

Proyecto de Programación Orientada a Objetos (Primer Sprint)

Emiliano Luna Casablanca
Eduardo Gutierrez Huerta
Gabriel Alvarez Mendoza
Sebastian Alejandro Nava Diaz

¿De qué trata nuestro proyecto?

Asesorías

Planteo del proyecto (BACKLOG)

- **ALUMNOS:**

- Tendrán su propio usuario (con id y contraseña)
- Acceso a un menú con las siguientes opciones:
- 1- **Visualizar asesorías disponibles** (correspondientes a la carrera del alumno, con sus horarios y profesores)
- 2- **Solicitar asesorías con el cupo disponible** (si se llena una asesoría no podrán inscribirse a ella)
- 3- **Ver asesorías agendadas** (con posibilidad de cancelarlas con anticipación)
- 4- **Algún sistema para contactar a profesores**(Puede ser que haya un chat dentro de la app, o que el profesor pueda agregar un numero o mail para comunicarse)
- 5- (posiblemente) Ver historial de asesorías.
- 6- (posiblemente/otro sprint) Calificar a profesores (solo profesores con quienes hayan tenido asesorías previamente)
- 7- (posiblemente/otro sprint) Realizar comentarios sobre profesores o asesorías.

- **PROFESORES:**

- Tendrán su propio usuario (con id y contraseña)
- Acceso a un menú con las siguientes opciones:
- 1- **Agregar asesorías** (con un cupo máximo y fechas disponibles, al igual que materia y carrera correspondiente)
- 2- **Eliminar asesorías**
- 3- **Visualizar asesorías pendientes**
- 4- **Aceptar/Rechazar alumnos** (viendo quienes se quieren inscribir a las asesorías)
- 5- **Algún sistema para contactar a alumnos** (Puede ser que haya un chat dentro de la app, o que el profesor pueda agregar un numero o mail para comunicarse)

- **ADMINISTRADORES (otro sprint):**

- Tendrán su propio usuario (con id y contraseña)
- Acceso a un menú con las siguientes opciones:
- 1- **Agregar usuarios** (de la app)
- 2- **Eliminar usuarios** (de la app)
- 3- **Ver usuarios** (con perfiles y datos del usuario)

LISTAS NECESARIAS (DONDE ALMACENAREMOS USUARIOS EN LO QUE APRENDEMOS OTRO METODO)

VAMOS A USAR DICCIONARIOS

Ya no se hizo con
listas***

Lista bidimensional para alumnos [Id, nombre, carrera, semestre, contraseña]

Lista bidimensional para profesores [Id, nombre, carrera, semestre, contraseña]

Lista bidimensional para Admin [Id, contraseña]

FUNCIONES NECESARIAS:

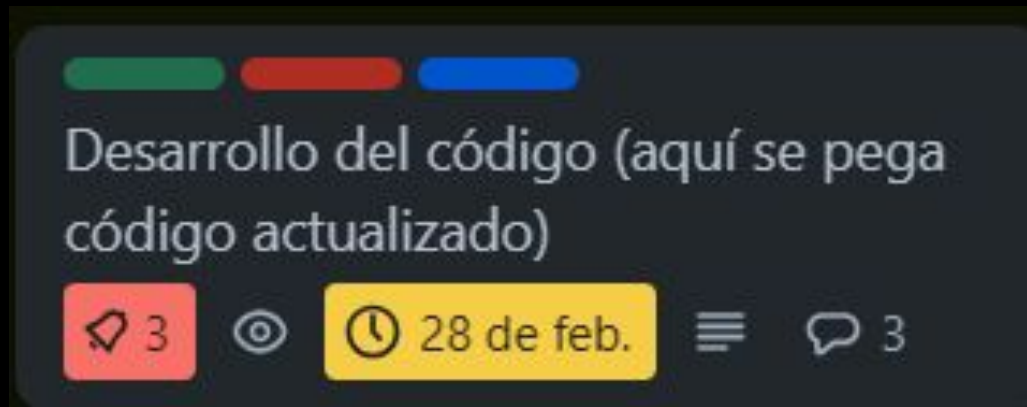
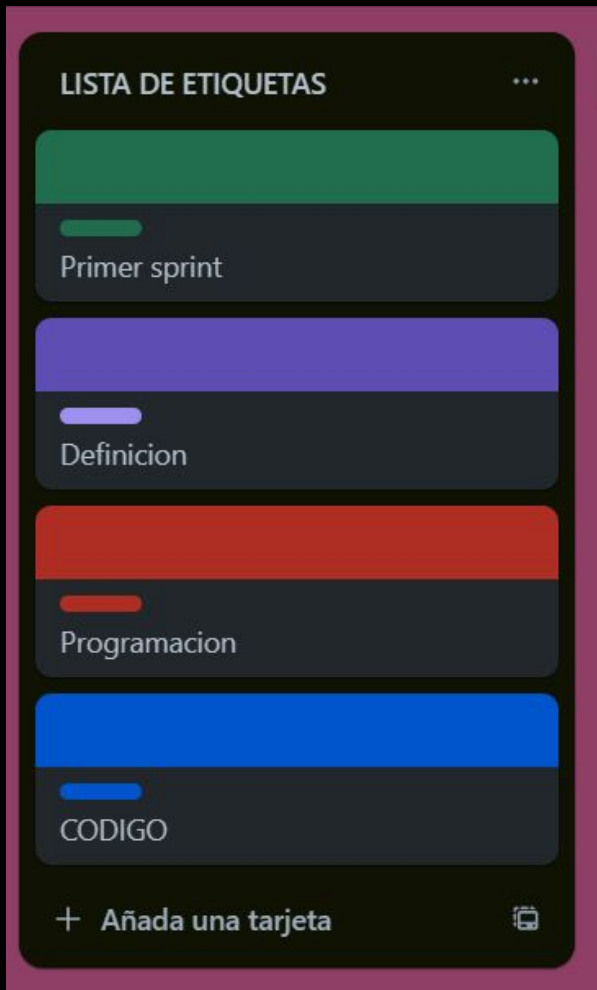
Función de ingreso de usuario (alumno, profesor o Admin)

Función de menú para alumno (cada opción del menú será su propia función)

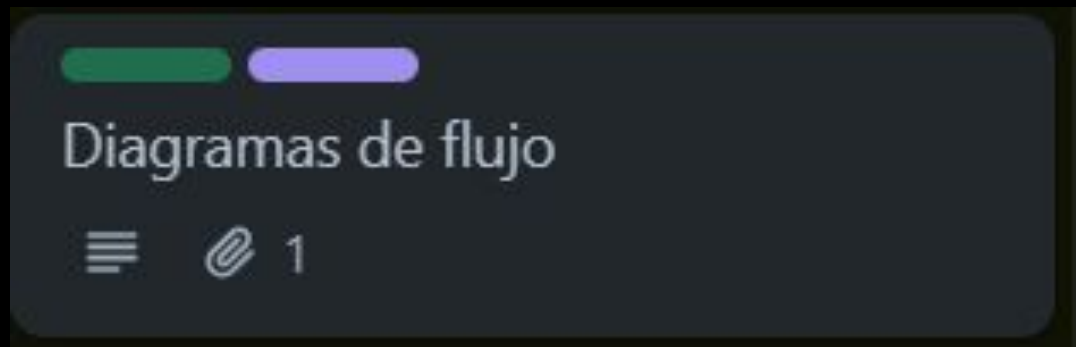
Función de menú para profesor (cada opción del menú será su propia función)

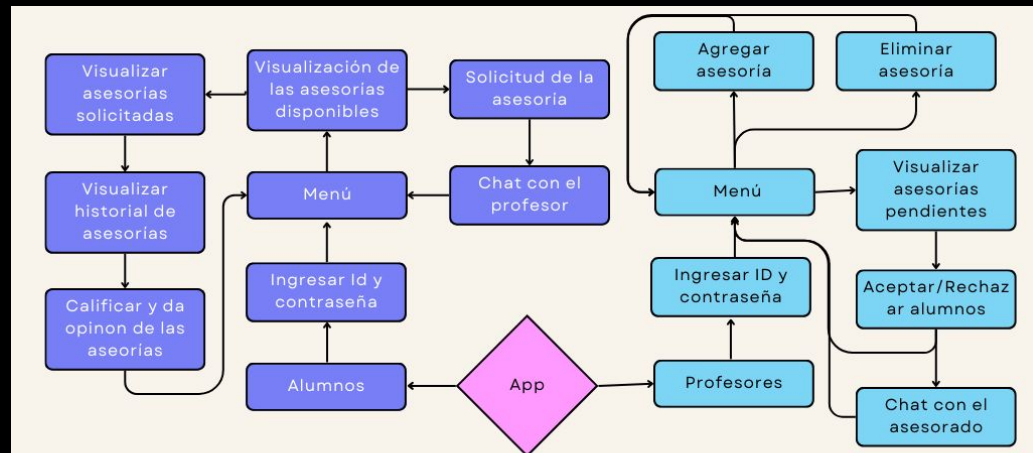
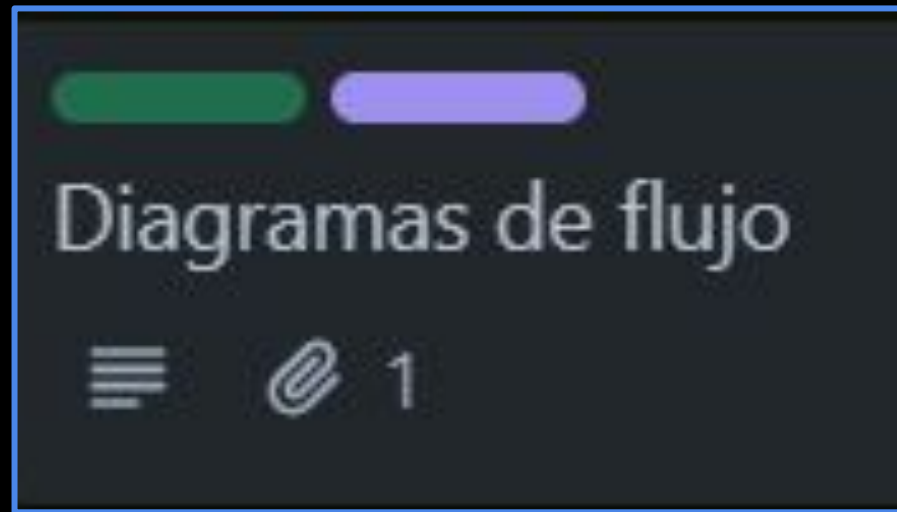
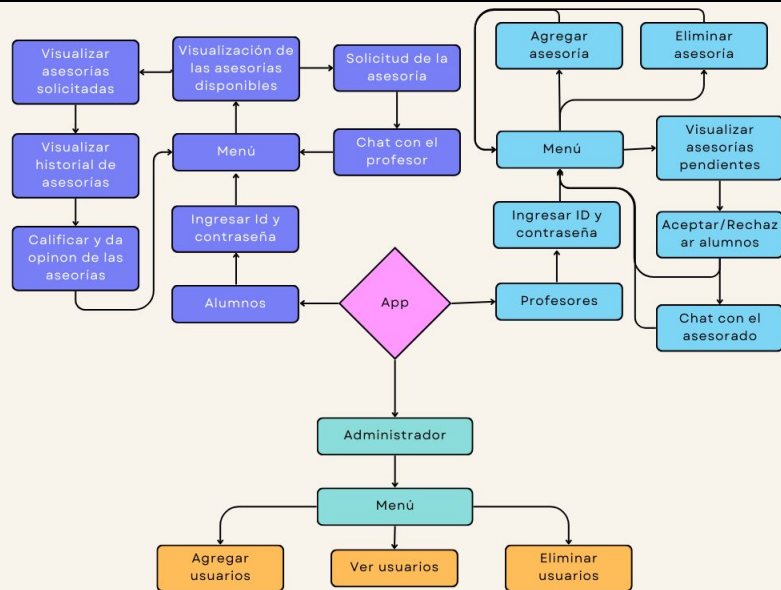
Describir listas y funciones (primer sprint) (LEER TARJETA)

☰ 1



Organización del proyecto





```

# Funcion para escoger entre alumno y profesor y posteriormente ingresar usuario/contraseña
def ingresoUsuario()->int:
    #bandera que determina el tipo de usuario (1= estudiante, 2= profesor, 3= admin)
    band = 0
    while True:
        while True:
            id= str(input("ID:: "))
            if (len(id)==8):
                break
            else:
                print(COLOR_ROJO+"ID incompleto"+FIN_COLOR)
        id = id.upper()
        if id[-1] in {"E","P","A"} and id[0:7].isdigit():
            if id[-1] == "E":
                if id in estudiantes.keys():
                    contraseña = str(getpass.getpass("Contraseña:: "))
                    if contraseña == estudiantes[id]["contrasena"]:
                        print(COLOR_VERDE+"Acceso correcto"+FIN_COLOR)
                        return(1)
                    else:
                        print(COLOR_ROJO+"Contraseña incorrecta"+FIN_COLOR)
                elif id[-1] == "P":
                    if id in profesores.keys():
                        contraseña = str(getpass.getpass("Contraseña:: "))
                        if contraseña == profesores[id]["contrasena"]:
                            print(COLOR_VERDE+"Acceso correcto"+FIN_COLOR)
                            return(2)
                        else:
                            print(COLOR_ROJO+"Contraseña incorrecta"+FIN_COLOR)
                    else:
                        print(COLOR_ROJO+"Usuario no encontrado"+FIN_COLOR)
                        break
            else:
                print(COLOR_ROJO+"Usuario incorrecto"+FIN_COLOR)
        return(0)

```

¿Que se hizo
en este
primer sprint?


```
# Funcion para ver asesorias disponibles
def verAsesorias():
    while True:
        for x in asesorias.keys():
            print(x)
        Info_Clase = str(input("De que clase quieres ver horarios?")).upper()
        if Info_Clase in asesorias.keys():
            print(COLOR_AZUL_BRILLANTE+Info_Clase+FIN_COLOR)
            for k in range(len(asesorias[Info_Clase])):
                num_clase = f"clase{k + 1}"
                print(COLOR_AZUL_NEGRITA+num_clase[0:-1].upper() +f" {k+1}" +FIN_COLOR)
                print("HORARIO")
                print(asesorias[Info_Clase][num_clase]["horario"])
                print("DIA")
                print(asesorias[Info_Clase][num_clase]["dias"])
                print("PROFESOR")
                print(profesores[asesorias[Info_Clase][num_clase]["Profesor"]]["nombre"])
                print("CUPO")
                print(asesorias[Info_Clase][num_clase]["cupos"])
            break
        else:
            print(COLOR_ROJO+"No se encontro la clase"+FIN_COLOR)

    return
```

HACER FUNCION VER ASESORIAS
(ALUMNOS) (completado por lalo)




```
# Funcion para crear nueva asesoria (PRIMER SPRINT)
def agregarAsesorias():
    while True:
        clase = {}
        NombreMateria = input("¿Cual es el nombre de la materia que vas a agregar una asesoria?: ")
        if NombreMateria in asesorias:
            IDMateria = input("¿Cual es el ID de la asesoria?: ")
            horario = input("¿Cual va a ser el horario de la asesoria?: ")
            dias = input("¿Que dias va ser la asesorias?: ")
            while True:
                cupo = int(input("¿Cual es el cupo de maximo de la asesoria?: "))
                if cupo >= 1 and cupo <= 30 :
                    clase["horario"] = horario
                    clase["profesor"] = id
                    clase["dias"] = dias
                    clase["cupo"] = cupo
                    asesorias[NombreMateria][IDMateria] = clase
                    print (asesorias)
                    break
                else:
                    print("El cupo no puede ser menor a 1 y mayor a 30")
            else:
                print("Esa materia no existe")
```

HACER FUNCION AGREGAR
ASESORIAS (PROFESORES)
(completado por luna)



Funcion para desplegar menu de alumno (anidar cada funcion del menu de alumno a esta funcion) (PRIMER SPRINT)

```
def menuAlumno():
    opc = 0
    print("MENU ALUMNOS")
    while(True):

        print("1. Ver asesorias disponibles")
        print("2. Solicitar ingreso a asesoria")
        print("3. Ver asesorias inscritas")
        print("4. Contactar a profesores")
        print("5. Regresar a inicio de sesion/FIN")
        opc = int(input("Ingrese opcion deseada: "))
        print("")

        if opc == 1:
            verAsesorias()
        elif opc == 2:
            solicitarAsesoria()
        elif opc == 3:
            verAsesoriasInscritas()
        elif opc == 4:
            contactarProfesores()
        elif opc == 5:
            print("Cerrando menu de alumno, gracias")
            break
        else:
            print(COLOR_ROJO+"OPCION INVALIDA"+FIN_COLOR)
            print("")
```



HACER FUNCION MENU DE
ALUMNOS (completado)

```
# Funcion para desplegar menu de profesor (anidar cada funcion del menu de profesor a esta funcion) (PRIMER SPRINT)
def menuProfesor():
    opc = 0
    print("MENU PROFESORES")
    while(True):

        print("1. Agregar asesoria para alumno")
        print("2. Eliminar asesoria existente")
        print("3. Ver asesorias pendientes(a futuro)")
        print("4. Administrar alumnos en asesorias")
        print("5. Contactar a alumnos")
        print("6. Regresar a inicio de sesion/FIN")
        opc = int(input("Ingrese opcion deseada: "))
        print("")

        if opc == 1:
            agregarAsesorias()


        elif opc == 2:
            eliminarAsesorias()

        elif opc == 3:
            verAsesoriasPendientes()

        elif opc == 4:
            administrarAlumnos()

        elif opc == 5:
            contactarAlumnos()

        elif opc == 6:
            print("Cerrando menu de profesor, gracias")
            break
        else:
            print(COLOR_ROJO+"OPCION INVALIDA"+FIN_COLOR)
            print("")
```



HACER FUNCION MENU DE
PROFESORES (completado)

Función adicional-contactar profesores

```
# Funcion para comunicarse con profesores de asesorias a las que se esta inscribiendo

def contactarProfesores(lista_prof : dict):
    #for que pasa por el diccionario de profesores y muestra su id y nombres
    for n,Profesor in enumerate(lista_prof):
        print(n+1, " :")
        print("ID del profesor: " ,Profesor)
        print(lista_prof[Profesor]["nombre"])
    #while para que se repita si es que no seleccionan una de las opciones
    while True:
        seleccion = int((input("Seleccione el profesor con el que desea hablar: ")))
        if seleccion > len(lista_prof) or seleccion < 1:
            print("Seleccione una de las opciones por favor")
        else:
            mensaje = str(input("Deposite el mensaje que quiere dar al profesor: "))
            print("El mensaje ha sido enviado.")
            break
    return
```

Se junta todo en función llamada main, ahí se llaman los menús y el ingreso.

```
#Codigo principal
def main():
    usuario = ingresoUsuario()
    if usuario == 1:
        menuAlumno()
    elif usuario == 2:
        menuProfesor()
    elif usuario == 3:
        print("PLACEHOLDER MENU ADMIN")
    else:
        print("SIN USUARIO")
    return
```

*Correr código

Preguntas?

GRACIAS
POR SU
ATENCIÓN