**GUÍA DE APRENDIZAJE**

**Análisis de Requisitos y Modelado de los Artefactos del Software**

**Proyecto: Desarrollo de Software Aplicando Tecnologías 4RI para Productores Agrícolas**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Lluvia de ideas — “El Poder del Análisis”**

**Pregunta 1**: ¿Qué pasa si se construye una casa sin planos?

* Si se construyera una casa sin planos, el resultado sería caótico porque no habría control sobre los materiales, los espacios quedarían mal distribuidos y podrían aparecer errores estructurales graves. En general se perdería tiempo, dinero y esfuerzo.

**Pregunta 2**: ¿Qué pasa si se desarrolla software sin análisis?

* Ocurre exactamente lo mismo, al avanzar sin analizar los requisitos, el software no resolverá las verdaderas necesidades del cliente, se duplicará trabajo, habrá fallas en la estructura y el resultado final será inútil o costoso de corregir.

**Conclusión**:

* El análisis es la base de todo buen software porque es el plano que guía el desarrollo, evita errores y garantiza que la solución cumpla lo que el usuario realmente necesita y sin análisis, no hay dirección ni calidad.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Foro técnico — “Tecnologías 4RI en el Agro”**

**Tecnología seleccionada: Internet de las Cosas (IoT)**

**Aplicación en la agricultura:**

El IoT permite instalar sensores en los cultivos para medir humedad del suelo, temperatura, pH y nivel de luz solar. Estos datos se envían a una aplicación web o móvil para que el agricultor sepa cuándo regar o fertilizar, optimizando el uso del agua y reduciendo costos.

**Ejemplo real:**

Un sensor de humedad conectado a una app que avisa cuando el terreno necesita riego, evitando desperdicio de agua.

**Comentario a compañeros (ejemplo porque no fue hecho en clase):**

Excelente aporte sobre IA. Sería interesante combinarla con IoT para que los sensores aprendan patrones y anticipen sequías.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**3. Taller “Del Problema al Modelo”**

**Plan de actividades (Scrum/Kanban)**

**Objetivo del proyecto:**

Desarrollar un sistema que ayude a agricultores a monitorear el estado de sus cultivos mediante sensores IoT.

**Backlog principal:**

1. Levantamiento de requisitos.
2. Diseño de casos de uso.
3. Creación del diagrama de clases inicial.
4. Diseño de base de datos.
5. Diseño de interfaz de usuario.

**Metodología**: Scrum (entregas cortas tipo sprint).

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Diagrama, Texto

El contenido generado por IA puede ser incorrecto.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Diagrama de Casos de Uso (descripción textual)**

**Actores:**

* Agricultor (usuario principal)
* Administrador del sistema
* Sensor IoT (dispositivo externo)

**Casos de uso:**

1. Iniciar sesión
2. Registrar cultivo
3. Ver estado de sensores
4. Recibir alertas de humedad
5. Generar reporte de rendimiento

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Diagrama de Clases inicial (descripción textual)**

**Clases principales:**

* Usuario: nombre, correo, contraseña
* Agricultor (hereda de Usuario): finca, ubicación
* Cultivo: nombre, tipo, fecha\_siembra, id\_sensor
* Sensor: id, tipo, valor, fecha\_lectura
* Reporte: id, fecha, promedio\_humedad, recomendación

**Relaciones:**

Un Agricultor tiene muchos Cultivos.

Un Cultivo tiene varios Sensores.

Un Reporte pertenece a un Cultivo.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4. Análisis del Informe y Definición Tecnológica**

**Caso común de necesidad**

Los agricultores del municipio necesitan controlar la humedad del suelo para evitar pérdidas de cultivo por exceso o falta de riego.

**Solución propuesta**

**Software**: Sistema web llamado AgroSmart, conectado a sensores IoT.

**Objetivo**: Mostrar en tiempo real la humedad y emitir alertas automáticas.

**Requisitos funcionales**

* RF1: Registrar usuarios (agricultores).
* RF2: Registrar cultivos y sensores asociados.
* RF3: Visualizar lecturas en tiempo real.
* RF4: Generar reportes históricos.
* RF5: Enviar alertas por bajo nivel de humedad.

**Requisitos no funcionales**

* RNF1: Disponibilidad 24/7.
* RNF2: Interfaz fácil de usar.
* RNF3: Seguridad en los datos.
* RNF4: Escalabilidad en la nube.

**Riesgos identificados**

* Fallos de red.
* Sensores dañados.
* Falta de capacitación del usuario.

**Propuesta tecnológica**

* Lenguaje: Python (Flask).
* Base de datos: MySQL.
* IoT: ESP32 con sensores DHT11.
* Plataforma: AWS (Cloud).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5. Taller UML — Diagrama de Clases Detallado**

**Clases**:

**Usuario**

* Atributos: id, nombre, correo, contraseña
* Métodos: iniciar\_sesion(), cerrar\_sesion()

**Agricultor (hereda Usuario)**

* Atributos: finca, telefono
* Métodos: registrar\_cultivo(), consultar\_reportes()

**Cultivo**

* Atributos: id, nombre, tipo, fecha\_siembra
* Métodos: calcular\_promedio\_humedad()

**Sensor**

* Atributos: id, tipo, valor, estado
* Métodos: enviar\_dato(), verificar\_estado()

**Reporte**

* Atributos: id, fecha, humedad\_promedio, recomendacion
* Métodos: generar\_pdf()

**Relaciones:**

Un Agricultor tiene muchos Cultivos.

Cada Cultivo tiene varios Sensores.

Cada Cultivo genera varios Reportes.

Diagrama, Texto

El contenido generado por IA puede ser incorrecto.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6. Foro — Aplicación de Patrones de Diseño**

**Escenario 1:**

Se necesita un solo acceso a la base de datos para evitar conflictos.

**Patrón aplicado**: Singleton

**Justificación**: Garantiza una única instancia de conexión a la base de datos en toda la aplicación, mejorando la eficiencia.

**Escenario 2:**

Se requiere crear distintos tipos de sensores (temperatura, humedad).

**Patrón aplicado**: Factory Method

**Justificación**: Centraliza la creación de objetos sin acoplar el código al tipo específico de sensor.

**Escenario 3:**

La interfaz del sistema es compleja y los agricultores se confunden.

**Patrón aplicado**: Facade

**Justificación**: Simplifica la interacción mostrando solo las funciones esenciales al usuario.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**7. Diseño de la Arquitectura del Software**

**Arquitectura seleccionada:** MVC (Modelo–Vista–Controlador)

**Justificación:**

Permite separar la lógica de negocio (Modelo), la interfaz (Vista) y el control (Controlador), facilitando mantenimiento y escalabilidad.

**Ventajas:**

* Separación clara de responsabilidades.
* Facilita pruebas unitarias.
* Escalable para futuras versiones.

**Desventajas:**

* Puede requerir más configuración inicial.

**Diagrama de Componentes (descripción):**

* Módulo Web (interfaz).
* Módulo IoT (lecturas de sensores).
* Módulo de Lógica (procesa datos).
* Base de datos MySQL.

**Diagrama de Despliegue (descripción):**

* Servidor web → Flask + API
* Servidor de BD → MySQL en AWS
* Dispositivos IoT → Sensores en campo
* Cliente → Navegador web del agricultor

Interfaz de usuario gráfica, Diagrama

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**8. Modelado y Normalización de Base de Datos**

**Tablas principales:**

* usuarios (id\_usuario, nombre, correo, contraseña)
* agricultores (id\_agricultor, finca, telefono, id\_usuario)
* cultivos (id\_cultivo, nombre, tipo, fecha\_siembra, id\_agricultor)
* sensores (id\_sensor, tipo, valor, estado, id\_cultivo)
* reportes (id\_reporte, fecha, humedad\_promedio, id\_cultivo)

**Relaciones:**

Un agricultor tiene muchos cultivos.

Un cultivo tiene varios sensores.

Un cultivo genera varios reportes.

**Normalización:**

* 1FN: Todos los datos son atómicos.
* 2FN: Cada campo depende completamente de la PK.
* 3FN: No hay dependencias transitivas.

**Diccionario de Datos (ejemplos):**

* cultivos.nombre → VARCHAR(50) → Nombre del cultivo
* sensores.valor → FLOAT → Dato leído por el sensor
* reportes.humedad\_promedio → FLOAT → Promedio diario de humedad

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**9. Prototipado de Interfaz de Usuario (UI)**

**Mapa de Navegación:**

**Inicio**

├── Iniciar sesión

├── Panel del agricultor

│ ├── Cultivos

│ │ ├── Ver sensores

│ │ └── Registrar cultivo

│ ├── Reportes

│ └── Perfil

└── Cerrar sesión

**Pantallas principales:**

1. Inicio de sesión: campos de usuario y contraseña.
2. Panel del agricultor: lista de cultivos y alertas.
3. Reporte de humedad: gráfico de humedad por día y botón “Exportar PDF”.

**Principios aplicados:**

* Botones grandes y legibles.
* Colores verdes y azules (relacionados con el campo).
* Textos simples y lenguaje amigable.

