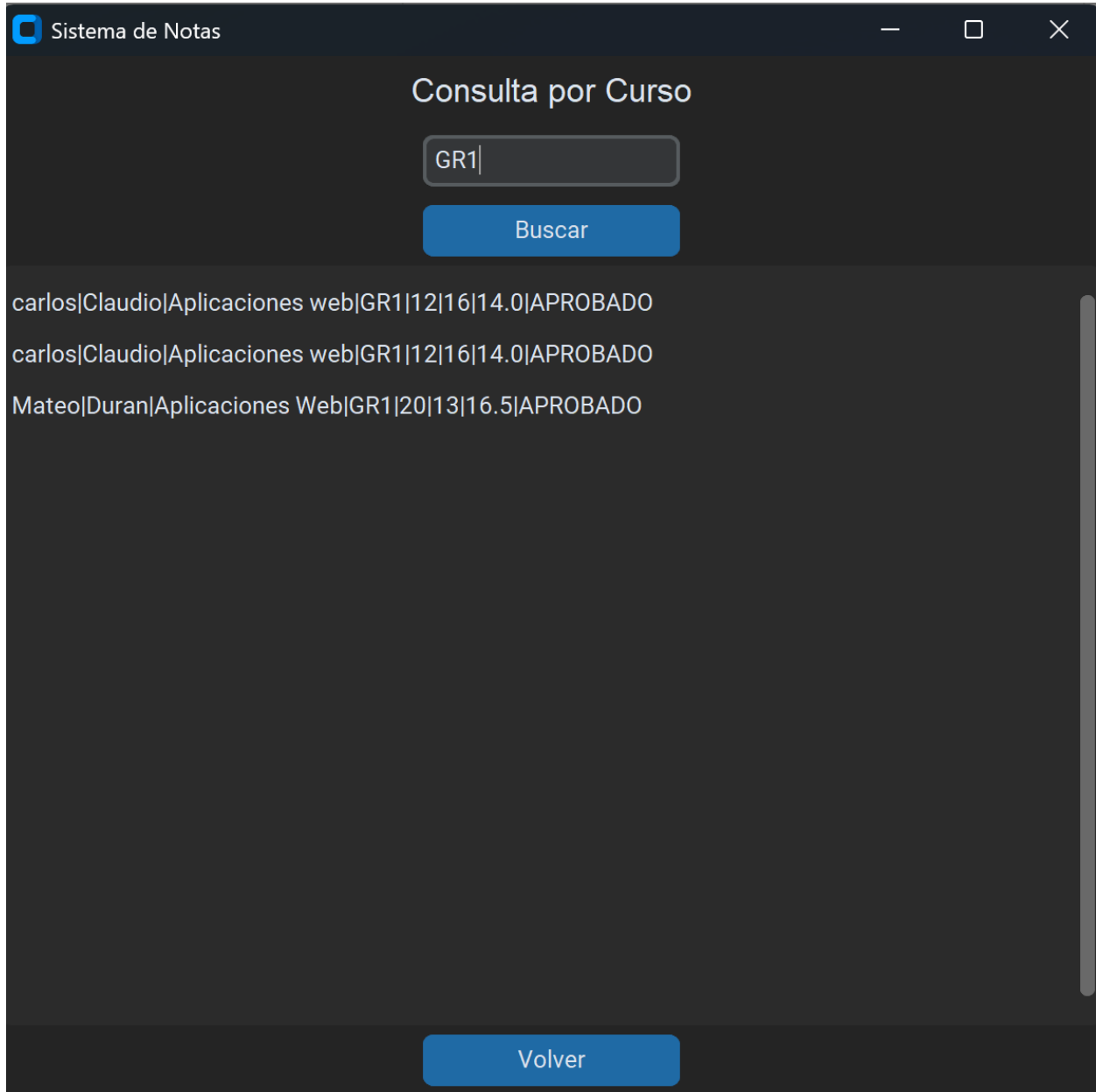
The screenshot shows a web application window titled "Sistema de Notas". The interface has a dark background. At the top, it says "Bienvenido, Dylan Claudio". Below this, there are three blue buttons with white text: "1. Registro de Notas", "2. Consultar por Curso", and "Cerrar Sesión".

Pantalla de registro de notas



The screenshot shows the "Registro de Notas" screen. It features a dark background with a title "Registro de Notas" in white. Below the title, there are several input fields: a text field, a field labeled "Apellido", a field labeled "Materia", a field labeled "Curso (ej: gr1)", a field labeled "Nota 1", and a field labeled "Nota 2". At the bottom, there are two blue buttons with white text: "Guardar" and "Volver".

Pantalla de consulta por curso



Sistema de Notas

### Consulta por Curso

Buscar

carlos|Claudio|Aplicaciones web|GR1|12|16|14.0|APROBADO  
carlos|Claudio|Aplicaciones web|GR1|12|16|14.0|APROBADO  
Mateo|Duran|Aplicaciones Web|GR1|20|13|16.5|APROBADO

Volver

- **Historias de usuario / Casos de Uso**

#### **Historia 1: Iniciar sesión**

Como profesor registrado, quiero poder iniciar sesión con mi correo y contraseña, para acceder al sistema de notas.

**Criterios de Aceptación:**

- El sistema valida que el correo y contraseña coincidan.
- Si son incorrectos, muestra un error.
- Si son correctos, me lleva al menú principal.

### **Historia 2: Registrar notas**

Como profesor, quiero ingresar los datos de los estudiantes y sus notas, para calcular su promedio y estado.

#### **Criterios de Aceptación:**

- Todos los campos son obligatorios.
- Las notas deben ser numéricas.
- El sistema calcula el promedio y muestra el estado (Aprobado/Supletorio/Reprobado).
- El registro se guarda en un archivo.

### **Historia 3: Consultar notas por curso**

Como profesor, quiero buscar estudiantes por curso, para visualizar rápidamente los registros guardados.

#### **Criterios de Aceptación:**

- El profesor puede ingresar un curso.
- Se muestran todos los estudiantes de ese curso con sus notas y estado.

### **Historia 4: Registrarme**

Como nuevo profesor, quiero registrarme con mis datos personales, para poder iniciar sesión en el sistema.

#### **Criterios de Aceptación:**

- Todos los campos deben completarse.
- Al guardar, el usuario puede iniciar sesión inmediatamente.