SPRINT PLANNING

ERP ACADEMICO/ADMINISTRATIVO

ARQUITECTURA DE SOFTWARE

HEMER SANTIAGO PEREZ NIEVES

MIGUEL ANGEL ROMERO ALFONSO

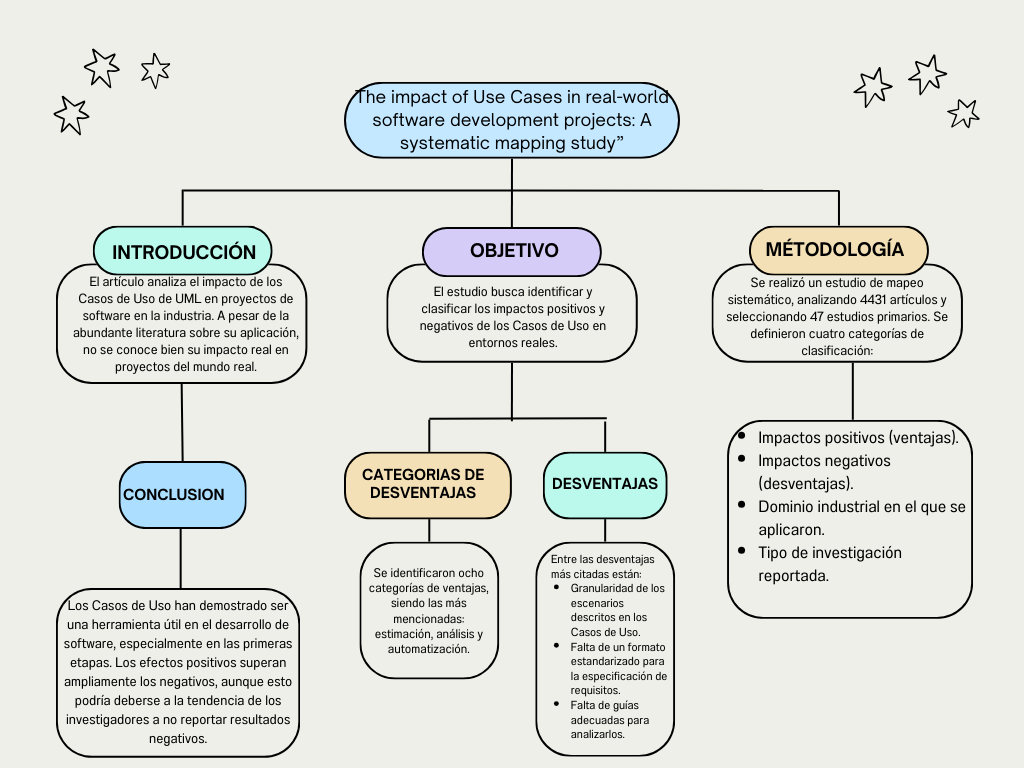
UNIVERSIDAD MANUELA BELTRAN

2025-1

**QUIDING QUESTIONS**

It refers to the structures of a system composed of elements with externally visible properties and  
relations between them. This definition is:  
1. UML  
2. Software Architecture  
3. View  
4. Controller  
It is a technical leader must know the principles related to software architecture, have extensive  
knowledge about technology and have excellent skills in written and oral communication. The previous  
profile refers to:  
1. developer  
2. Software architect  
3. Tester  
4. Analyst

**AUTONOMUS WORK ACTIVITY**

.

**PROCEDURE AND METODOLOGY OF PRACTICE**

**1. Product Backlog**

El backlog del producto contiene todas las funcionalidades necesarias para el desarrollo del ERP Escolar. Se ha priorizado utilizando dos métodos:

**1.1 Método MoSCoW**

* **Must Have** (Imprescindible): Funcionalidades críticas para el funcionamiento del sistema.
* **Should Have** (Importante): Características deseables pero no esenciales.
* **Could Have** (Opcionales): Mejoras que pueden agregarse en futuras versiones.
* **Won’t Have** (No necesarias por ahora): Características que no serán implementadas en esta fase.

**1.2 Priorización por Valor al Usuario**

Cada funcionalidad se ha ordenado según el impacto directo que tiene en la experiencia del usuario final.

**Lista de funcionalidades priorizadas**

1. **Gestión de Estudiantes**
   * Registro e inscripción de estudiantes (**Must Have**)
   * Seguimiento de asistencia y comportamiento (**Must Have**)
   * Historial académico y calificaciones (**Must Have**)
2. **Gestión de Docentes y Administrativos**
   * Registro y administración de profesores y personal (**Must Have**)
   * Asignación de materias y horarios (**Should Have**)
   * Control de permisos y asistencias (**Should Have**)
3. **Gestión Académica**
   * Creación y administración de materias y cursos (**Must Have**)
   * Asignación de horarios y grupos (**Must Have**)
   * Registro de notas y generación de boletines (**Must Have**)
4. **Gestión de Finanzas y Pagos**
   * Control de matrículas y mensualidades (**Must Have**)
   * Emisión de facturas y reportes financieros (**Must Have**)
   * Control de becas y descuentos (**Should Have**)
5. **Gestión de Recursos Humanos**
   * Administración de empleados y contratos (**Should Have**)
   * Nómina y pagos de salarios (**Should Have**)
   * Control de licencias y beneficios (**Could Have**)
6. **Gestión de Mantenimiento y Servicios**
   * Control de infraestructura y equipos (**Should Have**)
   * Registro de solicitudes de mantenimiento (**Could Have**)
   * Programación de servicios y reparaciones (**Could Have**)

**Definición de "Done"**

El equipo ha definido los siguientes criterios para considerar una historia de usuario como terminada:

* Código desarrollado y revisado por otro miembro del equipo.
* Pruebas unitarias implementadas y aprobadas.
* Código desplegado en el entorno de pruebas.
* Validación de funcionalidad de pruebas manuales.
* Documentación de la funcionalidad actualizada en el repositorio.

**Sprint Backlog - Primer Sprint**

**Historias de Usuario Seleccionadas y Especificación**

1. **Registro de Estudiantes**
   * **Prioridad:** Alta
   * **Descripción:** Como administrador, quiero registrar estudiantes en el sistema para gestionar su información académica.
   * **Criterios de aceptación:**
     1. Permitir el ingreso de datos personales (nombre, edad, documento de identidad, etc.).
     2. Validar la información ingresada.
     3. Guardar los datos en la base de datos.
     4. Mostrar confirmación del registro exitoso.
     5. Enviar un correo de bienvenida al estudiante registrado.
   * **Tareas:**
     1. Diseñar formulario de registro.
     2. Implementar validaciones en el backend.
     3. Crear API para guardar los datos.
     4. Diseñar interfaz de usuario.
     5. Implementar envío de correo electrónico.
2. **Asignación de Horarios y Cursos**
   * **Prioridad:** Media
   * **Descripción:** Como docente, quiero asignar horarios a mis cursos para gestionar mejor mis clases.
   * **Criterios de aceptación:**
     1. Seleccionar curso desde la interfaz.
     2. Permitir asignación de horarios por días y horas.
     3. Validar que no haya superposición de horarios.
     4. Guardar cambios en la base de datos.
     5. Notificar a los estudiantes inscritos en el curso.
   * **Tareas:**
     1. Diseñar interfaz para selección de cursos.
     2. Implementar lógica de validación de horarios.
     3. Crear API para actualizar datos en la base de datos.
     4. Implementar sistema de notificaciones.

**Estimación de Esfuerzo con Planning Poker**

Se ha utilizado la serie de Fibonacci para asignar la complejidad a cada historia de usuario.

|  |  |
| --- | --- |
| **Historia de Usuario** | **Complejidad** |
| Registro de Estudiantes | 5 |
| Asignación de Horarios y Cursos | 8 |

**5. Planificación de Pruebas**

Se han definido casos de prueba para validar la funcionalidad de cada historia de usuario.

**Casos de Prueba - Registro de Estudiantes**

1. **Validación de datos obligatorios**
   * **Descripción:** Verificar que todos los campos obligatorios sean requeridos antes de enviar el formulario.
   * **Resultado esperado:** No se permite el envío si falta información.
2. **Verificación de duplicados**
   * **Descripción:** Intentar registrar un estudiante con un documento de identidad ya existente.
   * **Resultado esperado:** Se muestra un mensaje de error.
3. **Prueba de formato de correo electrónico**
   * **Descripción:** Ingresar un correo electrónico con formato incorrecto.
   * **Resultado esperado:** Se muestra una alerta de error y no se permite el registro.
4. **Registro exitoso**
   * **Descripción:** Completar todos los campos correctamente y enviar el formulario.
   * **Resultado esperado:** El estudiante se registra correctamente y recibe un correo de bienvenida.
5. **Verificación en la base de datos**
   * **Descripción:** Revisar que los datos del estudiante se almacenen correctamente en la base de datos.
   * **Resultado esperado:** Los datos son persistidos correctamente.

**CALENDARIO DE DESARROLLO**

|  |  |  |
| --- | --- | --- |
| Día | Actividad | Responsable(s) |
| Día 1 | Reunión de planificación del Sprint | Miguel y Santiago |
| Día 2-3 | Configuración del entorno y base de datos | Miguel (backend), Santiago (base de datos) |
| Día 4-6 | Desarrollo del Módulo de Gestión de Estudiantes (CRUD y asistencia) | Miguel (CRUD estudiantes), Santiago (Registro de asistencia) |
| Día 7-8 | Desarrollo del Módulo de Gestión Académica (CRUD materias y calificaciones) | Miguel (CRUD materias), Santiago (Registro de calificaciones) |
| Día 9-10 | Desarrollo del Módulo de Gestión de Docentes y Administrativos (Registro y horarios) | Miguel (Registro docentes), Santiago (Asignación de horarios) |
| Día 11-12 | Pruebas unitarias y corrección de errores | Miguel y Santiago |
| Día 13-14 | Integración de módulos y pruebas finales | Miguel y Santiago |
| Día 15 | Revisión y retroalimentación | Miguel y Santiago |