

Imagen que contiene Texto

Descripción generada automáticamente

**Proyecto Final**

Sistema de administración de equipos

# 

|  |  |
| --- | --- |
| **Estudiantes** | **Firma** |
| **Araya Briceño Marcelo** | Imagen que contiene Texto  Descripción generada automáticamente |
| **ROJAS RODRIGUEZ NOE ROBERTO** |  |
| **ALPIZAR VIACHICA JOSHUA FABIAN** |  |
| **MEJIA BARRIENTOS DENIS JOSE** |  |

# Introducción

**Este programa es una aplicación web desarrollada con ASP.NET Core y Razor Pages, que gestiona la entrada y seguimiento de equipos de mantenimiento. El sistema se centra en registrar información detallada sobre cada equipo que ingresa, incluyendo detalles como la marca, modelo, el nombre del cliente, y el motivo de ingreso. Además, permite llevar un historial de cambios realizados a cada equipo, lo que es esencial para la trazabilidad y la gestión efectiva de la información de los equipos.**

# Objetivo del Programa

**El objetivo principal del programa es facilitar el registro y seguimiento de equipos en un entorno de mantenimiento o reparación. Este sistema está diseñado para:**

1. **Registrar Información de Equipos: Permite a los usuarios ingresar datos clave de cada equipo que ingresa al sistema, asegurando que se tenga un registro detallado y accesible.**
2. **Mantener un Historial de Cambios: A través de la tabla HistorialEquipos, el sistema puede mantener un registro de todos los cambios realizados a cada equipo, proporcionando una trazabilidad completa de las modificaciones y reparaciones realizadas.**
3. **Validar y Asegurar la Integridad de los Datos: Utilizando la validación del lado del servidor y las claves foráneas (en un escenario con integridad referencial), el sistema asegura que los datos ingresados sean válidos y consistentes, lo que reduce errores y mejora la calidad de la información almacenada.**

# Arquitectura en Capas del Proyecto

* **Descripción**: La arquitectura en capas es una forma de organizar un proyecto en diferentes niveles de responsabilidad, como presentación, lógica de negocio, y acceso a datos.
* **Cumplimiento**:
  + **APS.Web**: Se encarga de la capa de presentación (UI), manejando las vistas y los controladores.
  + **APS.Data**: Es la capa de acceso a datos que incluye el contexto de Entity Framework y las clases de modelo.
  + **APS.Security**: Podría representar la lógica de seguridad o lógica de negocio específica, dependiendo de cómo esté implementada.

# Uso de ActionResults, Endpoints, HTTP Requests

* **Descripción**: Los ActionResults se utilizan en los controladores para manejar las solicitudes HTTP y devolver respuestas apropiadas (como vistas o datos JSON).
* **Cumplimiento**:
  + En tus controladores, como EquiposController, utilizas ActionResults para manejar las solicitudes y devolver vistas o redirecciones.
  + Los endpoints en ASP.NET MVC están representados por las acciones dentro de los controladores que responden a las solicitudes HTTP.

# Creación de la BD, Uso de EntityFramework (ORM) con Patrón Repository

* **Descripción**: Entity Framework se utiliza como ORM (Object-Relational Mapping) para manejar la base de datos. El patrón Repository permite abstraer las operaciones de datos.
* **Cumplimiento**:
  + **EntityFramework**: Se utiliza para manejar la persistencia de datos en ApdatadbContext.
  + **Patrón Repository**: Aunque no se ha detallado explícitamente, si utilizas servicios que abstraen el acceso a la base de datos (como en EquipoService), esto cumple parcialmente con el patrón Repository.

# Utiliza Interfaces

* **Descripción**: Las interfaces se utilizan para definir contratos que deben cumplir las clases, permitiendo la implementación de diferentes versiones de una lógica.
* **Cumplimiento**:
  + Las interfaces como ISecurityService pueden estar siendo utilizadas para definir la lógica de seguridad, lo que permite la inyección de dependencias y pruebas unitarias.

# Evita Instanciaciones de Clases (Utiliza Patrón de DI)

* **Descripción**: El patrón de Inversión de Control (IoC) o Inyección de Dependencias (DI) evita la creación directa de instancias dentro de las clases, en lugar de inyectarlas a través del constructor.
* **Cumplimiento**:
  + En EquiposController, inyectas dependencias como ApdatadbContext y EquipoService en el constructor, lo cual es un ejemplo de Inyección de Dependencias.

# Flujo de Aprobación Ejecutado Exitosamente

* **Descripción**: Se refiere al proceso de verificar las aprobaciones y luego decidir si un equipo es aceptado o rechazado.
* **Cumplimiento**:
  + El flujo de aprobación se implementó exitosamente utilizando una vista que verifica los criterios antes de permitir el registro del equipo. Este flujo se maneja en el controlador EquiposController.

# Implementa SOLID Principles

* **Descripción**: SOLID son principios de diseño que ayudan a crear un código más mantenible y extensible.
* **Cumplimiento**:
  + **Single Responsibility**: Las clases parecen estar diseñadas para cumplir una única responsabilidad, como EquipoService para manejar la lógica de equipos.
  + **Dependency Inversion**: Se cumple mediante la inyección de dependencias.
  + **Otros principios** (como OCP, LSP, ISP) pueden estar implementados dependiendo de cómo se estructuren las interfaces y clases.

# Implementa Design Patterns

* **Descripción**: Los patrones de diseño como Singleton, Factory, Repository, etc., se utilizan para resolver problemas comunes en el desarrollo de software.
* **Cumplimiento**:
  + **Repository Pattern**: Aunque no se menciona explícitamente, si los servicios están abstraídos con interfaces, se cumple parcialmente.
  + **Dependency Injection**: Es un patrón de diseño que ya se está utilizando en tu proyecto.

# Creación de la UI de MVC

* **Descripción**: Se refiere a la implementación de la interfaz de usuario utilizando el patrón Model-View-Controller (MVC).
* **Cumplimiento**:
  + **Views**: Están implementadas utilizando Razor Pages para manejar la presentación.
  + **Controllers**: Manejan la lógica de flujo de la aplicación y los datos que se envían a las vistas.
  + **Models**: Definen los datos que se manejan y se muestran en las vistas.

# Link de GitHub:

En el siguiente Link, se encuentra el enlace al repositorio de GitHub que contiene el código fuente y los archivos relacionados como la base de datos del proyecto:  
<https://github.com/TheMarchelo/PALN_SC-601-Grupo-4>

Este repositorio incluye todas las implementaciones, configuraciones y documentación necesarias para entender y ejecutar el sistema de gestión de equipos desarrollado por el grupo.

Un pizarrón negro con letras blancas sobre fondo negro

Descripción generada automáticamente con confianza media