2023

ENTORNOS DE DESARROLLO

SUBIR NOTA

ALVARO VILLARES RODRIGUEZ
1ºDAM

INDICE

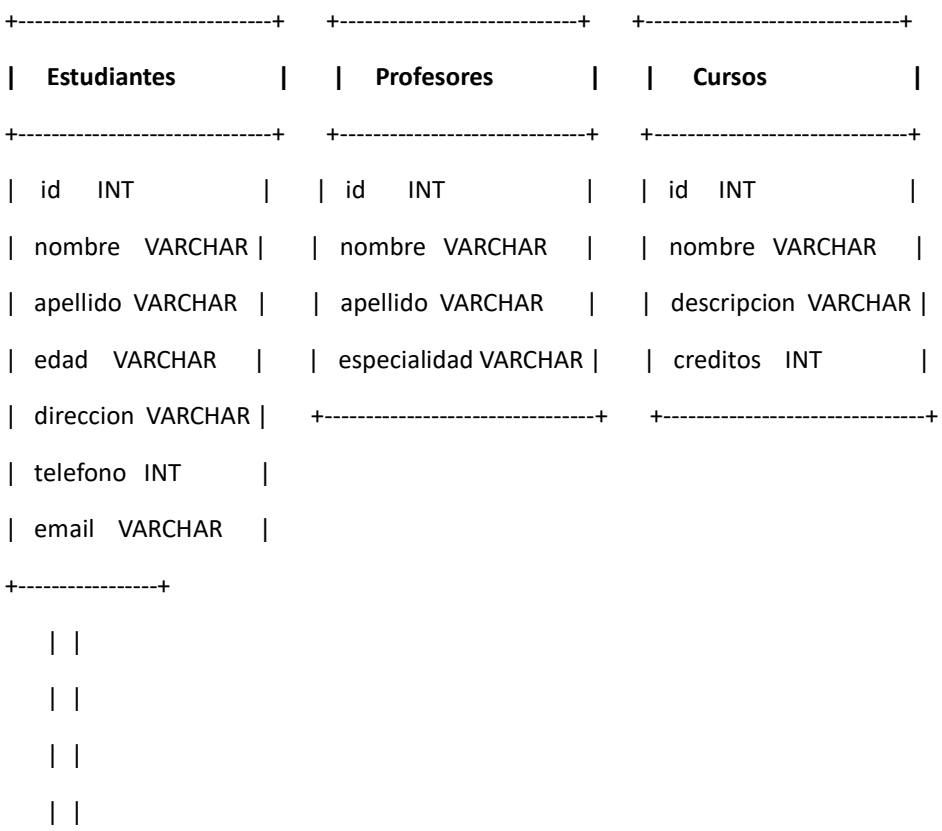
1. Resumen de la investigación

Vamos a simular un proyecto de una aplicación Java que consulte datos de una base de datos de un instituto, donde tendremos Alumnos, Profesores, etc. Todas las gestiones propias de una aplicación de gestión de institutos.

Voy a usar el editor de código **Visual Studio Code**, **ArgoUML** para crear todos los diagramas y **GitHub** para subir los archivos a un repositorio privado.

2. Diagramas de la base de datos

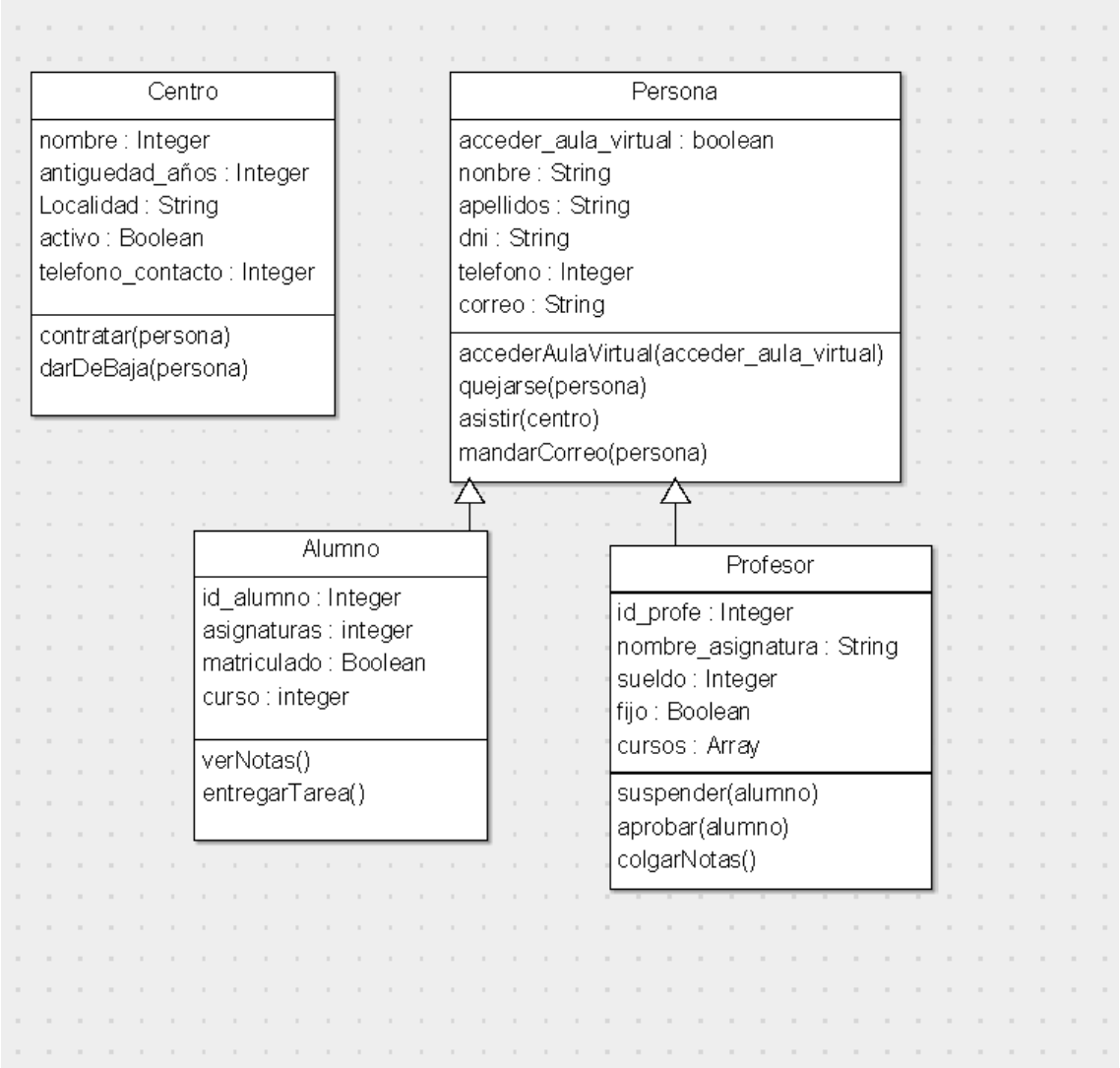
ArgoUML no ofrece herramientas para hacer diagramas de clases de bases de datos, por lo que haré en el mismo Word con las cosas que me pueda permitir



```
| |          +----------------------------+
| +------->|  Calificaciones            |
|            +----------------------------+
|            | id          INT            |
+----------- | id_estudiante INT    |
             | id_curso    INT          |
             | calificacion INT         |
             +---------------------------------+
```

3. Diagramas de clases

Ahora si que empezaremos a usar **ArgoUML** con sus varias herramientas de diagramas

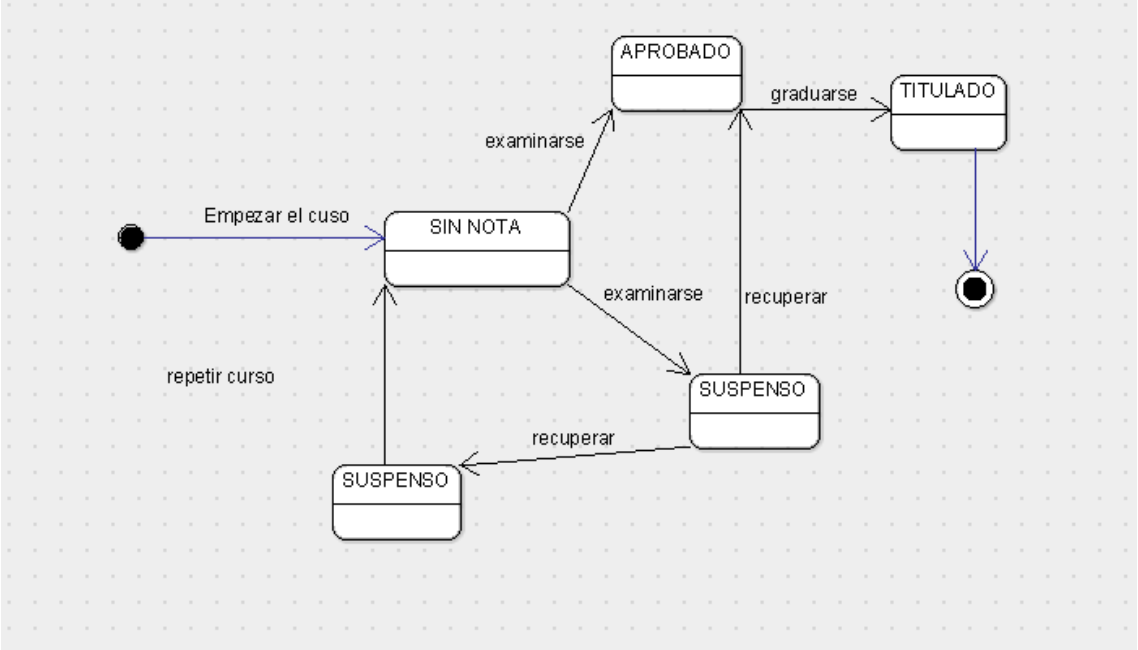


4. Diagramas de comportamiento

Diagramas de comportamiento son:

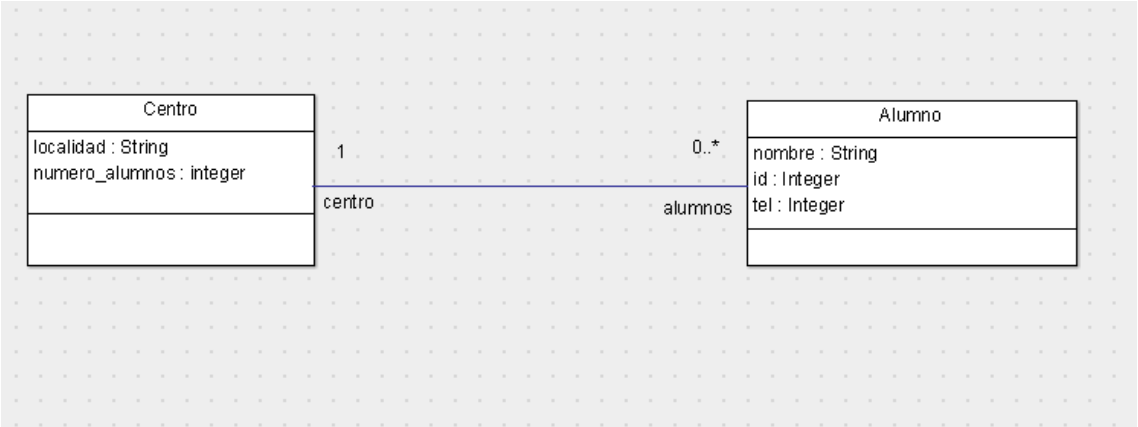
- **Diagramas de casos de uso (en el punto 5 del PDF se explican)**
- **Diagrama de Estado**

Este diagrama de Estado que tenemos delante lo que representa es el paso de un Alumno por un curso. Si aprueba todo a la primera se titulará, pero si suspende tendrá que recuperar o incluso repetir el curso.



Diagramas de relación

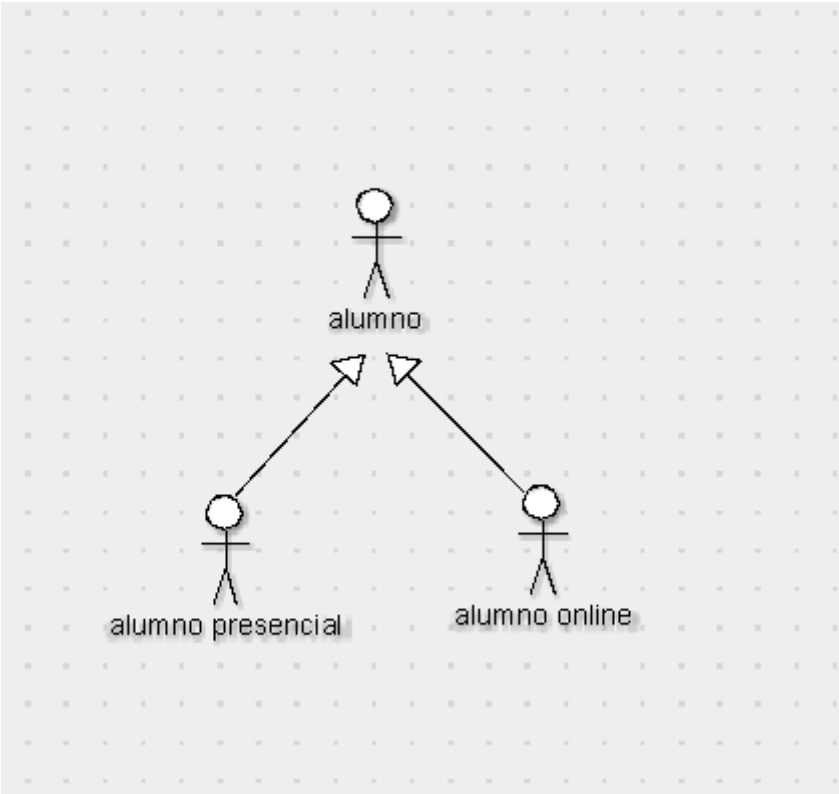
Entiendo como diagramas de relación las distintas relaciones que tiene un factor del diagrama con los demás. Por ejemplo una persona solo puede tener un seguro de vida y el seguro de vida puede tener a muchas personas.



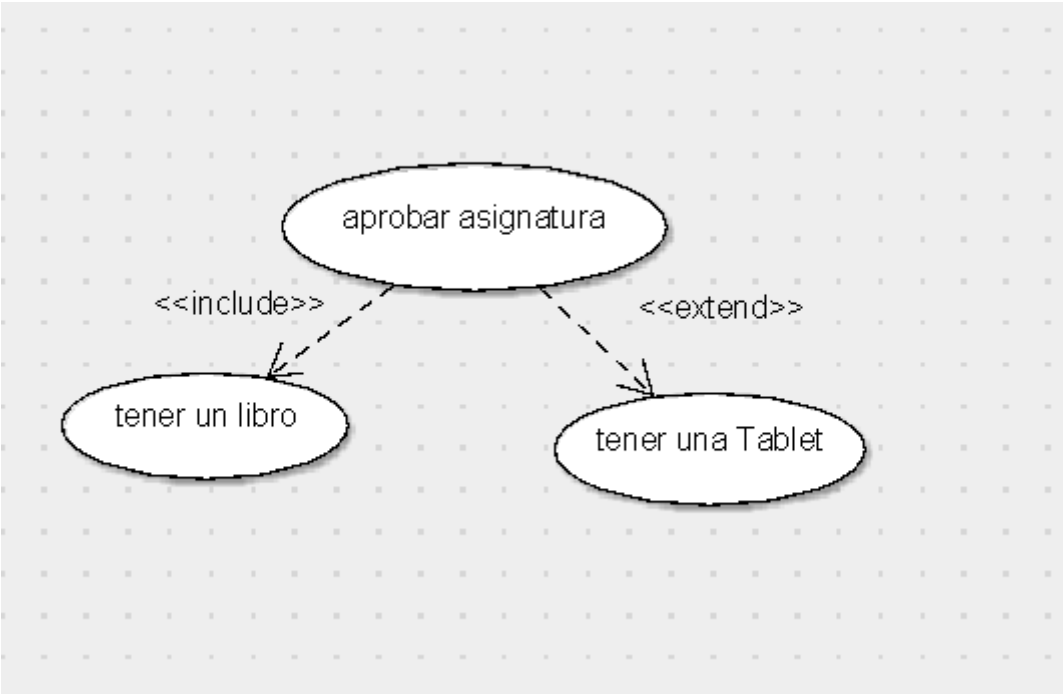
He creado este diagrama de clases que tienen una simple relación. Esta relación consiste en que un centro puede tener de 0 a infinitos alumnos, pero un alumno puede estar únicamente en un centro.

5. Casos de uso

En este caso de uso, tenemos que un alumno puede ser online o presencial, que igualmente es un alumno.



En el siguiente caso de uso, según nos indica, para poder aprobar la asignatura debemos de tener un libro OBLIGATORIAMENTE, pero OPCIONALMENTE nos podemos apoyar de una Tablet.



6. Casos de pruebas de la aplicación

1- Prueba del camino básico

```java
import java.util.Scanner;

public class Alumno {

    public boolean matricula;

    public Scanner sc = new Scanner(System.in);

    public void comprobarMatricula(boolean matricula){
        if(matricula = false){
            System.out.println(x:"El alumno no está matroculado. ¿Quieres matricular al alumno? Y/N");
            String respuesta  ="";
            try{
                respuesta = sc.nextLine();
            }catch(Exception a){
                System.out.println(x:"has introducido mas de un caracter!");
            }

            respuesta = respuesta.toLowerCase();
            char comparar = respuesta.charAt(index:0);
            if(comparar == 'y'){
                matricula = true;
            }
        }else{
            System.out.println(x:"El alumno está matriculado!");
        }
    }
}

```

En este código tenemos varios caminos posibles, y las pruebas de camino básico se tratan de probar esos caminos y encontrar un posible error.

- Si tenemos matrícula, nos dirá que la tenemos
- Si no tenemos matrícula, nos preguntará que si la queremos tener
    - Si no queremos matrícula no hará nada el programa
    - En cambio, si sí la queremos, el programa pondrá en true la matrícula

2- Análisis de valores límites

```
src > Clase.java > ...
```

```java

public class Clase {
    public int contador = 0;
    public int maximos_alumnos = 20;
    public Alumno alumno[] = new Alumno[20];

    public void insertarAlumno(){
        if(!(contador>maximos_alumnos)){
            alumno[contador] = new Alumno();
            contador ++;
        }else{
            System.out.println(x:"La clase ya está llena de alumnos");
        }
    }
}

```

En este ejemplo tenemos una clase con un límite de alumnos permitidos.

Tendremos que probar esos límites y ver lo que pasa:

→¿Qué pasa cuando metemos 20 alumnos?

→¿Qué pasa cuando metemos 21 alumnos, da fallo?

PRUEBAS CON JUNIT

Para la clase Clase he creado unas pruebas con JUNIT, lo que pasa que en VS CODE no coge bien el JUNIT aunque lo importes.

src > ClaseTest.java > ...

```java
import org.junit.Assert;
import org.junit.Test;

public class ClaseTest {

    @Test
    public void testInsertarAlumnoExitoso() {
        Clase clase = new Clase();
        clase.insertarAlumno();

        int contadorEsperado = 1;
        int cantidadAlumnos = clase.contador;

        Assert.assertEquals(contadorEsperado, cantidadAlumnos);
    }

    @Test
    public void testInsertarAlumnoClaseLlena() {
        Clase clase = new Clase();

        // Llenar la clase con alumnos hasta alcanzar el máximo
        for (int i = 0; i < clase.maximos_alumnos; i++) {
            clase.insertarAlumno();
        }

        // Intentar insertar un alumno adicional
        clase.insertarAlumno();

        int contadorEsperado = clase.maximos_alumnos;
        int cantidadAlumnos = clase.contador;

        Assert.assertEquals(contadorEsperado, cantidadAlumnos);
    }
}

```

Básicamente lo que hace el código es llenar la clase al máximo de Alumnos y luego comparar entre el contadorEsperado y la cantidadAlumnos (si son iguales, la prueba no falla y el código no debería fallar).

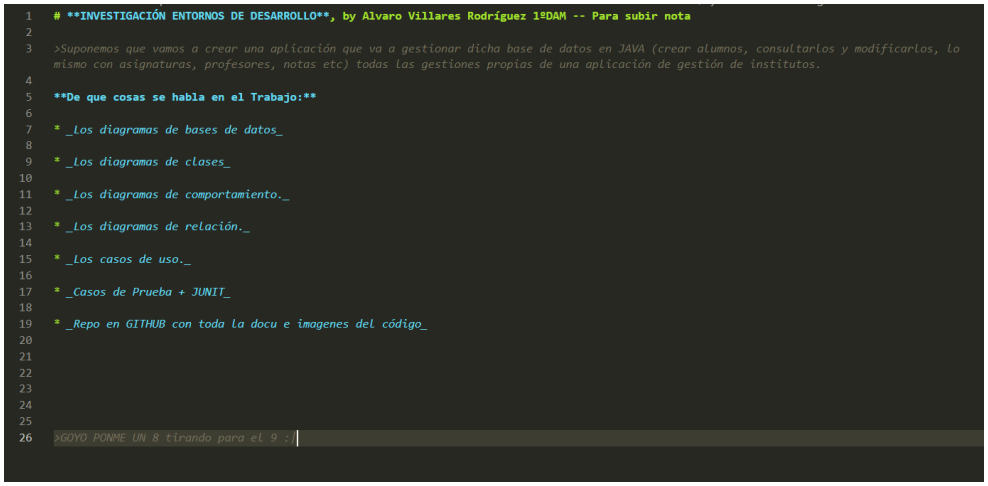
7. GITHUB

¿Qué es GITHUB?

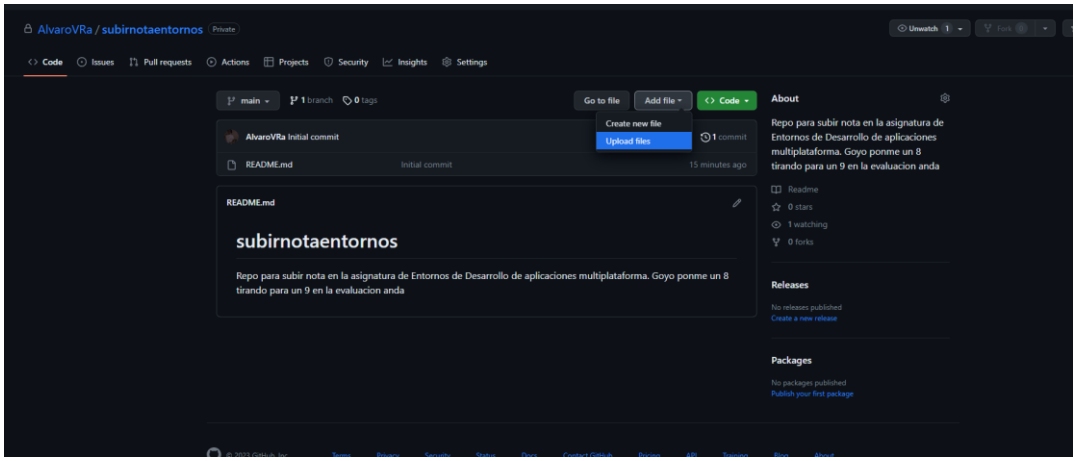
GitHub es una plataforma de alojamiento y colaboración para el desarrollo de software utilizando el sistema de control de versiones Git. Permite a los desarrolladores y equipos de desarrollo compartir, colaborar y administrar proyectos de software de manera eficiente.

He creado un repositorio de GITHUB en el que hay un README.md donde se habla de todo lo que se trata el repositorio. Normalmente casi todos los repositorios serios que están en GITHUB lo tienen.

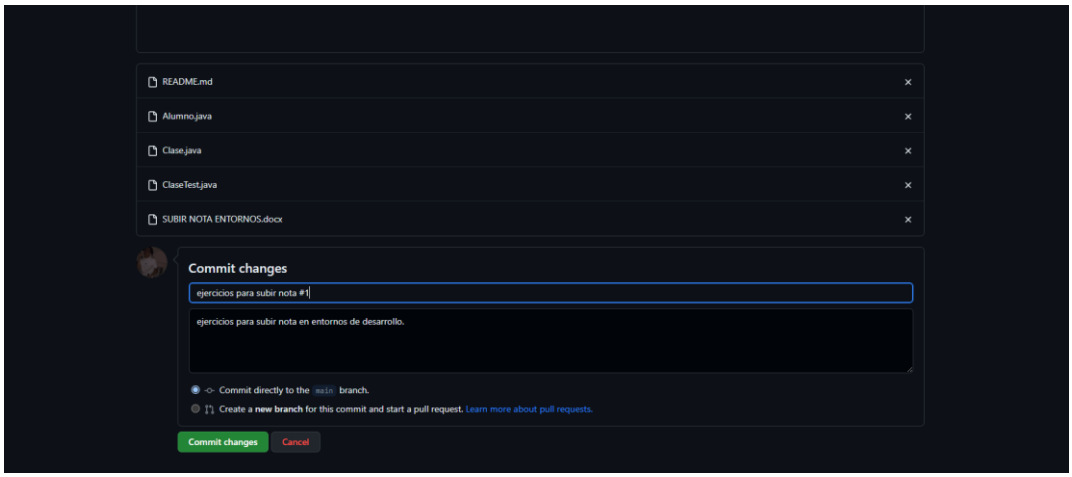
El mío, en VS Code, se ve de esta manera:



En upload files arrastraremos o buscaremos los archivos a subir.



Una vez subidos hacemos un commit de los cambios



DEJO POR AQUÍ EL ENLACE AL GITHUB PARA QUE PUEDAS REVISAR LO QUE QUIERAS (lo dejo público hasta que se corrija el trabajo):

https://github.com/AlvaroVRa/subirnotaentornos