

# PRIMER SPRINT

---

Seguimiento y Progreso Mediante la Evaluación Sistemática.

-Jhon Jairo Vera Acevedo

-Deyner Alberto Contreras Sandoval

-Nicolás Steven Pineda Rodríguez



Backlog del Proyecto 2: Sistema de Gestión de Estudiantes y Calificaciones

SPRINT 1

Duración: 18 Noviembre – 25 Noviembre



BacklogID	Tarea	Descripción	PuntosHistoria	Estado	Observaciones
S1-T01	Documentacion inicial	Llenar la documentacion inicial para iniciar el proyecto	5	Por hacer	
S1-T02	Configurar proyecto Java Web	Creación del proyecto base y estructura MVC.	3	Por hacer	
S1-T03	Clase Estudiante	Implementar atributos y métodos.	2	Por hacer	
S1-T04	Clase GestorEstudiantes	CRUD inicial en memoria.	3	Por hacer	
S1-T05	Prototipo consola/Endpoints	Probar lógica inicial.	3	Por hacer	
S1-T06	Validaciones iniciales	Evitar duplicados y datos inválidos.	2	Por hacer	
S1-T07	Historia agregar estudiantes	Funcionalidad completa de registro.	3	Por hacer	

SPRINT 2

Duración: 25 Noviembre – 02 Diciembre

BacklogID	Tarea	Descripción	PuntosHistoria	Estado	Observacion
-----------	-------	-------------	----------------	--------	-------------

# BACKLOG



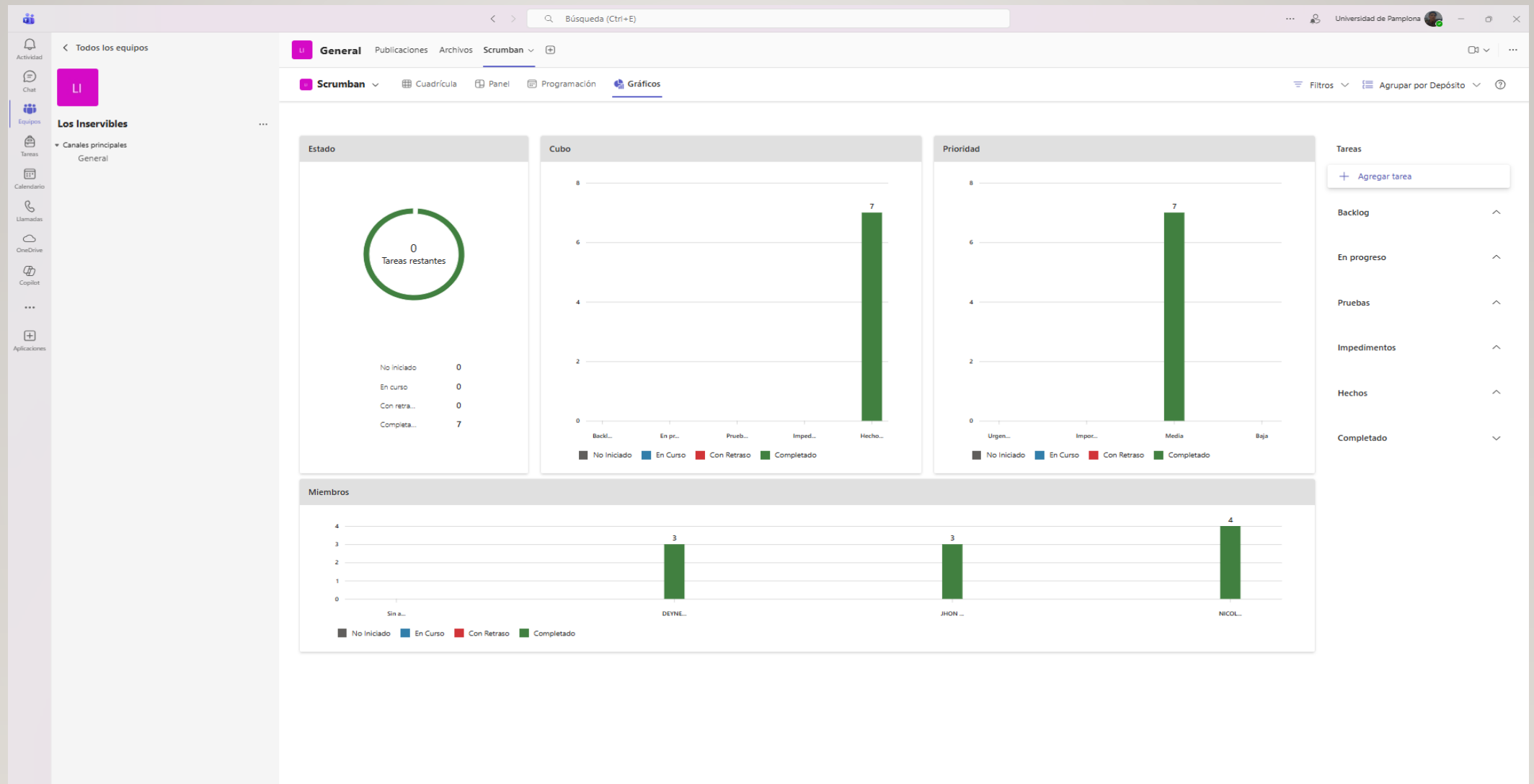
S2-T01	Ingreso/actualización de calificaciones	Agregar y editar notas.	3	Por hacer
S2-T02	Función promedio	Implementar calcularPromedio.	2	Por hacer
S2-T03	Servicio Web Estudiantes	Capa de servicio backend.	5	Por hacer
S2-T04	Controladores Web	Endpoints REST CRUD.	5	Por hacer
S2-T05	Vista web básica	Formularios HTML para estudiantes.	5	Por hacer
S2-T06	Consultar estudiantes y promedios	Vista con tabla.	3	Por hacer
S2-T07	Validaciones completas	Frontend y backend.	3	Por hacer
S2-T08	Página Acerca de	Ficha técnica del equipo.	2	Por hacer

### SPRINT 3


Duración: 02 Diciembre – 10 Diciembre

BacklogID	Tarea	Descripción	PuntosHistoria	Estado	Observaciones
S3-T01	Integración frontend-backend	Conectar formularios con API.	5	Por hacer	
S3-T02	Eliminar estudiantes	Funcionalidad desde web.	3	Por hacer	
S3-T03	Mejorar página de consulta	Filtros y ordenamientos.	5	Por hacer	
S3-T04	Pruebas funcionales	Validación completa del sistema.	5	Por hacer	

# TAREAS EN TEAMS




# SPRINT PLANNING

	SPMES					9812348	
						57 3214740490	
						EMAIL	<a href="mailto:jhon.verajaac@unipamplona.edu.co">jhon.verajaac@unipamplona.edu.co</a>
						CARGO	PRODUCT OWNER
						GRUPO	2
DATOS SPRINT	SPRINTS						
	FECHA INICIO	FECHA FIN	TAREAS SPRINT	TAREAS ENTREGADAS	CUMPLIMIENTO		
SPRINT 1	18/11/2025	25/11/2025	7	7	100%		
SPRINT 2	25/11/2025	2/11/2025	8	0	0%		
SPRINT 3	2/11/2025	10/11/2025	4	0	0%		



# SCRUM ROLES



SPMES (Seguimiento y Progreso Mediante la Evaluación Sistemática)

NIT: 893218741

CONTACTO:

EMAIL

[jhon.verajaac@gmail.com](mailto:jhon.verajaac@gmail.com)

TELEFONO

[3214740390](tel:3214740390)

GRUPO

2

Rol	Descripción	Encargado(s)
Product Owner	Es responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo. Define la visión del producto, gestiona el backlog del producto y asegura que el equipo trabaje en las tareas con mayor prioridad.	JHON JAIRO VERA ACEVEDO
Scrum Master	Es responsable de asegurar que Scrum se entienda y se implemente correctamente. Facilita eventos de Scrum, elimina impedimentos y ayuda al equipo a mejorar continuamente.	NICOLÁS STEVEN PINEDA RODRÍGUEZ
Development Team	Es un grupo de profesionales que trabajan juntos para entregar incrementos de producto listos para la producción. Son auto-organizados y multifuncionales, con todas las habilidades necesarias para crear un incremento de producto.	DEYNER ALBERTO CONTRERAS SANDOVAL

# CLASE GESTIÓN DE ESTUDIANTES

Esta clase se encarga de crear la lista  
Donde se guardaran los estudiantes y  
verificar que el id sea distinto y dar  
los datos mediante el id.

```
Help  <  >  Search
GestorEstudiante[1].java  fibonacci[1].java  RaizDigital[1].java
C: > Users > usuario > AppData > Local > Microsoft > Windows > INetCache > IE > XRGNJTK3 > J GestorEstudiante[1].java
1  import java.util.ArrayList;
2
3  /*
4   * Clase: GestorEstudiantes
5   * Administra la lista de estudiantes del sistema.
6   */
7  public class GestorEstudiante {
8
9      private ArrayList<Estudiante> estudiantes; // Lista de estudiantes registrados
10
11     /*
12      * Inicializa la lista vacía al crear el gestor
13      */
14     public GestorEstudiante() {
15         estudiantes = new ArrayList<>();
16     }
17
18     /*
19      * Agrega un nuevo estudiante si el ID no está repetido
20      */
21     public boolean agregarEstudiante(String id, String nombre) {
22         if (buscarPorId(id) != null) {
23             return false; // Ya existe
24         }
25         estudiantes.add(new Estudiante(id, nombre));
26         return true;
27     }
28
29     /*
30      * Busca un estudiante por su ID
31      */
32     public Estudiante buscarPorId(String id) {
33         for (Estudiante e : estudiantes) {
34             if (e.getId().equals(id)) {
35                 return e;
36             }
37         }
38         return null;
39     }
}
```

C:\> Users > usuario > AppData > Local > Microsoft > Windows > INetCache > IE > KJV3V2D > J Estudiante[1].java > Estudiante > agregarCalificacion(double calificacion)

```
1  import java.util.ArrayList;
2
3  /*
4   * Clase: Estudiante
5   * Representa a un estudiante con ID, nombre y calificaciones.
6   */
7  public class Estudiante {
8
9      private String id;           // Identificador del estudiante
10     private String nombre;       // Nombre del estudiante
11     private ArrayList<Double> listaCalificaciones; // Lista de calificaciones
12
13     /*
14     * Constructor que inicializa ID, nombre y lista vacía de calificaciones
15     */
16     public Estudiante(String id, String nombre) {
17         this.id = id;
18         this.nombre = nombre;
19         this.listaCalificaciones = new ArrayList<>();
20     }
21
22     /*
23     * Devuelve el ID del estudiante
24     */
25     public String getId() {
26         return id;
27     }
28
29     /*
30     * Devuelve el nombre del estudiante
31     */
32     public String getNombre() {
33         return nombre;
34     }
35
36     /*
37     * Agrega una calificación a la lista del estudiante
38     */
39     public void agregarCalificacion(double calificacion) {
```

# CLASE ESTUDIANTE

Se encarga de inicializar los datos del estudiante y crear la lista donde se guardaran las calificaciones y retornar el promedio de las calificaciones ingresadas.



C: > Users > usuario > AppData > Local > Microsoft > Windows > INetCache > IE > KJYV3V2D > J Estudiante[1].java > Estudiante > agregarCalificacion(double calificacion)

```
7 public class Estudiante {
39     public void agregarCalificacion(double calificacion) {
41     }
42
43     /*
44     * Calcula y retorna el promedio de las calificaciones.
45     * Si no hay calificaciones, devuelve 0.
46     */
47     public double calcularPromedio() {
48         if (listaCalificaciones.isEmpty()) {
49             return 0.0;
50         }
51         double suma = 0;
52         for (Double c : listaCalificaciones) {
53             suma += c;
54         }
55         return suma / listaCalificaciones.size();
56     }
57
58     /*
59     * Devuelve la cantidad de calificaciones registradas
60     */
61     public int cantidadCalificaciones() {
62         return listaCalificaciones.size();
63     }
64
65     /*
66     * Devuelve una representación en texto del estudiante
67     * (Sin @Override porque el profesor no lo permite)
68     */
69     public String toString() {
70         return "ID: " + id +
71             " | Nombre: " + nombre +
72             " | Calificaciones registradas: " + cantidadCalificaciones() +
73             " | Promedio: " + calcularPromedio();
74     }
75 }
76
```

# MAIN

---

```
C: > Users > usuario > AppData > Local > Microsoft > Windows > INetCache > IE > ROA1P9SQ > J Main[1].java > ...  
1  import java.util.Scanner;  
2  
3  /*  
4   * Clase principal con menú para gestionar estudiantes y calificaciones.  
5   */  
6  public class Main {  
7  
8      public static void main(String[] args) {  
9  
10         Scanner scanner = new Scanner(System.in);  
11         GestorEstudiante gestor = new GestorEstudiante();  
12         int opcion;  
13  
14         do {  
15             System.out.println("\n=== SISTEMA DE GESTIÓN DE ESTUDIANTES ===");  
16             System.out.println("1. Agregar estudiante");  
17             System.out.println("2. Consultar estudiantes y promedios");  
18             System.out.println("3. Agregar calificación a estudiante");  
19             System.out.println("4. Eliminar estudiante");  
20             System.out.println("5. Salir");  
21             System.out.print("Seleccione una opción: ");  
22  
23             opcion = scanner.nextInt();  
24             scanner.nextLine();  
25  
26             switch (opcion) {  
27  
28                 case 1:  
29                     System.out.print("Ingrese ID del estudiante: ");  
30                     String id = scanner.nextLine();  
31  
32                     System.out.print("Ingrese nombre del estudiante: ");  
33                     String nombre = scanner.nextLine();  
34  
35                     if (gestor.agregarEstudiante(id, nombre)) {  
36                         System.out.println("✅ Estudiante agregado correctamente.");  
37                     } else {  
38                         System.out.println("❌ Ya existe un estudiante con ese ID.");  
39                     }  
40                 }  
41             } while (opcion != 5);  
42         }  
43     }  
44 }
```

```

6   public class Main {
8       public static void main(String[] args) {
39           }
40           break;
41
42       case 2:
43           System.out.println("\n--- LISTA DE ESTUDIANTES ---");
44
45           if (gestor.consultarEstudiantes().isEmpty()) {
46               System.out.println("No hay estudiantes registrados.");
47           } else {
48               for (Estudiante e : gestor.consultarEstudiantes()) {
49                   System.out.println(e.toString());
50               }
51           }
52           break;
53
54       case 3:
55           System.out.print("Ingrese ID del estudiante: ");
56           String idCal = scanner.nextLine();
57
58           System.out.print("Ingrese calificación (0 a 5): ");
59           double nota = scanner.nextDouble();
60
61           if (nota < 0 || nota > 5) {
62               System.out.println("⚠ La calificación debe estar entre 0 y 5.");
63               break;
64           }
65
66           if (gestor.agregarCalificacionAEstudiante(idCal, nota)) {
67               System.out.println("✅ Calificación agregada.");
68           } else {
69               System.out.println("❌ No existe un estudiante con ese ID.");
70           }
71           break;
72
73       case 4:

```

```

72
73     case 4:
74         System.out.print("Ingrese ID del estudiante a eliminar: ");
75         String idEliminar = scanner.nextLine();
76
77         if (gestor.eliminarEstudiante(idEliminar)) {
78             System.out.println("✅ Estudiante eliminado.");
79         } else {
80             System.out.println("❌ No existe un estudiante con ese ID.");
81         }
82         break;
83
84     case 5:
85         System.out.println("Saliendo del sistema...");
86         break;
87
88     default:
89         System.out.println("⚠️ Opción no válida.");
90 }
91
92 } while (opcion != 5);
93
94 scanner.close();
95 }
96 }
97

```