

Curso AINC421

**Informe 2**

**Nombre integrante: Sebastián Cerda Fuentes**

**Fecha: 29-07-2025**

**Semana: 7**

Tabla de contenido

[Desarrollo y reflexión sobre el uso de UML para el diseño 2](#_Toc204711187)

[Propuesta de controles y evidencia a implementar en los flujos de la propuesta de solución 3](#_Toc204711188)

[Organización del desarrollo del producto: iteraciones o sprint (Agilidad) 4](#_Toc204711189)

[Matriz de trazabilidad de requerimientos (IL 2.2) 6](#_Toc204711190)

[Arquitectura de software a implementar (IL 2.3 y 2.4) 7](#_Toc204711191)

[Modelo de base de datos y/o clases 9](#_Toc204711192)

[Relaciones clave del modelo: 12](#_Toc204711193)

### ****Desarrollo y reflexión sobre el uso de UML para el diseño****

Para el desarrollo del sistema **ControlRepro**, se utilizó UML (Lenguaje Unificado de Modelado) como herramienta clave para representar gráficamente los distintos aspectos estructurales y funcionales del software. UML permite visualizar de manera clara la arquitectura del sistema, sus interacciones, flujos y componentes, lo que facilita la planificación, el análisis y la validación del diseño antes de la implementación.

Durante el proceso de diseño, se aplicaron principalmente tres tipos de diagramas UML:

* **Diagramas de casos de uso**, para identificar y documentar las funcionalidades clave desde la perspectiva del usuario (por ejemplo, iniciar sesión, registrar productos o generar pedidos).
* **Diagramas de clases**, que describen la estructura de los objetos del sistema, sus atributos y relaciones, como en el caso de las entidades Producto, Trabajador o Pedido.
* **Diagramas de secuencia**, que detallan cómo interactúan los componentes entre sí en el tiempo, mostrando claramente la lógica de flujos como el login, la validación de stock y la exportación de pedidos.

El uso de UML permitió anticipar posibles conflictos de lógica, facilitar la comunicación del diseño y asegurar una implementación alineada con los requerimientos. Además, contribuyó a organizar el desarrollo en sprints ágiles con mayor claridad, permitiendo mapear cada módulo con sus respectivos diagramas.

Esta metodología gráfica resultó ser una herramienta fundamental para transformar la idea inicial en un diseño robusto y validado, sirviendo tanto como guía de desarrollo como documento técnico de respaldo.

### ****Propuesta de controles y evidencia a implementar en los flujos de la propuesta de solución****

La solución ControlRepro considera un conjunto de controles internos y validaciones que permiten asegurar la calidad de los datos registrados, evitar errores humanos y garantizar la trazabilidad de cada acción. Estos controles están distribuidos en los distintos flujos del sistema, tanto en la etapa de ingreso de datos como en la generación automática de pedidos.

A continuación, se describen los controles principales y la evidencia generada en cada flujo:

#### 1. ****Ingreso al sistema (Login)****

* **Control:** Validación del RUT chileno en formato 12.345.678-9.
* **Evidencia:** Registro de sesión del trabajador (fecha, hora, RUT).
* **Validación adicional:** Comprobación del PIN correspondiente, almacenado localmente con cifrado simple.

#### 2. ****Registro de consumo de productos****

* **Control:** Solo se permite el ingreso de productos previamente definidos en la base de datos.
* **Evidencia:** Historial de movimientos con identificación del trabajador que realizó la acción.
* **Validación adicional:** El campo de búsqueda se bloquea tras autocompletar el producto correcto.

#### 3. ****Gestión del stock****

* **Control:** No se permiten valores negativos ni duplicación de productos en el mismo registro.
* **Evidencia:** Archivo temporal y registro interno en SQLite de cada operación de entrada o salida de productos.
* **Validación adicional:** Control visual con alertas en rojo si el stock actual queda por debajo del mínimo.

#### 4. ****Generación de pedidos sugeridos****

* **Control:** Reglas fijas de generación basadas en consumo histórico, stock mínimo y lead time por proveedor.
* **Evidencia:** Reporte de pedido en formato Excel o PDF, indicando responsable de la acción.
* **Validación adicional:** Se bloquea la generación si algún dato esencial está incompleto (producto sin proveedor o sin lead time).

#### 5. ****Exportación de pedidos****

* **Control:** Solo usuarios autenticados pueden generar exportaciones.
* **Evidencia:** Nombre del archivo contiene el RUT del trabajador y la fecha de generación. Se guarda en el escritorio como respaldo.

#### 6. ****Alertas por bajo stock****

* **Control:** Revisión automática de stock actual versus stock mínimo al cargar la aplicación o al registrar un nuevo consumo.
* **Evidencia:** Mensaje visible en pantalla (“Stock Crítico”) y registro en tabla de alertas internas.

### ****Organización del desarrollo del producto: iteraciones o sprint (Agilidad)****

El desarrollo de la aplicación **ControlRepro** se planificó siguiendo un enfoque **ágil**, específicamente utilizando la metodología **Scrum**, que permite organizar el trabajo en **sprints semanales** para facilitar la entrega incremental de funcionalidades y mejorar la adaptabilidad ante cambios.

A continuación, se detalla la organización del desarrollo mediante **cuatro sprints principales**, cada uno con sus objetivos, entregables y artefactos UML asociados:

#### ****Sprint 1: Estructura base y autenticación****

* **Objetivo:** Implementar el sistema de login con validación de RUT y PIN.
* **Tareas clave:** Diseño de pantalla de ingreso, validación de campos, conexión con base de datos local (SQLite).
* **Diagramas UML asociados:**
  + Caso de uso: “Iniciar sesión”
  + Diagrama de clases: Usuario
  + Diagrama de secuencia: Flujo de autenticación

#### ****Sprint 2: Registro y validación de productos****

* **Objetivo:** Permitir la búsqueda, visualización y validación de productos existentes.
* **Tareas clave:** Crear módulo de productos, cargar productos desde BD, bloquear campos tras selección válida.
* **Diagramas UML asociados:**
  + Caso de uso: “Registrar producto”
  + Diagrama de clases: Producto
  + Diagrama de secuencia: Validación de producto y bloqueo

#### ****Sprint 3: Módulo de consumo y control de stock****

* **Objetivo:** Habilitar el registro manual de consumos con control de stock mínimo.
* **Tareas clave:** Registrar entradas y salidas, mostrar alertas por bajo stock, vincular responsable.
* **Diagramas UML asociados:**
  + Caso de uso: “Registrar consumo”
  + Diagrama de clases: Movimiento
  + Diagrama de secuencia: Flujo de entrada/salida de stock

#### ****Sprint 4: Generación automática de pedidos y exportación****

* **Objetivo:** Implementar el motor de sugerencias y generar archivos Excel/PDF listos para enviar.
* **Tareas clave:** Calcular cantidades sugeridas según lead time y consumo histórico, incluir responsable en el archivo, exportar automáticamente.
* **Diagramas UML asociados:**
  + Caso de uso: “Generar pedido sugerido”
  + Diagrama de clases: Pedido, Proveedor
  + Diagrama de secuencia: Generación y exportación del pedido

Los diagramas mencionados se incluirán en el cuerpo del informe o en anexos, según su extensión. Esta estructura modular ha permitido validar funcionalidades en cada etapa, reducir errores acumulativos y obtener retroalimentación constante sobre el sistema.

### ****Matriz de trazabilidad de requerimientos (IL 2.2)****

La siguiente matriz permite evidenciar cómo cada requerimiento funcional identificado en el sistema **ControlRepro** se relaciona directamente con componentes específicos del diseño, los módulos funcionales implementados y los diagramas UML desarrollados. Esto garantiza una trazabilidad clara entre lo solicitado, lo diseñado y lo construido.

| **ID Requerimiento** | **Tipo** | **Descripción breve** | **Sprint / Iteración** | **Componente de diseño que lo aborda** |
| --- | --- | --- | --- | --- |
| RF-01 | Funcional | Validar RUT y PIN al iniciar sesión | Sprint 1 | Caso de uso: Iniciar sesión, Clase: Usuario |
| RF-02 | Funcional | Solo permitir registrar productos existentes en BD | Sprint 2 | Caso de uso: Registrar producto, Clase: Producto |
| RF-03 | Funcional | Registrar consumos de productos y controlar stock mínimo | Sprint 3 | Caso de uso: Registrar consumo, Clase: Movimiento |
| RF-04 | Funcional | Emitir alertas automáticas cuando el stock es crítico | Sprint 3 | Secuencia: Generación de alerta |
| RF-05 | Funcional | Generar pedidos sugeridos usando consumo histórico y lead time | Sprint 4 | Caso de uso: Generar pedido, Clase: Pedido, Proveedor |
| RF-06 | Funcional | Exportar pedidos en formato Excel/PDF | Sprint 4 | Diagrama de secuencia: Exportación |
| RNF-01 | No funcional | Aplicación debe funcionar sin conexión a internet | Todos | Arquitectura local, BD SQLite, lógica en frontend |
| RNF-02 | No funcional | Nombre del trabajador debe quedar registrado en cada acción | Sprint 1–4 | Registro de logs, inclusión en archivos exportados |
| RNF-03 | No funcional | Interfaz debe ser intuitiva, con controles visuales y alertas claras | Todos | Estilo visual con alertas, botones e íconos |
| RNF-04 | No funcional | El archivo exportado debe almacenarse automáticamente en el escritorio del usuario | Sprint 4 | Lógica de exportación con librería xlsx y fs |

Esta matriz permite asegurar que **todos los requerimientos están abordados** desde el diseño del sistema, facilitando el seguimiento y validación durante el desarrollo. También constituye una guía para futuras pruebas de aceptación y verificación funcional.

### ****Arquitectura de software a implementar (IL 2.3 y 2.4)****

La arquitectura seleccionada para el desarrollo del sistema **ControlRepro** es una **arquitectura en tres capas**, adecuada para aplicaciones de escritorio que requieren separación clara entre lógica de presentación, procesamiento y almacenamiento local. Esta arquitectura permite un desarrollo modular, facilita el mantenimiento futuro y asegura una mayor robustez en la manipulación de datos.

#### ****1. Capa de Presentación (Frontend)****

* **Tecnología utilizada:** Vue.js (integrado en Electron).
* **Responsabilidades:** Mostrar las interfaces al usuario final (login, gestión de productos, consumo, sugerencias de pedido, alertas).
* **Interacciones clave:**
  + Validación de campos como RUT y PIN.
  + Mostrar alertas visuales y formularios dinámicos.
  + Disparar eventos hacia la lógica de negocio.

#### ****2. Capa de Lógica (Backend local)****

* **Tecnología utilizada:** Node.js + funciones integradas vía preload y contextBridge de Electron.
* **Responsabilidades:**
  + Procesar la lógica de reposición.
  + Generar sugerencias de pedido.
  + Validar existencia y consistencia de datos.
  + Controlar permisos de exportación.
* **Flujos principales:**
  + Cálculo de cantidad a pedir en base al consumo histórico.
  + Validación de stock mínimo vs actual.
  + Control de lead time.

#### ****3. Capa de Datos (Persistencia local)****

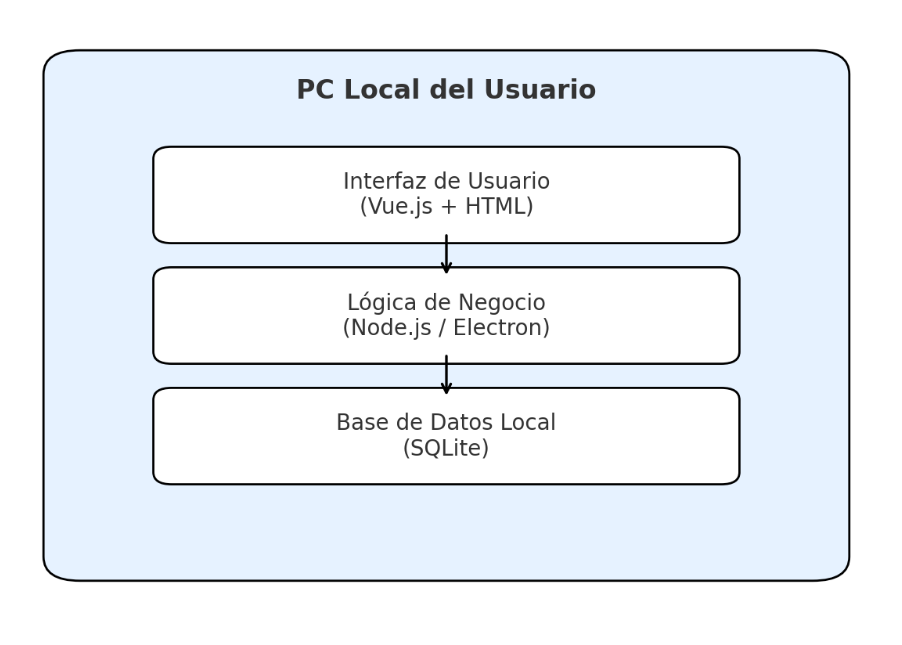
* **Tecnología utilizada:** SQLite.
* **Responsabilidades:**
  + Almacenamiento local de productos, proveedores, movimientos y trabajadores.
  + Mantener la trazabilidad de acciones por usuario.
* **Interacciones clave:**
  + Consultas y actualizaciones simples mediante SQL.
  + Tabla de logs internos para registrar consumos y pedidos generados.

#### ****Diagrama de Despliegue (resumen gráfico)****

Puedo generarlo como imagen si lo deseas. Representaría visualmente cómo interactúan las capas dentro del entorno de ejecución local de Electron (una única máquina, sin conexión a servidor externo).

Esta arquitectura se ajusta perfectamente al contexto del proyecto, ya que:

* No requiere conexión a internet.
* Asegura encapsulamiento y separación de responsabilidades.
* Permite construir una aplicación autónoma, ligera y mantenible.



La arquitectura de tres capas representada en el diagrama permite observar claramente cómo se distribuyen las responsabilidades dentro del sistema ControlRepro. Esta separación estructural no solo mejora el mantenimiento y la escalabilidad del software, sino que también facilita la validación de cada componente por separado. Dado que la aplicación funciona completamente de manera local, se asegura un alto grado de autonomía, privacidad de los datos y rapidez en la respuesta, aspectos fundamentales para su uso en PYMES sin conexión permanente a internet.

### ****Modelo de base de datos y/o clases****

El modelo de datos de ControlRepro fue diseñado con el objetivo de mantener una estructura clara, consistente y suficiente para representar las operaciones internas del sistema sin conexión a internet. Dado que la aplicación se basa en almacenamiento local mediante **SQLite**, se optó por un **modelo relacional simplificado**, priorizando velocidad de acceso, integridad referencial y facilidad de exportación.

A continuación, se describe la estructura de las principales tablas:

1. Tabla trabajadores



2. Tabla productos



3. Tabla proveedores



4. Tabla consumos



5. Tabla pedidos



### ****Relaciones clave del modelo:****

* Un **producto** pertenece a **un proveedor**.
* Un **consumo** está relacionado a un **producto** y un **trabajador**.
* Un **pedido** es generado por un **trabajador**, para un **producto**, y puede exportarse en diferentes formatos.

En resumen, el presente informe ha documentado de manera estructurada el diseño funcional y técnico de ControlRepro, abordando cada uno de los criterios establecidos por la rúbrica. Se han aplicado diagramas UML relevantes, estructurado el desarrollo en sprints, y definido una arquitectura escalable y autónoma, adecuada al contexto real de las PYMES. Este documento deja establecida la base técnica para la implementación completa del sistema.