Sala Comunitaria de Elaboración de Productos

Java y Aplicaciones Avanzadas sobre Internet

Grupo 17
Integrantes:
Ana Mariela Cossio Aquino 14349/4
Leandro David Svetlich 14474/8

UNIVERSIDAD
NACIONAL
DE LA PLATA

Introducción

Integración de Conocimientos en Contextos Reales

A lo largo de la cursada, se propone el desarrollo de un **proyecto final** con el objetivo de **integrar los conocimientos adquiridos** en cada clase, en un entorno de trabajo que puede asemejarse al real y cotidiano como desarrolladores profesionales.

La temática y el problema a resolver mediante el proyecto final se articula con la **Secretaría de Extensión de la Facultad** y busca **dar respuesta a problemáticas de organizaciones de la sociedad civil y/o instituciones de gestión pública**.

Este año, la institución designada para la realización del proyecto fue la **Sala comunitaria de elaboración de productos con agregado de valor de la Agricultura Familiar.**

Agregando Valor a la Agricultura Familiar

La "**Sala comunitaria de elaboración de productos con agregado de valor de la Agricultura Familiar**" es un área educativa y productiva ubicada en la **Facultad de Ciencias Veterinarias de la UNLP**, donde se elaboran principalmente **dulces y conservas** con el objetivo de **agregar valor a los alimentos** durante el proceso productivo y **contribuir a la economía de los trabajadores rurales** de la zona.

Surgió como una **iniciativa de la Universidad** para los sectores productivos de la región, siendo un punto de encuentro para la **colaboración y el intercambio de experiencias**. Aquí también se imparten **capacitaciones, charlas y talleres**. Después del proceso de producción, los alimentos se comercializan en **canales de venta locales**, principalmente en **ferias agroecológicas**.

Prosecretaría de AGRICULTURA FAMILIAR | Facultad de Ciencias VETERINARIAS | UNIVERSIDAD NACIONAL DE LA PLATA

Presentación de la Sala Comunitaria

El día **10/04/2024** durante el horario de práctica, se realizó la presentación de parte del equipo de la sala comunitaria.

Aquí conocimos sobre su **historia**, las **familias productoras** que la integran, el **personal** que compone la sala, las **actividades** como **cursos**, **talleres** y **capacitaciones** que se llevan a cabo, además de sus **objetivos** y sus **desafíos** diarios.

Una vez finalizada la presentación, tuvimos la oportunidad de hacer preguntas que nos ayuden a comprender mejor el problema, y conocer detalles más finos sobre el funcionamiento y lo que se esperaría de un posible software de gestión.

Desarrollo

Etapa 1: Análisis, diseño y bocetado

Conociendo los desafíos...

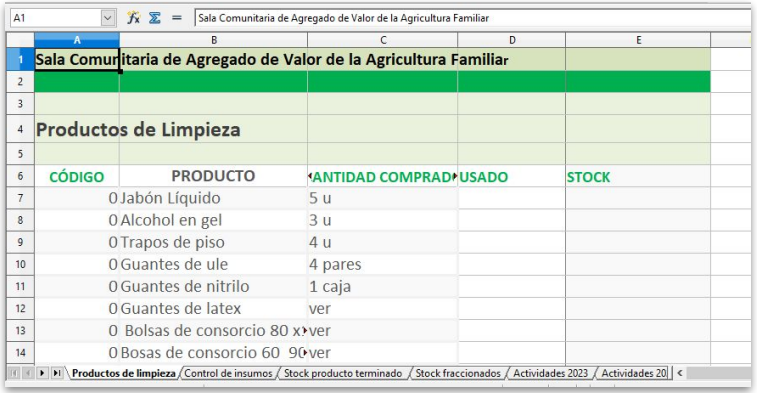
Hasta el momento, la administración se ha llevado a cabo mediante **hojas de cálculo**, pero esta metodología se está volviendo cada vez más laboriosa debido a la **variabilidad semanal** en los productos a elaborar y en la **variabilidad de los proveedores** de materias primas para cada lote de producción. Esta situación añade un nivel adicional de **complejidad** a la gestión de la sala, ya que se deben coordinar diversos aspectos, desde la **planificación de la producción** hasta la **gestión de compras** y **almacenamiento**.

Para abordar estos desafíos, se ha explorado la posibilidad de introducir nuevas **estrategias y herramientas de gestión**, incluyendo **software especializado** que permita un **seguimiento más preciso y automatizado** de los inventarios.

Esta tecnología podría facilitar la **toma de decisiones informadas** sobre la producción, la adquisición de materias primas y la **administración general del centro**, permitiendo anticiparse a posibles **faltantes o excesos de inventario**.

Análisis del problema

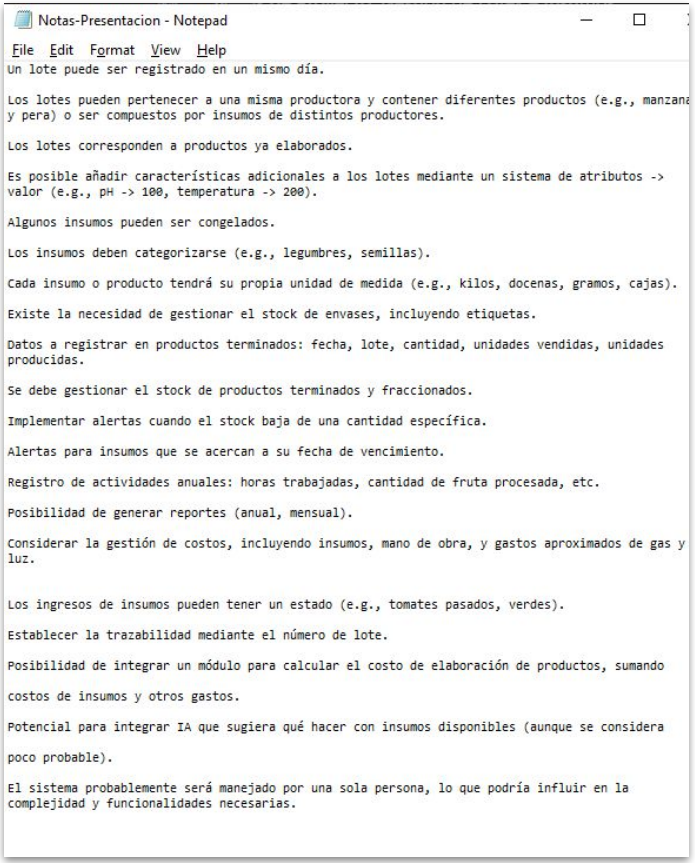
Durante la presentación realizada el **10/04/2024**, tomamos nota de diferentes aspectos que deberíamos tener en cuenta a la hora de pensar y diseñar una solución que realmente sea de utilidad. Adicionalmente, analizamos la **Hoja de Cálculo** que nos fue provista luego de la reunión, para conocer mejor cómo es el funcionamiento actual para la gestión.



Sala Comunitaria de Agregado de Valor de la Agricultura Familiar

	A	B	C	D	E
1	Sala Comunitaria de Agregado de Valor de la Agricultura Familiar				
2					
3					
4	Productos de Limpieza				
5					
6	CÓDIGO	PRODUCTO	CANTIDAD COMPRAD	USADO	STOCK
7	0	Jabón Líquido	5 u		
8	0	Alcohol en gel	3 u		
9	0	Trapos de piso	4 u		
10	0	Guantes de ule	4 pares		
11	0	Guantes de nitrilo	1 caja		
12	0	Guantes de latex	ver		
13	0	Bolsas de consorcio 80 x	ver		
14	0	Bosas de consorcio 60 90	ver		

Productos de limpieza / Control de insumos / Stock producto terminado / Stock fraccionados / Actividades 2023 / Actividades 20



Notas-Presentacion - Notepad

File Edit Format View Help

Un lote puede ser registrado en un mismo día.

Los lotes pueden pertenecer a una misma productora y contener diferentes productos (e.g., manzana y pera) o ser compuestos por insumos de distintos productores.

Los lotes corresponden a productos ya elaborados.

Es posible añadir características adicionales a los lotes mediante un sistema de atributos -> valor (e.g., pH -> 100, temperatura -> 200).

Algunos insumos pueden ser congelados.

Los insumos deben categorizarse (e.g., legumbres, semillas).

Cada insumo o producto tendrá su propia unidad de medida (e.g., kilos, docenas, gramos, cajas).

Existe la necesidad de gestionar el stock de envases, incluyendo etiquetas.

Datos a registrar en productos terminados: fecha, lote, cantidad, unidades vendidas, unidades producidas.

Se debe gestionar el stock de productos terminados y fraccionados.

Implementar alertas cuando el stock baja de una cantidad específica.

Alertas para insumos que se acercan a su fecha de vencimiento.

Registro de actividades anuales: horas trabajadas, cantidad de fruta procesada, etc.

Posibilidad de generar reportes (anual, mensual).

Considerar la gestión de costos, incluyendo insumos, mano de obra, y gastos aproximados de gas y luz.

Los ingresos de insumos pueden tener un estado (e.g., tomates pasados, verdes).

Establecer la trazabilidad mediante el número de lote.

Posibilidad de integrar un módulo para calcular el costo de elaboración de productos, sumando costos de insumos y otros gastos.

Potencial para integrar IA que sugiera qué hacer con insumos disponibles (aunque se considera poco probable).

El sistema probablemente será manejado por una sola persona, lo que podría influir en la complejidad y funcionalidades necesarias.

Buscando una mejor herramienta

Si bien las hojas de cálculo son herramientas poderosas, presentan algunos problemas a medida que aumenta el tamaño y la complejidad del sistema:

- **Escalabilidad Limitada**: Difíciles de manejar con grandes volúmenes de datos.
- **Errores Humanos**: Propensas a errores de entrada y fórmulas.
- **Falta de Automatización**: No automatizan procesos repetitivos.
- **Seguridad Débil**: Control limitado sobre el acceso a datos.
- **Capacidad de Análisis Reducida**: Sin herramientas avanzadas para informes complejos.
- **Necesidad de Capacitación**: Requieren formación extensa para su uso efectivo.

El sistema de gestión personalizado debería **facilitarle el trabajo a quien actualmente administra la sala**, así como también dar posibilidad a que **otros integrantes se involucren** en el registro de información, **sin comprometer la integridad de los datos.**

Requisitos clave para el software de gestión especializado

En base a la presentación, identificamos que a grandes rasgos, el software especializado de gestión a realizar debería considerar los siguientes aspectos:

- **Supervisión Integral de Inventarios**: Mejorar la gestión de insumos, materias primas y productos finales, reemplazando las hojas de cálculo por herramientas avanzadas para manejar la creciente complejidad.
- **Sistema de Seguimiento Semi-Automatizado**: Facilite el registro y la deducción automática de consumos de insumos y materias primas.
- **Sistema de Trazabilidad**: Asegurar la calidad y transparencia en la cadena de suministro, permitiendo identificar las materias primas utilizadas en cada elaboración y sus orígenes.
- **Registro de Entregas**: Llevar control de las entregas realizadas a las diferentes instituciones que se encargan de la comercialización de los productos elaborados.

Secciones de la aplicación

En detalle, la aplicación debe incluir las siguientes secciones:

- **Manejo de Sesión**: Permitir el ingreso y salida del sistema.
- **Roles de Usuario**: Definir responsabilidades para Administrador y Encargado de sala.
- **Gestión de Usuarios**: Controlar el acceso de otros actores al sistema.
- **Gestión de Materias Primas e Insumos**: Controlar el ingreso y consumo de materiales.
- **Gestión de Familias Productoras**: Asociar las materias primas a los productores.
- **Gestión de Recetas**: Establecer plantillas para las elaboraciones.
- **Gestión de Elaboraciones**: Registrar y ajustar inventarios según las cantidades elaboradas.
- **Gestión de Entregas**: Registrar la transferencia de productos a los puntos de venta.
- **Gestión de Puntos de Venta**: Asociar entregas con puntos de venta específicos.

Cada una de estas secciones está detallada en el documento original, formuladas como **Historias de Usuario**, que describen cómo debe interactuar el usuario con el sistema y los requisitos específicos para cada funcionalidad.

Historias de usuario y Maquetado preliminar

Una vez definidos los requisitos, procedemos a crear las **historias de usuario** que definen las necesidades y funcionalidades esperadas por cada sección desde el punto de vista del usuario. Adicionalmente realizamos **maquetas** que muestran cómo se verá cada sección de la aplicación una vez implementada.

Para el maquetado, utilizamos **Figma**, un potente software de diseño que facilita la creación de prototipos y maquetas visuales. Entre sus ventajas destacan la capacidad de **colaborar en tiempo real**, lo que mejora la comunicación y la retroalimentación, y la posibilidad de diseñar **componentes reutilizables** que aceleran el desarrollo y aseguran **uniformidad** en todos los diseños.

Al momento de realizar los diseños de interfaces, tratamos de prestar principal atención a los siguientes aspectos:

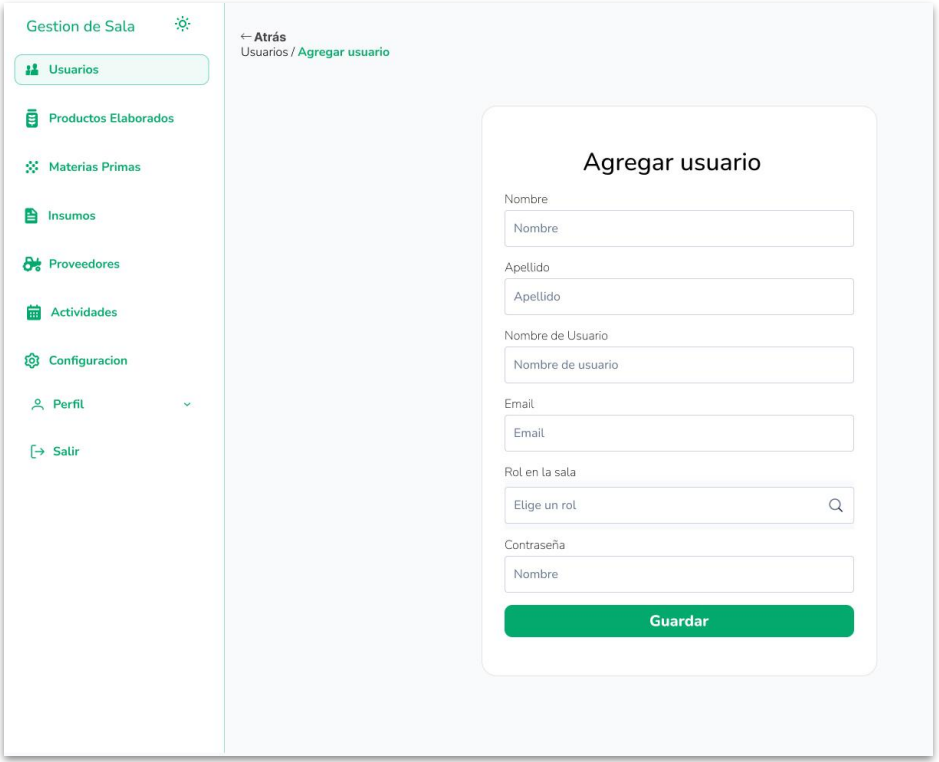
- **Consistencia**: lograr un diseño uniforme en todos los elemento. Esto incluye el uso coherente de colores, tipografías, iconos y estilos de botones, así como la disposición de los mismos.
- **Simplicidad**: mantener las interfaces limpias y sin desorden para que los usuarios puedan encontrar fácilmente la información y realizar acciones sin distracciones innecesarias.
- **Retroalimentación**: proporcionar respuestas claras a las acciones del usuario (mensajes de éxito, errores y estados de carga).

Para esta entrega realizamos un total de **18 maquetados**, a continuación mostramos algunos de ellos:

Historias de Usuario

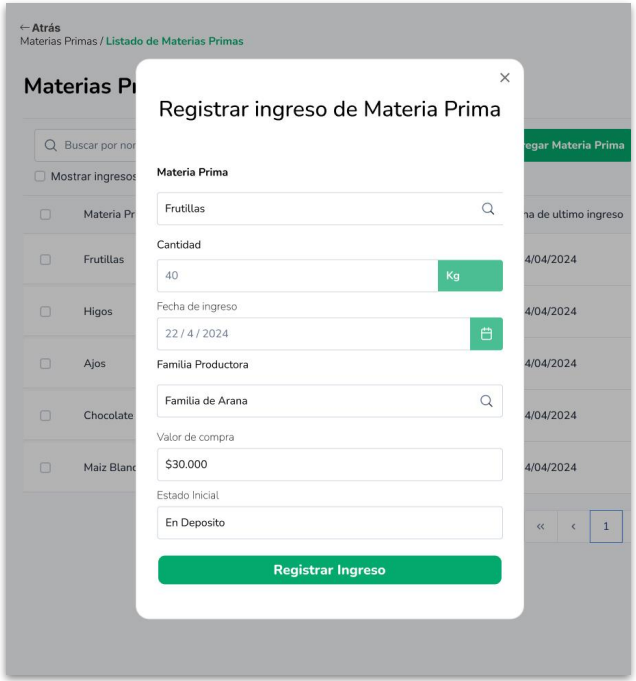
Alta de cuentas de usuarios

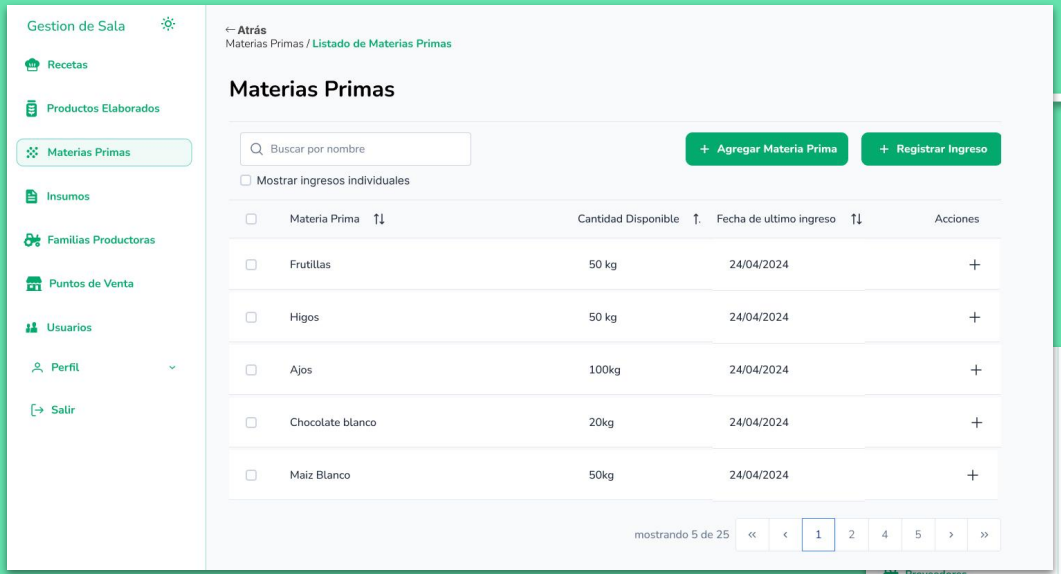
- **Descripción**: Como usuario administrador, quiero crear cuentas para que otros usuarios puedan ingresar al sistema, permitiéndoles visualizar/actualizar el stock, dependiendo de los roles que le sean otorgados.
- **Criterios de aceptación**:
  - Debo poder crear cuentas de manera sencilla y asignarles un rol específico (administrador o encargado de sala).
  - Al crear una cuenta de usuario invitado, debo poder ingresar los siguientes datos:
    - Email del usuario invitado
    - Nombre de usuario del usuario invitado.
    - Apellido del usuario invitado.
    - Nombre del usuario invitado.
    - Contraseña
    - Rol en la sala de producción (administrador, o encargado de sala).



Registrar ingreso de materia prima

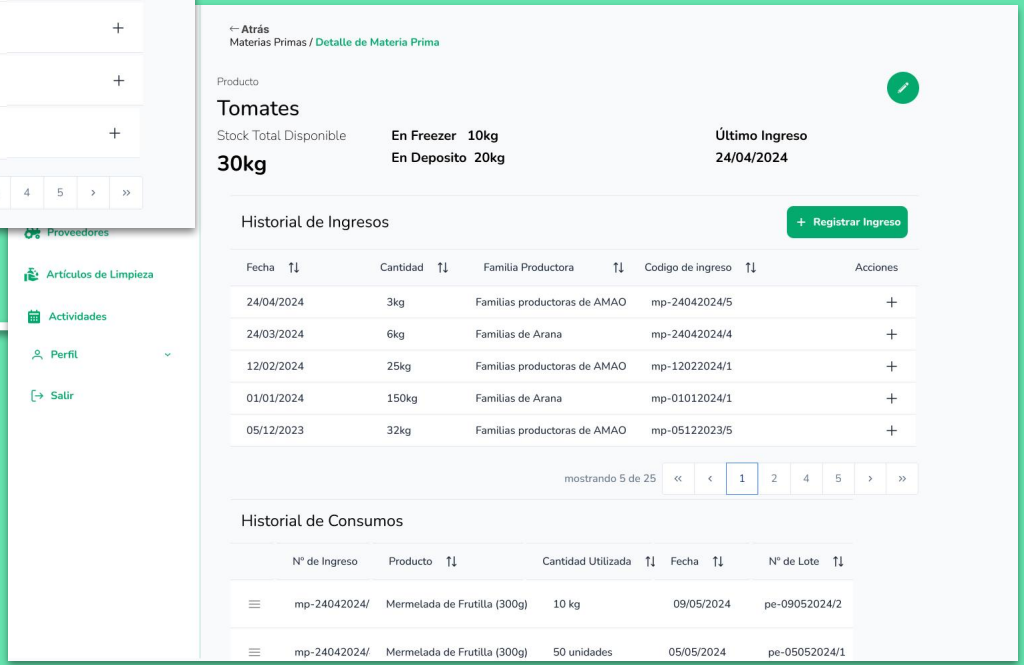
- **Descripción**: Como usuario administrador, necesito registrar el ingreso de nueva materia prima al almacén para mantener un seguimiento preciso de los recursos disponibles en la sala de producción.
- **Criterios de aceptación**:
    - El formulario debe incluir campos para ingresar la siguiente información:
        - **Seleccionar la materia prima**: Debe haber un menú desplegable para seleccionar la materia prima asociada al ingreso, dándose a elegir entre todas las materias primas registradas en el sistema.
        - **Seleccionar familia productora**: Debe haber un campo para seleccionar la familia productora de quien proviene la materia prima. Debe darse a elegir entre todas las familias productoras registradas en el sistema.
        - **Cantidad total ingresada**: Debe haber un campo para ingresar la cantidad de materia prima recibida.
        - **Valor de compra:** valor al cual se adquirió la materia prima.
        - **Registrar la fecha de ingreso**: Debe haber un campo para seleccionar la fecha de ingreso.
        - **Estado inicial (opcional):** Debe poder especificarse la forma de almacenamiento (en freezer, camara de frio, estante, etc). Estas opciones deben estar predefinidas de antemano. Debe quedar registro de la fecha en la que se estableció el lugar de almacenamiento, para luego tener un historial.
        - **Descripción (opcional):** Debe poder introducir información adicional que describa algún detalle importante sobre el ingreso.
    - Al agregarse debe auto-generarse un código de identificación legible, que luego sirva para reconocer dicho ingreso (ej: mp-09052024/1)

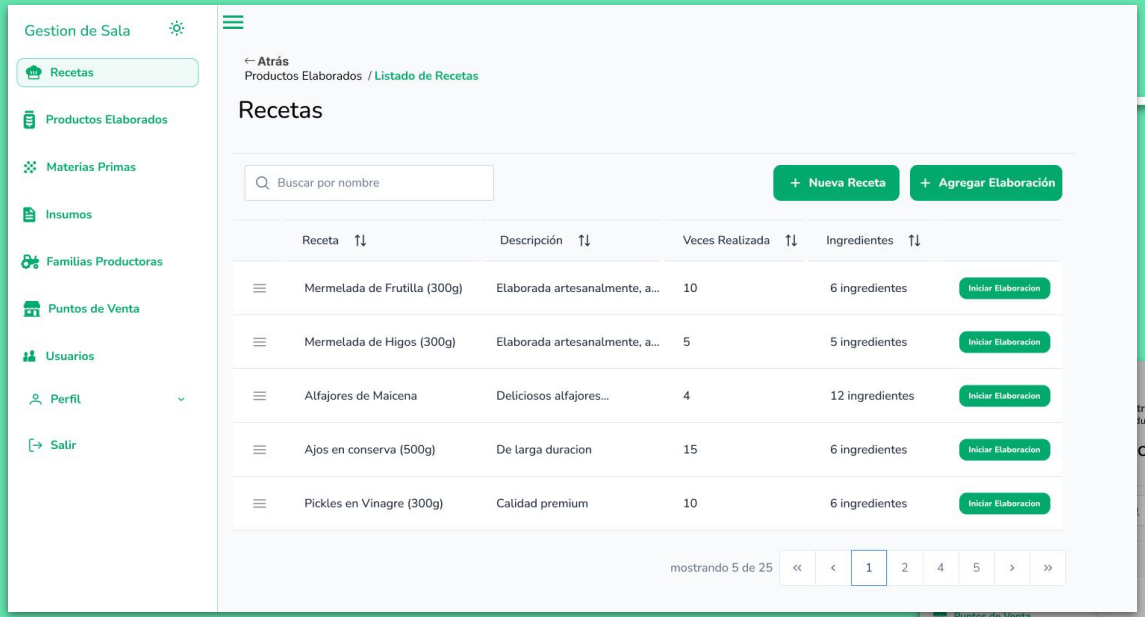




Listado de Materias Primas

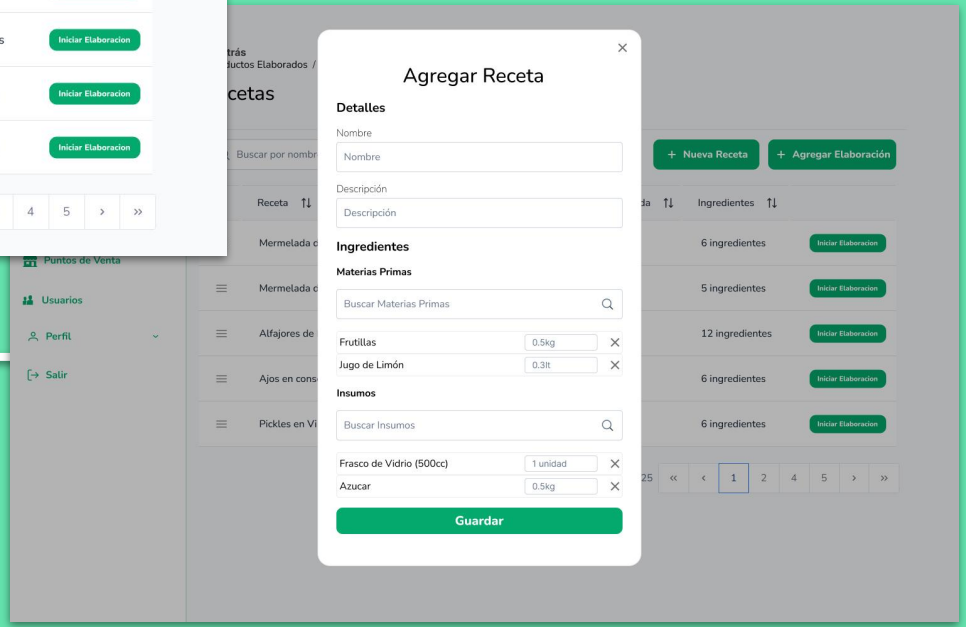
Detalle de Materia Prima



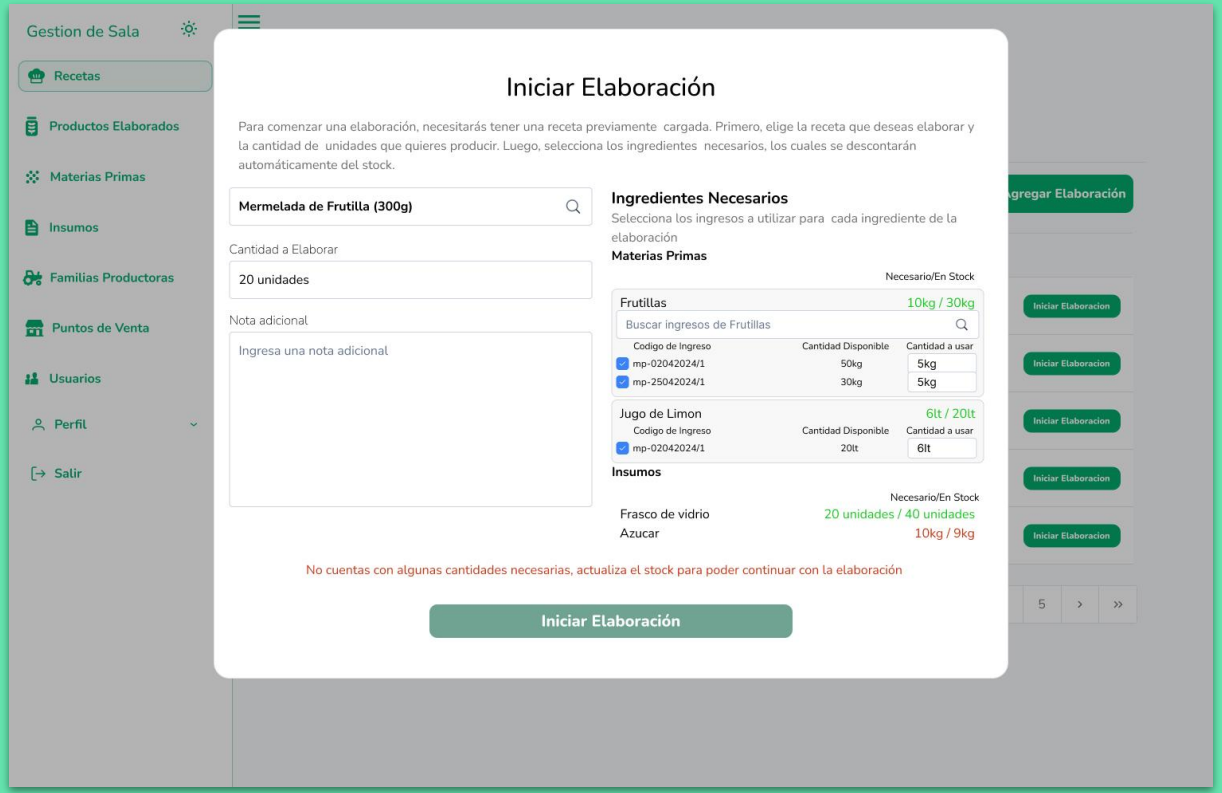


Listado de Recetas

Agregar Receta



Registrar Elaboración



Reflexiones sobre el maquetado

Algunos de los maquetados terminaron experimentando algunas modificaciones durante la implementación de la aplicación. Esto se realizó con el objetivo de **mejorar la visibilidad y navegación** entre las distintas secciones de la aplicación, **reducir la complejidad y la cantidad de elementos en pantalla**, y permitir que la aplicación se adapte a dispositivos móviles.

Además, algunas modificaciones se llevaron a cabo porque los **requerimientos cambiaron** en algunos casos, con el fin de ajustar el alcance del proyecto para que sea factible completarlo dentro del tiempo disponible de la cursada. A pesar de esto, la etapa de maquetado fue fundamental para **visualizar y planificar** la estructura de la aplicación, además de **identificar problemas** antes de comenzar con la implementación.

Desarrollo

Etapa 2:

- Definición de los objetos del modelo
- Control de versiones

Definiendo los objetos del modelo

En esta etapa, nos centramos en la **modelización de los objetos** para la capa de lógica de negocio del sistema, es decir, las reglas y procesos que definen cómo se manejan los datos y se toman decisiones dentro del software para cumplir con los objetivos del negocio.

Partimos del análisis y los maquetados acordados en la primera etapa para desarrollar una representación detallada de los objetos necesarios para la funcionalidad completa del sistema, la cual incluyó:

- **Diagrama de clases**
- **Archivos fuentes de las clases modeladas**

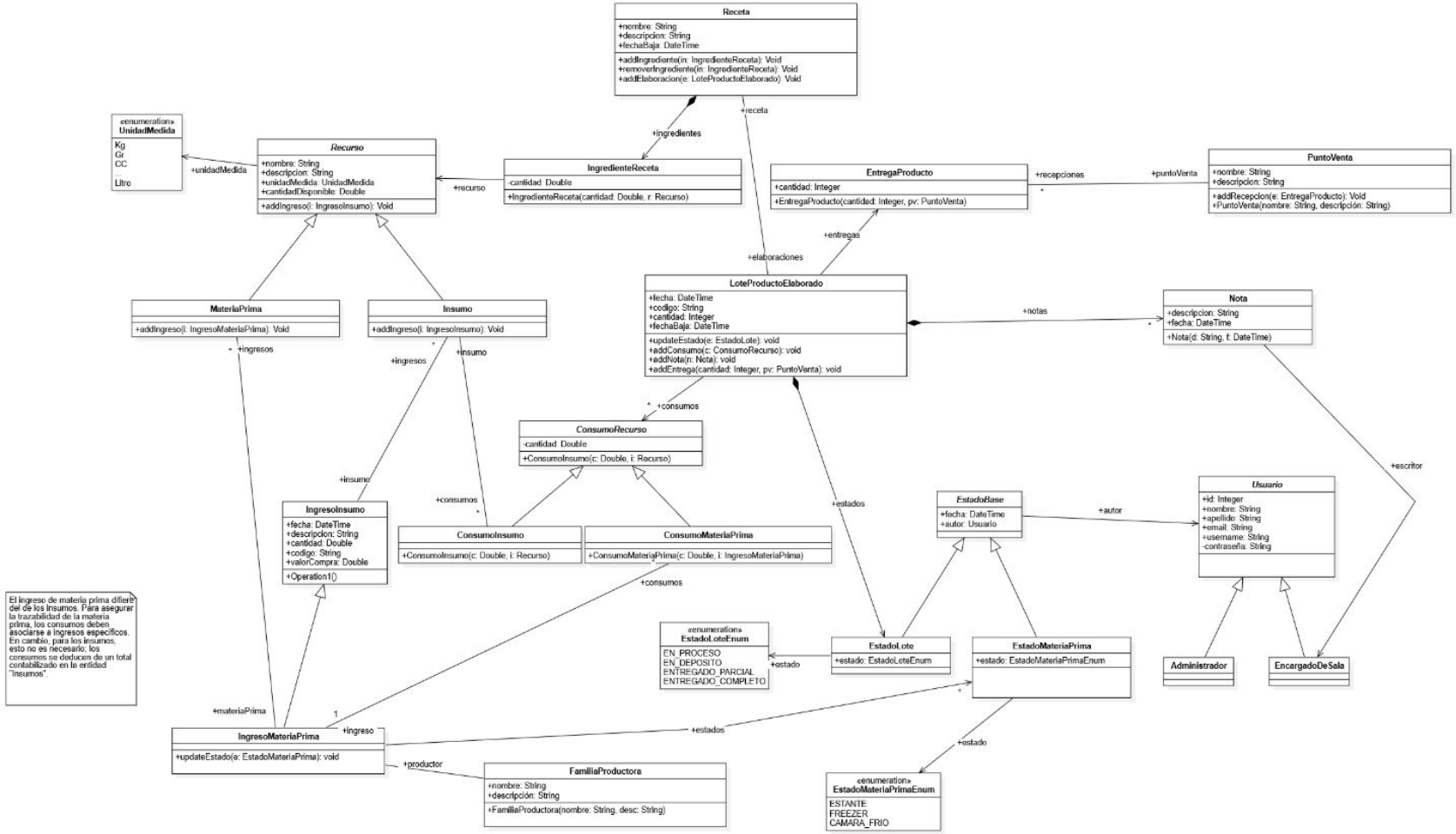
Diagrama de clases

Es una herramienta visual que muestra cómo cómo se organizan y se interrelacionan las partes principales de un programa de software. Imagina que un diagrama de clases es como un plano para construir una casa, donde se indican las habitaciones, sus funciones y cómo se conectan entre sí.

- **Clases**: Son como los planos de las habitaciones, describiendo qué propiedades (como tamaño y color) y qué acciones (cómo abrir una puerta) tiene cada parte del software.
- **Relaciones**: Muestran cómo las diferentes partes del software se conectan o se relacionan entre sí.

El desarrollo del diagrama incluyó varias versiones y modificaciones a medida que recibimos **retroalimentación** de los profesores y detectamos **posibles problemas y mejoras** en cada diseño.

Software utilizado para la realización del diagrama: StarUML™ The Open Source UML/MDA Platform


Receta
+nombre: String
+descripcion: String
+fechaBaja: DateTime
+addIngrediente(in: IngredienteReceta): Void
+removerIngrediente(in: IngredienteReceta): Void
+addElaboracion(e: LoteProductoElaborado): Void
+ingredientes
+receta
«enumeration»
UnidadMedida
Kg
Gr
CC
Litro
+unidadMedida
Recurso
+nombre: String
+descripcion: String
+unidadMedida: UnidadMedida
+cantidadDisponible: Double
+addIngreso(i: IngresoInsumo): Void
+recurso
IngredienteReceta
-cantidad: Double
+IngredienteReceta(cantidad: Double, r: Recurso)
EntregaProducto
+cantidad: Integer
+EntregaProducto(cantidad: Integer, pv: PuntoVenta)
+recepciones
+puntoVenta
PuntoVenta
+nombre: String
+descripcion: String
+addRecepcion(e: EntregaProducto): Void
+PuntoVenta(nombre: String, descripción: String)
+entregas
+elaboraciones
MateriaPrima
+addIngreso(i: IngresoMateriaPrima): Void
Insumo
+addIngreso(i: IngresoInsumo): Void
+ingresos
+ingresos
+insumo
LoteProductoElaborado
+fecha: DateTime
+codigo: String
+cantidad: Integer
+fechaBaja: DateTime
+updateEstado(e: EstadoLote): void
+addConsumo(c: ConsumoRecurso): void
+addNota(n: Nota): void
+addEntrega(cantidad: Integer, pv: PuntoVenta): void
+notas
Nota
+descripcion: String
+fecha: DateTime
+Nota(d: String, f: DateTime)
+consumos
ConsumoRecurso
-cantidad: Double
+ConsumoInsumo(c: Double, i: Recurso)
+escritor
+insumo
+consumos
+consumos
+estados
EstadoBase
+fecha: DateTime
+autor: Usuario
+autor
Usuario
+id: Integer
+nombre: String
+apellido: String
+email: String
+username: String
-contraseña: String
IngresoInsumo
+fecha: DateTime
+descripcion: String
+cantidad: Double
+codigo: String
+valorCompra: Double
+Operation1()
ConsumoInsumo
+ConsumoInsumo(c: Double, i: Recurso)
ConsumoMateriaPrima
+ConsumoMateriaPrima(c: Double, i: IngresoMateriaPrima)
+consumos
El ingreso de materia prima difiere del de los Insumos. Para asegurar la trazabilidad de la materia prima, los consumos deben asociarse a ingresos específicos. En cambio, para los insumos, esto no es necesario; los consumos se deducen de un total contabilizado en la entidad "Insumos".
«enumeration»
EstadoLoteEnum
EN_PROCESO
EN_DEPOSITO
ENTREGADO_PARCIAL
ENTREGADO_COMPLETO
+estado
EstadoLote
+estado: EstadoLoteEnum
EstadoMateriaPrima
+estado: EstadoMateriaPrimaEnum
Administrador
EncargadoDeSala
+materiaPrima
1
+ingreso
+estados
IngresoMateriaPrima
+updateEstado(e: EstadoMateriaPrima): void
+productor
FamiliaProductora
+nombre: String
+descripción: String
+FamiliaProductora(nombre: String, desc: String)
+estado
«enumeration»
EstadoMateriaPrimaEnum
ESTANTE
FREEZER
CAMARA_FRIO

Entidades del sistema

En limpio, los modelos/entidades fundamentales del sistema son los siguientes:

- **Usuario**
- **Receta**
- **Elaboracion**
- **Familia Productora**
- **Materia Prima**
- **Insumo**
- **Recurso**
- **Consumo Insumo**
- **Consumo Materia Prima**
- **Consumo Recurso**
- **Estado Elaboración**
- **Estado Materia Prima**
- **Nota**
- **Punto Venta**
- **Ingrediente Receta**
- **Ingreso Insumo**
- **Ingreso Materia Prima**
- **Entrega Producto**

Archivos fuentes de la clases modeladas

Los **archivos fuentes de las clases** son archivos de código que definen las clases que diagramamos y sus características en un lenguaje de programación específico, en este caso Java. Son esenciales para transformar el diseño en código funcional, proporcionando una guía clara y estructurada para la implementación del sistema.

Cada archivo fuente contiene:

- **Definición de Clases**: Estructura básica de la clase, incluyendo su nombre y propiedades.
- **Atributos**: Variables que almacenan datos relevantes para la clase.
- **Métodos**: Funciones que definen el comportamiento y las acciones que la clase puede realizar.
- **Constructores**: Métodos especiales para crear y configurar instancias de la clase.

Para esta entrega se realizaron 24 archivos fuente.

Control de versiones

Es una práctica esencial en el desarrollo de software que permite **gestionar y rastrear los cambios realizados en el código fuente a lo largo del tiempo.**

Ventajas:

- **Seguridad y Recuperación**: Permite registrar todas las modificaciones hechas al código, facilitando la revisión y recuperación de versiones anteriores en caso de errores.
- **Colaboración**: Facilita el trabajo en equipo, permitiendo a varios desarrolladores trabajar en el mismo proyecto sin sobrescribir el trabajo de los demás.
- **Resolución de Conflictos**: Ayuda a gestionar y resolver conflictos que surgen cuando múltiples desarrolladores modifican el mismo archivo.
- **Documentación**: Proporciona un registro claro de por qué se hicieron ciertos cambios, lo que ayuda en la toma de decisiones.

En esta etapa de la cursada comenzamos a utilizar **Git** (sistema de control de versiones) y **GitLab** (plataforma de gestión de repositorios)

Desarrollo

Etapa 3: Desarrollo de la capa de persistencia

Importancia de la Capa de Persistencia

La **capa de persistencia** es una parte fundamental de una aplicación que se encarga de **gestionar cómo los datos son almacenados y recuperados** en una **base de datos**, asegurando que la información que maneja la aplicación pueda ser guardada de forma permanente, y luego recuperada para su uso cuando sea necesario.

Tecnologías utilizadas:

Hibernate: herramienta/framework de mapeo objeto-relacional (ORM) que simplifica la relación entre los objetos de la aplicación y las tablas de una base de datos. Permite mapear los atributos de las clases a columnas en la base de datos, facilitando la interacción, reduciendo la necesidad de escribir código repetitivo y mejorando el flujo de trabajo en el desarrollo.

- **DAO (Data Access Object)**: patrón de diseño utilizado para separar la lógica de acceso a datos de la lógica de negocio. Oculta los detalles específicos de implementación del almacenamiento de datos y proporciona una interfaz coherente para que la capa de negocio interactúe con los datos.

Desarrollo de la capa de persistencia

En esta entrega, implementamos la capa de persistencia para **todas las entidades desarrolladas en la segunda etapa.**

El desafío principal fue representar con precisión las relaciones entre las entidades y configurar adecuadamente Hibernate, para que se ajuste a nuestras necesidades y gestione la correcta inicialización de la base de datos.

Además, desarrollamos una serie de **casos de prueba** para garantizar el correcto funcionamiento de la capa de persistencia y la adecuada gestión de las relaciones entre las entidades, mostrando los resultados de cada prueba en una interfaz web sencilla, que permite seguir el progreso en tiempo real.

Usuarios

Se creara un nuevo usuario Administrador:

{"id":"1", "apellido":"Min", "email":"admin@test.com", "password":"asdasd123", "username":"admin", "nombre":"Ad"}

↳ Se creara un nuevo usuario Encargado de Sala:

{"id":"2", "apellido":"Sala", "email":"encargado@test.com", "password":"asdasd123", "username":"encargadosala", "nombre":"Encargado"}

Listado de Usuarios:

{"id":"2", "apellido":"Sala", "email":"encargado@test.com", "password":"asdasd123", "username":"encargadosala", "nombre":"Encargado"}
{"id":"1", "apellido":"Min", "email":"admin@test.com", "password":"asdasd123", "username":"admin", "nombre":"Ad"}

Se modificará el usuario con id: 1

Obtener Usuario con id 1:

{"id":"1", "apellido":"Min (modificado)", "email":"admin@test.com", "password":"asdasd123", "username":"admin", "nombre":"Ad (modificado)"}

Listado de Usuarios:

{"id":"2", "apellido":"Sala", "email":"encargado@test.com", "password":"asdasd123", "username":"encargadosala", "nombre":"Encargado"}
{"id":"1", "apellido":"Min (modificado)", "email":"admin@test.com", "password":"asdasd123", "username":"admin", "nombre":"Ad (modificado)"}

Familia Productora

Se creará una nueva familia:

{"id":"1", "descripcion":"Gran quinta", "nombre":"Quinta de tomates"}

Se creará otra nueva familia:

{"id":"2", "descripcion":"Rica miel", "nombre":"Panales del sur"}

Listado de Familias Productoras:

{"id":"2", "descripcion":"Rica miel", "nombre":"Panales del sur"}
{"id":"1", "descripcion":"Gran quinta", "nombre":"Quinta de tomates"}

Se modificará la Familia Productora con id 1

Obtener Familia Productora con id 1:

{"id":"1", "descripcion":"Proucen tomate libre de agroquimicos", "nombre":"Quinta de tomates ARANA"}

Punto de Venta

Se creara un nuevo Punto de Venta:

{"nombre":"Punto 1", "descripcion":"Descripcion Punto 1", "recepciones":[], "id":"1"}

Listado de Puntos de Venta:

{"nombre":"Punto 1", "descripcion":"Descripcion Punto 1", "recepciones":[], "id":"1"}

Se modificará el Punto de Venta con id 1

Obtener Punto de Venta con id 1:

Insumo

Se crearán varios insumos:

{"id":"1", "descripcion":"Azucar Rubia ZZZZ", "unidadMedida":"KG", "cantidadDisponible":"120.0", "nombre":"Azucar Rubia ZZZZ", "ingresos":[]}
{"id":"2", "descripcion":"", "unidadMedida":"UNIDAD", "cantidadDisponible":"1000.0", "nombre":"Frascos", "ingresos":[]}
{"id":"3", "descripcion":"", "unidadMedida":"KG", "cantidadDisponible":"1000.0", "nombre":"Gelificante", "ingresos":[]}

Listado de Insumos:

{"id":"3", "descripcion":"", "unidadMedida":"KG", "cantidadDisponible":"1000.0", "nombre":"Gelificante", "ingresos":[]}
{"id":"2", "descripcion":"", "unidadMedida":"UNIDAD", "cantidadDisponible":"1000.0", "nombre":"Frascos", "ingresos":[]}
{"id":"1", "descripcion":"Azucar Rubia ZZZZ", "unidadMedida":"KG", "cantidadDisponible":"120.0", "nombre":"Azucar Rubia ZZZZ", "ingresos":[]}

Se modificará el insumo con id 1

{"id":"1", "descripcion":"Azucar Rubia", "unidadMedida":"KG", "cantidadDisponible":"120.0", "nombre":"Azucar", "ingresos":[]}

Obtener el insumo con el id 1:

{"id":"1", "descripcion":"Azucar Rubia", "unidadMedida":"KG", "cantidadDisponible":"120.0", "nombre":"Azucar", "ingresos":[]}

Ingreso de Insumo

Se creará un nuevo Ingreso de Insumo (Azucar):

{"id":"1","fecha":Thu Aug 08 19:01:06 ART 2024, "descripcion":"Ingreso de Azucar", "cantidad":"5.5", "codigo":"I-04052023", "valorCompra":"20000.0", "insumoId":1}

Listado de Ingresos de Insumos:

{"id":"1","fecha":Thu Aug 08 19:01:06 ART 2024, "descripcion":"Ingreso de Azucar", "cantidad":"5.5", "codigo":"I-04052023", "valorCompra":"20000.0", "insumoId":1}

Se modificará el Ingreso de Insumo con id 1

Obtener Ingreso de Insumo con id 1

{"id":"1","fecha":Thu Aug 08 19:01:06 ART 2024, "descripcion":"Ingreso de Azucar(modificado)", "cantidad":"7.0", "codigo":"I-06052023", "valorCompra":"22000.0", "insumoId":1}

Materia Prima

↳ Se crearan dos Materias Primas:

Materia Prima{id='1', nombre='Tomattte', cantidadDisponible='60.0', descripcion='Tomates para elssssssaboración de salsas', unidadMedida=KG}
Materia Prima{id='2', nombre='Miel de abeja', cantidadDisponible='100.0', descripcion='Miel fresca de abejas locales', unidadMedida=KG}

Listado de MateriaPrimas:

Materia Prima{id='2', nombre='Miel de abeja', cantidadDisponible='100.0', descripcion='Miel fresca de abejas locales', unidadMedida=KG}
Materia Prima{id='1', nombre='Tomattte', cantidadDisponible='60.0', descripcion='Tomates para elssssssaboración de salsas', unidadMedida=KG}

↳ Se modificará la Materia Prima con id 1

Obtener el detalle de Materia Prima con el id 1:

Materia Prima{id='1', nombre='Tomate', cantidadDisponible='60.0', descripcion='Tomates para elaboración de salsas', unidadMedida=KG}

Ingreso de Materia Prima

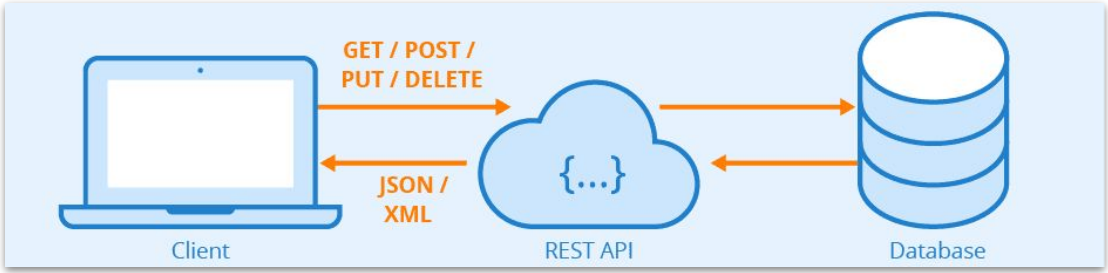
Desarrollo

Etapa 4: Desarrollo de servicios REST

REST (Representational State Transfer)

Es un estilo arquitectural para diseñar Servicios Web. Se basa en el uso del protocolo HTTP en conjunto con una serie de lineamientos de organización:

- **Recursos direccionables**: todo lo que se puede manipular o consultar se considera un recurso
- **Interfaz uniforme y acotada**: utiliza métodos HTTP estándar para realizar operaciones comunes sobre los recursos.
- **Orientado a representación**: trabaja con representaciones bien definidas de los recursos (**JSON**, XML, HTML)
- **Protocolo de comunicación sin estado**: cada solicitud es independiente y no depende de solicitudes anteriores.



Adoptando REST en el proyecto

Nos apoyamos en algunas tecnologías:

JAX-RS: Framework que facilita la parte comunicacional en la implementación de servicios RESTful mediante el uso de anotaciones.

Jackson: Librería que provee conversión automática entre JSON y clases Java de manera directa.

HK2: Framework para la inyección de dependencias, que ayuda a organizar mejor el código y mantener una separación clara entre la lógica de negocio y otras capas de la aplicación.

Swagger y OpenAPI: Utilizadas en conjunto, permiten documentar y probar APIs RESTful de manera interactiva, proporcionando una interfaz gráfica para explorar y entender las rutas y modelos de datos.

Ejemplo: Controlador de Elaboraciones

Define **/elaboraciones** como ruta base para operaciones de elaboraciones.

Define el **título** que veremos en la interfaz de Swagger para esta sección

Inyectamos los **servicios** que contienen la **lógica de negocio**

Devuelve **todas las elaboraciones** para una receta.

Devuelve el **detalle de una elaboración en concreto**.

Crea una nueva elaboración en el sistema.

```java
@Path("/elaboraciones")  no usages
@Tag(name = "Elaboraciones")
public class ElaboracionResource extends BaseResource {

    @Inject  4 usages
    private IElaboracionService elaboracionService;

    @Inject  no usages
    private IIngredienteRecetaService ingredienteRecetaService;

    @GET
    public List<ElaboracionViewModel> getAll(@PathParam("recetaId") Long recetaId) {
        return elaboracionService.getAll(recetaId);
    }

    @Path("{id}")  no usages
    @GET
    public ElaboracionDetalleViewModel getDetail(@PathParam("id") Long id) {
        return elaboracionService.getById(id);
    }

    @POST
    public ElaboracionDetalleViewModel create(ElaboracionCreateViewModel viewModel) {
        return elaboracionService.create(this.getUsuarioId(), viewModel);
    }
```

View Models

Son **objetos de datos simples** que representan la información que se va a enviar o recibir a través de la API.

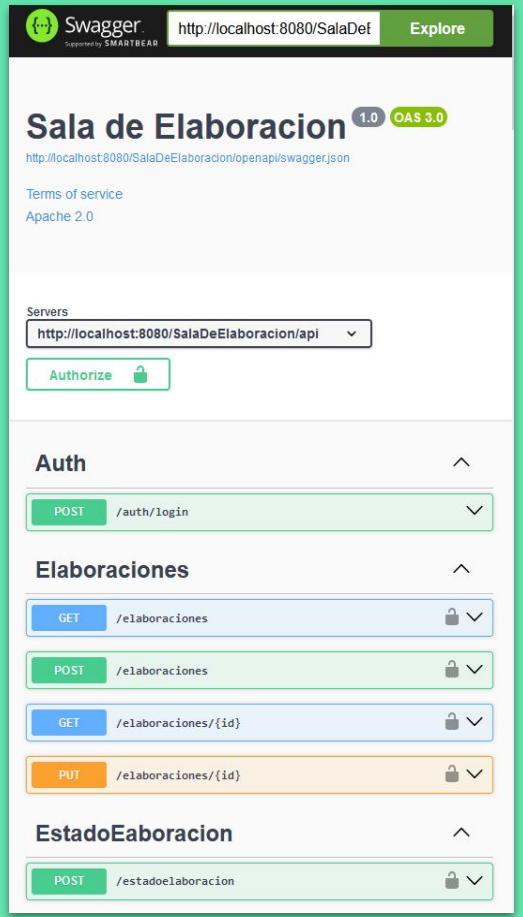
Se utilizan para lograr un **control más preciso** sobre la información que queremos exponer, tanto a la hora de recibir o enviar solicitudes, en contraposición a manejar directamente las entidades del dominio.

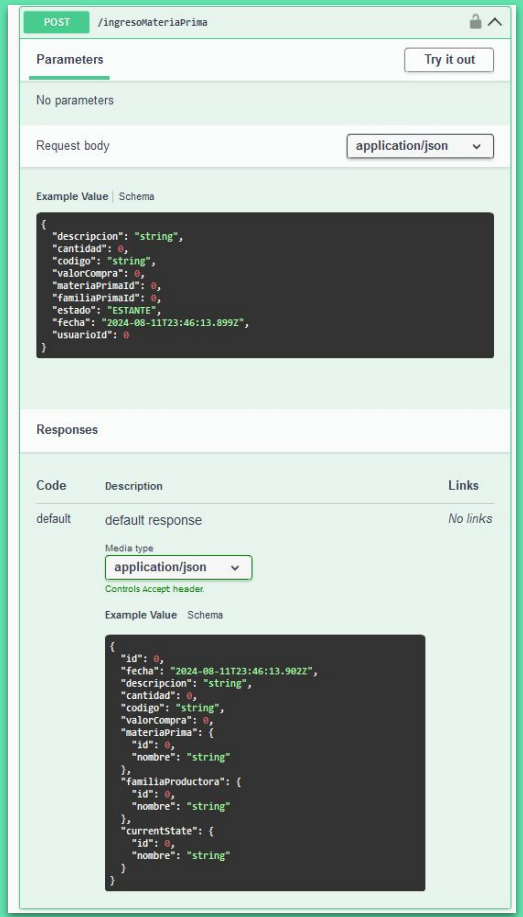
Algunas ventajas:

- **Mayor control**
- **Mejora de la Seguridad**
- **Consistencia en la comunicación**
- **Adaptabilidad**
- **Mejor Comunicación con la UI**

```
public class EstadoElaboracionCreateViewModel {  8 usages
    public EstadoElaboracionEnum estado;
    public Date fecha = new Date();
    public Long elaboracionId;
    public Long usuarioId;

    public EstadoElaboracionCreateViewModel() {  no usages
    }
}
```

Swagger generado automáticamente en base al proyecto





Desarrollo

Etapa 5: Desarrollo de la vista e integración con la capa de servicios

Desarrollo de la vista (UI)

En esta etapa desarrollamos la interfaz gráfica de la aplicación, para finalmente tener un primer prototipo funcional del proyecto que abarque todo lo desarrollado durante la cursada.

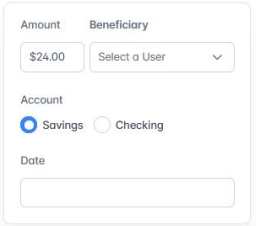
La principal tecnología utilizada fue **Angular**, una plataforma muy robusta para la construcción de aplicaciones clientes, que combina **HTML** y **TypeScript**, permitiendo crear **SPA**s muy interactivas y mantenibles.

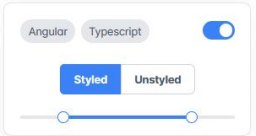
- **HTML**: componente básico que define el significado y la estructura del contenido de la web mediante un lenguaje de etiquetas.
- **TypeScript**: lenguaje de programación que extiende **JavaScript** al agregarle tipos estáticos, lo que ayuda a prevenir errores comunes durante el desarrollo.
- **SPA (Single Page Application)**: tipo de aplicación web que carga una única página y actualiza dinámicamente el contenido a medida que el usuario interactúa en lugar de cargar páginas nuevas desde el servidor para cada interacción, logrando una experiencia más fluida, similar a una aplicación de escritorio.

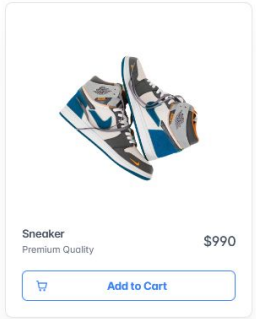
Aprovechando el Potencial de las Librerías

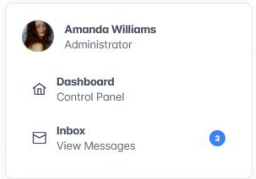
Una de las ventajas de la popularidad y trayectoria de Angular es su extenso **ecosistema de librerías**, muchas de las cuales son desarrolladas por organizaciones o individuos que buscan contribuir a la comunidad.

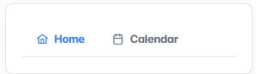
Para acelerar el desarrollo de la interfaz de usuario en nuestro proyecto, utilizamos **PrimeNG**, una biblioteca que ofrece una amplia variedad de componentes visuales, como botones, tablas, formularios, calendarios, menús y gráficos, entre otros. Esto nos permitió **enfocarnos más en la funcionalidad**, ya que no tuvimos que implementar desde cero cada pequeña parte de la aplicación, y al mismo tiempo logramos mantener una consistencia visual a través de los componentes que provee la librería.







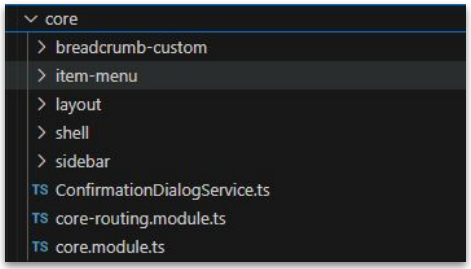


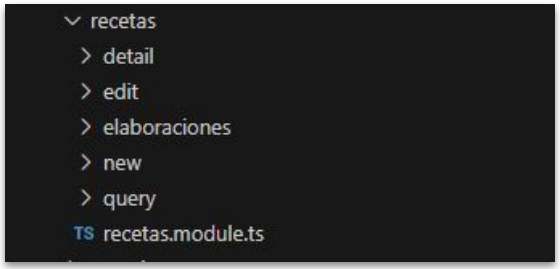


Estableciendo una Estructura de Carpetas

Para asegurar un desarrollo ordenado y evitar complicaciones a lo largo del proyecto, uno de los primeros pasos fue definir una **estructura de carpetas clara y funcional**. Esta organización permite mantener el código bien distribuido, minimizando la confusión, teniendo en cuenta que somos dos integrantes en el equipo.

Cada carpeta tiene una **responsabilidad específica**, separando los componentes **reutilizables** de aquellos diseñados para **funcionalidades principales** de la aplicación (features), **rutas**, **objetos del modelo** (interfaces), **servicios**, etc. Esto mejora mucho la experiencia al trabajar en equipo.





Comunicación con la API REST

Para gestionar la comunicación con la API, creamos **servicios dedicados** para cada entidad del sistema.

Estos servicios encapsulan la lógica necesaria para realizar **solicitudes HTTP al servidor**, utilizando **HttpClient**, un servicio propio de Angular diseñado para interactuar con servidores, el cual trabaja con **Observables**, lo que permite manejar las respuestas asíncronas de manera eficiente y alineada con la arquitectura reactiva de Angular.

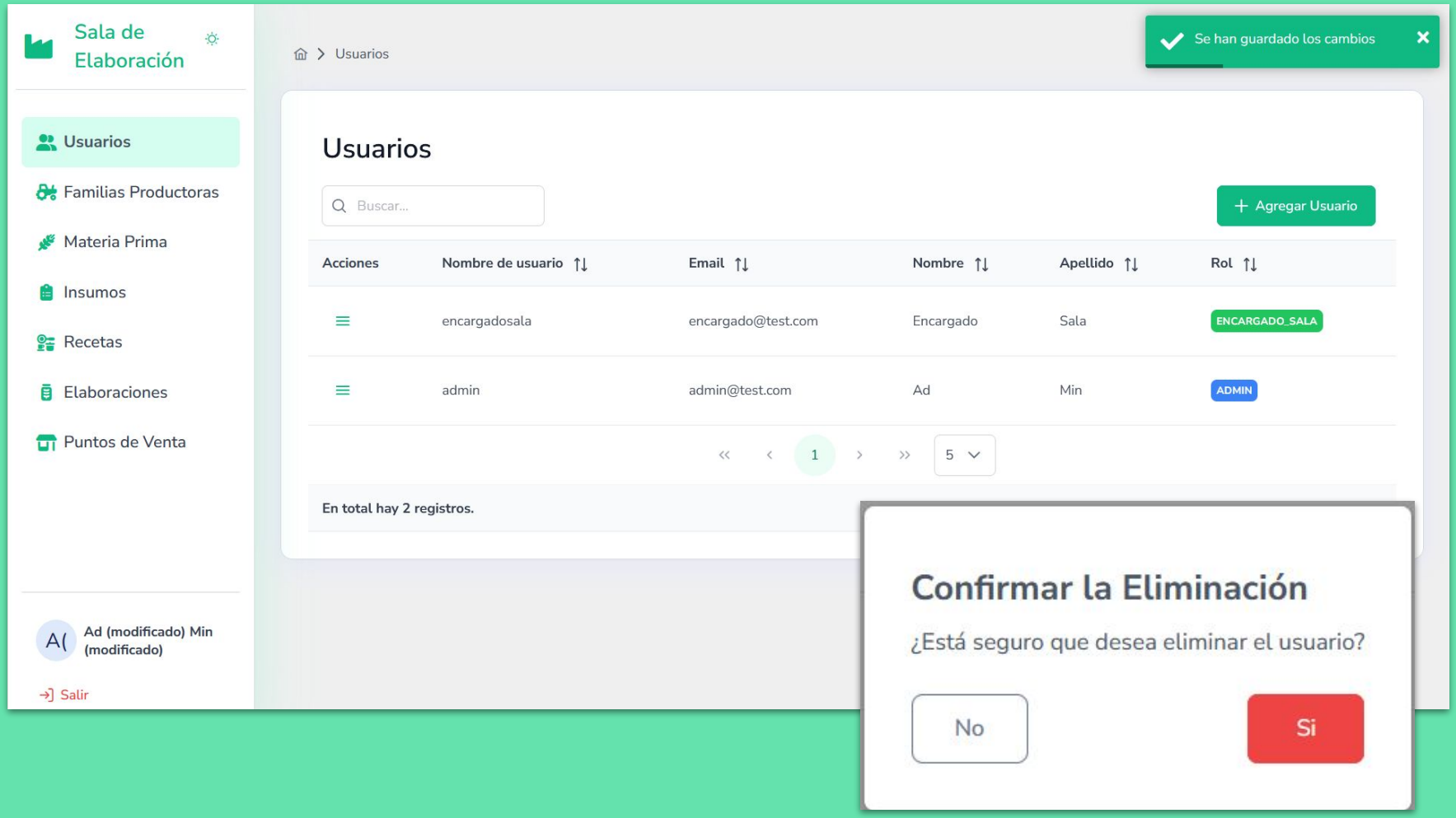
Cada ruta definida en estos servicios **corresponde a un endpoint** definido en la API. Además, hemos creado **interfaces** que representan los **ViewModels**, lo que proporciona **mayor seguridad tipográfica** y simplifica el manejo de las respuestas de la API.

```
@Injectable()
export class ElaboracionService {
  public API_URL = environment.API_URL + 'elaboraciones';

  constructor(private http: HttpClient) {}

  getAll(recetaId?:number) {
    return this.http.get<ElaboracionViewModel[]>(this.API_URL + (recetaId ? ("?re
  }

  detail(id: number) {
    return this.http.get<ElaboracionDetalleViewModel>(this.API_URL + '/' + id);
  }

  create(receta: ElaboracionCreateViewModel) {
    return this.http.post<ElaboracionViewModel>(this.API_URL, receta);
  }

  edit(id: number, receta: Partial<RecetaCreateViewModel>) {
    return this.http.put<RecetaDetalleViewModel>(
      this.API_URL + '/' + id,
      receta
    );
  }
```

Layout y Navegación

Una vez definida la estructura básica del proyecto, procedimos a crear el **layout** que serviría como la plantilla base para la aplicación, junto con componentes, directivas y servicios adicionales esenciales para el desarrollo de las funcionalidades.

- **Sidebar**: Facilita la navegación entre secciones de la aplicación, cargando el contenido de forma asíncrona para reducir el tamaño del bundle inicial y mejorar los tiempos de carga.
- **Breadcrumbs (migas de pan)**: Automáticos para mostrar la ubicación del usuario dentro de la estructura de navegación, facilitando la orientación y mejorando la experiencia de usuario.
- **Toast de notificaciones**: Permite mostrar notificaciones de éxito o error con solo llamar a una función desde cualquier parte de la aplicación.
- **Diálogos de confirmación**: Implementados para gestionar confirmaciones de acciones dentro de la app, también disponible llamando sólo a una función.

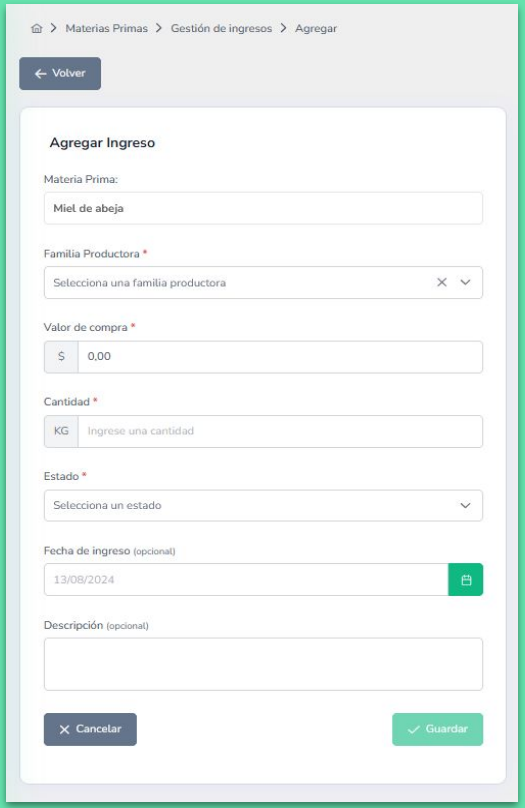

Sala de Elaboración
Usuarios
Familias Productoras
Materia Prima
Insumos
Recetas
Elaboraciones
Puntos de Venta
Ad (modificado) Min (modificado)
Salir
Usuarios
Se han guardado los cambios
Usuarios
Buscar...
Agregar Usuario
Acciones
Nombre de usuario
Email
Nombre
Apellido
Rol
encargadosala
encargado@test.com
Encargado
Sala
ENCARGADO_SALA
admin
admin@test.com
Ad
Min
ADMIN
1
5
En total hay 2 registros.
Confirmar la Eliminación
¿Está seguro que desea eliminar el usuario?
No
Si

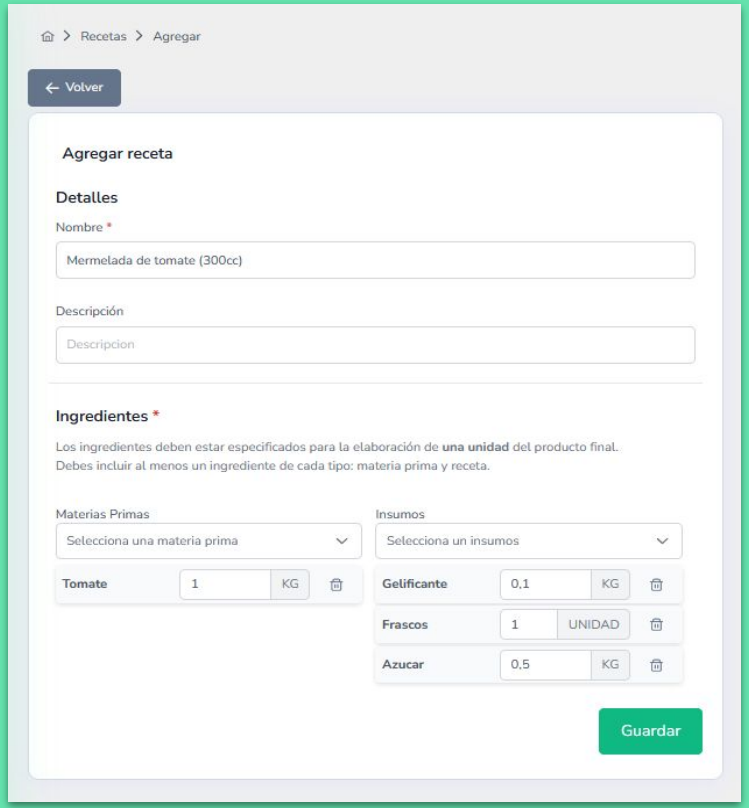
Desarrollo de funcionalidades

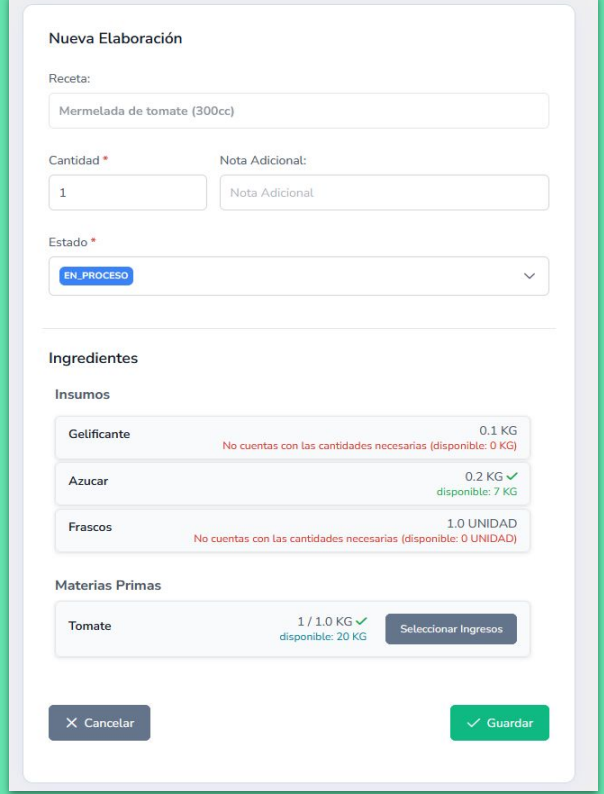
Una vez establecidas las bases del proyecto, nos enfocamos en implementar las **funcionalidades específicas** según las **historias de usuario**.

Algunas de estas funcionalidades, como los procesos de alta, baja y modificación (ABM) de usuarios y familias productoras, fueron relativamente sencillas.

Otras secciones, como el **ingreso de materias primas**, **insumos** y **recetas**, requirieron formularios más complejos. En estos casos, la tarea de realizar validaciones y gestionar los mensajes de error fue más demandante.


Materias Primas > Gestión de ingresos > Agregar
← Volver
Agregar Ingreso
Materia Prima:
Miel de abeja
Familia Productora *
Selecciona una familia productora
Valor de compra *
$ 0,00
Cantidad *
KG Ingrese una cantidad
Estado *
Selecciona un estado
Fecha de ingreso (opcional)
13/08/2024
Descripción (opcional)
✕ Cancelar
✓ Guardar


Recetas > Agregar
← Volver
Agregar receta
Detalles
Nombre *
Mermelada de tomate (300cc)
Descripción
Descripcion
Ingredientes *
Los ingredientes deben estar especificados para la elaboración de una unidad del producto final.
Debes incluir al menos un ingrediente de cada tipo: materia prima y receta.
Materias Primas
Selecciona una materia prima
Tomate 1 KG
Insumos
Selecciona un insumos
Gelificante 0,1 KG
Frascos 1 UNIDAD
Azucar 0,5 KG
Guardar


Nueva Elaboración
Receta:
Mermelada de tomate (300cc)
Cantidad *
1
Nota Adicional:
Nota Adicional
Estado *
EN_PROCESO
Ingredientes
Insumos
Gelificante 0.1 KG
No cuentas con las cantidades necesarias (disponible: 0 KG)
Azucar 0.2 KG ✓
disponible: 7 KG
Frascos 1.0 UNIDAD
No cuentas con las cantidades necesarias (disponible: 0 UNIDAD)
Materias Primas
Tomate 1 / 1.0 KG ✓
disponible: 20 KG
Seleccionar Ingresos
✕ Cancelar
✓ Guardar

Desarrollo

Desarrollo final y conclusiones

Trabajando en equipo

El trabajo en equipo fue fundamental para poder cumplir con los objetivos de cada entrega en tiempo y forma, tanto para realización de informes, historias de usuario, maquetados, e implementacion. Algunos de los aspectos que tratamos de cuidar como equipo fueron:

Distribución de Tareas: nos pusimos de acuerdo en la asignación de tareas, definiendo claramente qué parte de cada entrega le correspondía a cada uno, lo cual evitó solapamientos.

Manejo de Conflictos: Para minimizar conflictos en el código, tratamos de hacer commits frecuentes, y evitar tocar archivos con los que estaba trabajando el otro.

Revisión y Pruebas Cruzadas
Cada uno revisaba el trabajo del otro, lo cual ayudó a detectar errores, y mantener consistencia en algunas interfaces, navegación, y detalles que cada uno hacía como le parecía mejor, llegando a acuerdos.

Progreso Actual y Mejoras en Desarrollo

Hasta el momento de realizar esta presentación, nos encontramos trabajando en completar las funcionalidades restantes, aquellas que no estaban incluidas en las entregas anteriores, con el objetivo de lograr un prototipo funcional completo de la aplicación.

Algunas de las mejoras en las que estamos trabajando son:

- **Autenticación** mediante JWT tokens.
- Implementación de **tema claro/oscuro**.
- **Filtros** de búsqueda y ordenamiento adicionales en los listados.
- **Enlaces adicionales** para facilitar la navegación. Por ejemplo, desde el detalle de una receta se puede hacer clic en una materia prima y acceder directamente al detalle de la misma. Esto también se aplica a todas las otras entidades.
- ABM de elaboraciones, puntos de venta, entregas a puntos de venta, y actualización de estados de elaboraciones automáticas.

¡Muchas gracias!

Leandro Svetlich
Ana Cossio