# Arquitectura de Software Corte 2 – Parcial

# Miembros: Promoción: 16A - Camilo Sánchez, Miguel Rocha, Julian Bejarano

**Diseño del Sistema Alimenta**

## 1. Resumen del Problema y Contexto Estratégico

El proyecto AliMenta ha nacido porque se quiere abordar una de las contradicciones más severas que ha afrentado Colombia. La coexistencia de una crisis de inseguridad alimentaria en conjunto al desperdicio masivo de alimentos perfectamente consumibles. Esta situación no solo representa una falla logística y de información, sino que también representa una emergencia social y una ineficiencia económica que demanda soluciones innovadoras y sistémicas.

### La Doble Crisis: Inseguridad Alimentaria y Desperdicio Masivo

Por un lado, la malnutrición ha persistido como una emergencia de salud pública, según datos conjuntos del Departamento Administrativo Nacional de Estadística y la Organización de las Naciones Unidas para la Alimentación y la Agricultura para el año 2024 ha revelado que el 25.5% de los hogares en Colombia, haciendo un equivalente a más de 14.4 millones de personas, experimentan inseguridad alimentaria de moderada a grave.Esta situación es particularmente crítica en las zonas rurales, donde la prevalencia alcanza el 34.2%, y en departamentos como La Guajira, se ha registrado un alarmante 52.4%.El impacto más trágico de esta realidad se refleja en la etapa de la niñez, con más de 1,748 muertes de menores de cinco años por causas que han sido asociadas a la desnutrición entre 2017 y 2023.

Por otro lado, el país desperdicia una cantidad monumental de recursos. Según el estudio más exhaustivo del Departamento Nacional de Planeación, en Colombia se pierden o desperdician anualmente más de 9.76 millones de toneladas de alimentos. De esta cifra, un 40.5% (3.95 millones de toneladas) se pierde en las etapas de distribución y consumo. Específicamente, el sector

retail (supermercados y minoristas) es responsable del desperdicio de 2.01 millones de toneladas, compuestas mayormente por frutas, verduras y tubérculos, alimentos ricos en micronutrientes esenciales que podrían cerrar brechas nutricionales en poblaciones vulnerables.

### Contexto Socioeconómico y Tecnológico

La viabilidad y relevancia de AliMenta se fundamentan en un análisis del entorno que revela una confluencia de factores favorables:

* **Social:** El sistema se enmarca en un contexto de persistente desigualdad, donde el 33.3% de la población se encontraba en situación de pobreza monetaria en 2024, limitando el acceso a la canasta básica. Sin embargo, existe una creciente y robusta red de organizaciones de la sociedad civil, como fundaciones y bancos de alimentos, dedicada a mitigar el hambre.Esta red no es solo un actor pasivo, sino la infraestructura social fundamental sobre la cual AliMenta puede operar, proveyendo los usuarios receptores validados y la capacidad logística en el terreno.
* **Económico y Legal:** El marco legal colombiano ofrece un poderoso incentivo para la donación. La Ley 1990 de 2019 prioriza el consumo humano para los excedentes alimentarios y, en conjunto con el Estatuto Tributario, permite a las empresas donantes acceder a significativos beneficios fiscales.Aunque las normativas específicas pueden variar, el incentivo transforma la donación de un acto puramente altruista a una decisión financieramente inteligente frente al costo de desechar productos. Esta perspectiva es crucial, ya que posiciona a AliMenta no como una plataforma de caridad, sino como una herramienta de optimización de negocio con un claro retorno de inversión para los supermercados, cuyo subproducto es un impacto social positivo. Este enfoque B2B (Business-to-Business) debe guiar el diseño, priorizando funcionalidades que faciliten la gestión de excedentes y la obtención de beneficios tributarios.
* **Técnico:** La viabilidad tecnológica es alta. Con más de 49 millones de accesos a internet móvil en Colombia para finales de 2024, la penetración alcanza el 94% de la población. La amplia adopción de celulares inteligentes en todos los estratos socioeconómicos asegura que una plataforma web con diseño responsivo (mobile-first) para que sea accesible tanto para el personal operativo de los supermercados como para los coordinadores de las fundaciones.

Adicionalmente, el marco legal impone restricciones críticas que deben ser consideradas desde el diseño. La Ley 1581 de 2012 de Protección de Datos Personales exige un manejo riguroso de la información de los usuarios. Esto implica que la seguridad y la privacidad no son solo atributos de calidad deseables, sino requisitos legales ineludibles. El incumplimiento no representa un fallo técnico, sino un riesgo legal y reputacional para el proyecto y sus participantes.

### Declaración de la Misión del Proyecto

AliMenta es un puente tecnológico que conecta el excedente de alimentos consumibles en el sector *retail* con las organizaciones que combaten la inseguridad alimentaria en Colombia, transformando una falla logística y de información en una oportunidad de impacto social, económico y ambiental.

## 2. Modelo Conceptual de Datos

Para estructurar la información que gestionará el sistema AliMenta, se define un modelo conceptual de datos que refleja las entidades principales del dominio y sus interrelaciones. Este modelo es la base sobre la cual se construirán la lógica de negocio y la persistencia de datos.

### Entidades Principales

* **Usuario:** Representa a cualquier individuo que interactúa con el sistema. Funciona como una entidad base que se especializará según el rol. Atributos clave incluyen un identificador único, nombre, correo electrónico, contraseña (almacenada con hash y salt), y el rol asignado.
* **Organizacion:** Representa a las entidades legales que participan en la plataforma, ya sean Donantes (supermercados) o Beneficiarios (fundaciones, bancos de alimentos). Esta entidad contendrá información de verificación legal como NIT, razón social, dirección y estado de validación. La separación entre Usuario y Organización es una decisión de diseño deliberada. Permite que una organización tenga múltiples usuarios asociados con diferentes permisos, lo que mejora la escalabilidad y la gestión de la seguridad. Por ejemplo, un gerente de tienda puede crear donaciones, mientras que un gerente regional puede acceder a reportes tributarios consolidados. Esta estructura también facilita el cumplimiento de la Ley 1581, ya que los datos personales del individuo (Usuario) se gestionan de forma independiente a los datos corporativos (Organización).
* **Donacion:** Es la entidad central del sistema, representando una oferta de alimentos por parte de una organización donante. Sus atributos incluyen un estado (ej. disponible, asignada, recogida, cancelada), fechas de creación y de vencimiento estimada de los productos, y la ubicación para la recogida.
* **ItemAlimento:** Detalla cada tipo de alimento dentro de una donación. Permite registrar múltiples productos en una sola oferta. Sus atributos son el nombre, descripción, cantidad, unidad de medida (ej. kg, unidades) y categoría (ej. frutas, verduras, lácteos).
* **Solicitud:** Modela el interés de una organización beneficiaria en una donación específica. Contiene un estado (ej. pendiente, aprobada, rechazada) y la fecha en que se realizó.
* **Reporte:** Almacena datos agregados para la generación de informes de impacto y tributarios, un requerimiento funcional clave. Sus atributos incluyen el tipo de reporte, el período de tiempo que cubre y los datos consolidados.

### Relaciones y Cardinalidad

El mapa conceptual de relaciones se define de la siguiente manera:

* Una Organizacion de tipo Donante puede crear **muchas** Donaciones (relación 1 a N).
* Una Donacion pertenece a **una sola** Organizacion Donante (relación 1 a 1).
* Una Donacion puede contener **muchos** ItemAlimento (relación 1 a N).
* Una Organizacion de tipo Beneficiario puede realizar **muchas** Solicitudes (relación 1 a N).
* Una Donacion puede recibir **muchas** Solicitudes, pero la lógica de negocio asegurará que solo una sea aprobada para evitar doble asignación.
* **Muchos** Usuarios pueden pertenecer a **una** Organizacion (relación N a 1).

A continuación, se presenta una tabla que formaliza este modelo, sirviendo como un contrato técnico para el equipo de desarrollo.

**Tabla 1: Definición de Entidades y Atributos Clave**

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de Dato Sugerido | Descripción |
| **Usuario** | id\_usuario | UUID (PK) | Identificador único del usuario. |
|  | nombre\_completo | VARCHAR(255) | Nombre y apellido del usuario. |
|  | email | VARCHAR(255) (Unique) | Correo electrónico para inicio de sesión. |
|  | password\_hash | VARCHAR(255) | Contraseña cifrada. |
|  | rol | ENUM('Donante', 'Beneficiario', 'Admin') | Rol del usuario en el sistema. |
|  | id\_organizacion | UUID (FK) | Organización a la que pertenece el usuario. |
| **Organizacion** | id\_organizacion | UUID (PK) | Identificador único de la organización. |
|  | nit | VARCHAR(20) (Unique) | Número de Identificación Tributaria. |
|  | razon\_social | VARCHAR(255) | Nombre legal de la organización. |
|  | tipo | ENUM('Donante', 'Beneficiario') | Tipo de organización. |
|  | direccion | TEXT | Dirección física principal. |
|  | estado\_validacion | ENUM('Pendiente', 'Validada', 'Rechazada') | Estado de la verificación legal. |
| **Donacion** | id\_donacion | UUID (PK) | Identificador único de la donación. |
|  | estado | ENUM('Disponible', 'Asignada', 'Recogida', 'Cancelada') | Estado actual de la oferta de donación. |
|  | fecha\_creacion | TIMESTAMP | Fecha y hora de publicación. |
|  | fecha\_vencimiento\_est | DATE | Fecha de vencimiento estimada de los productos. |
|  | ubicacion\_recogida | TEXT | Dirección y detalles para la recogida. |
|  | id\_donante | UUID (FK) | Organización donante que crea la oferta. |
|  | id\_beneficiario\_asig | UUID (FK, Nullable) | Beneficiario al que se le asigna la donación. |
| **ItemAlimento** | id\_item | UUID (PK) | Identificador único del ítem. |
|  | nombre | VARCHAR(255) | Nombre del producto (ej. "Tomate Chonto"). |
|  | cantidad | DECIMAL(10, 2) | Cantidad del producto. |
|  | unidad | VARCHAR(50) | Unidad de medida (ej. "kg", "unidades"). |
|  | id\_donacion | UUID (FK) | Donación a la que pertenece el ítem. |
| **Solicitud** | id\_solicitud | UUID (PK) | Identificador único de la solicitud. |
|  | estado | ENUM('Pendiente', 'Aprobada', 'Rechazada') | Estado de la solicitud de un beneficiario. |
|  | fecha\_solicitud | TIMESTAMP | Fecha y hora en que se realizó la solicitud. |
|  | id\_beneficiario | UUID (FK) | Organización beneficiaria que solicita. |
|  | id\_donacion | UUID (FK) | Donación que está siendo solicitada. |

## 3. Requerimientos

Los requerimientos del sistema se han divido en dos secciones, tanto funcionales como no funcionales, las cuales tienen como objetivo describir lo que el sistema debe hacer y cómo debe hacerlo. Estos requerimientos son la guía al diseño, desarrollo y pruebas de la plataforma del sistema AliMenta.

### Requerimientos Funcionales

Se han identificado los siguientes requerimientos funcionales:

* **RF1: Gestión de Perfiles y Autenticación:** El sistema AliMenta debe permitir tanto el registro y como la autenticación segura de usuarios con tres roles distintos: Donante que puede ser el personal de supermercado, Beneficiario (receptor) que puede ser el personal de una fundación y finalmente rol de Administrador. El rol de Administrador debe tener la función exclusiva de validar la existencia legal y la competencia de las diferentes organizaciones beneficiarias antes de activar sus cuentas, garantizando la legitimidad de la red.
* **RF2: Publicación y Gestión de Donaciones (Rol Donante):** Un usuario con rol Donante debe poder crear una nueva publicación de donación de manera rápida y sencilla. El formulario de creación debe permitir especificar múltiples Items de Alimentos, incluyendo el tipo, la cantidad estimada, y la fecha de vencimiento, así como la ubicación y el horario que el donante ha establecido para la recolección. El usuario debe poder consultar un historial de sus donaciones (pasadas y activas) y tener la capacidad de editar o cancelar ofertas que aún no hayan sido asignadas a un beneficiario “receptor”.
* **RF3: Búsqueda y Adjudicación de Donaciones (Rol Beneficiario “receptor”):** Un usuario con rol Beneficiario “receptor” debe poder buscar las donaciones disponibles en su área o ubicación más cercana. La búsqueda se puede filtrar por criterios clave como lo son la ubicación geográfica (proximidad), el tipo de alimento y la cantidad, esto con la finalidad de optimizar la logística. El usuario debe poder solicitar una donación disponible y recibir una confirmación clara y oportuna cuando esta le sea asignada por el sistema o el donante.
* **RF4: Sistema de Notificaciones en Tiempo Real:** El sistema AliMenta debe hacer la implementación de un mecanismo de notificaciones automáticas para agilizar la comunicación. Esto deberá alertar a los Beneficiarios “receptores” sobre las nuevas donaciones publicadas en las zonas de interés. Simultáneamente, se deberá notificar al Donante cuando su oferta ha sido solicitada y, posteriormente, asignada, cumpliendo con el objetivo de mantener a ambas partes informadas del estado del proceso.
* **RF5: Generación de Reportes de Impacto:** El sistema debe ser capaz de generar reportes consolidados para los roles Administrador y Donante. Estos reportes permiten visualizar las métricas de impacto clave, como los kilogramos totales de alimentos rescatados, el número estimado de beneficiarios impactados y la reducción de emisiones de CO2 equivalente, esto se calculará a partir de los alimentos que no terminaron en vertederos.
* **RF6: Generación de Certificados de Donación:** Para materializar el incentivo económico para los donantes, el sistema debe generar automáticamente un certificado formal por cada donación completada y recogida. Este documento, destinado al rol Donante, debe tener toda la información necesaria para servir como el soporte en la solicitud de beneficios tributarios, conforme a lo estipulado por la Ley 1990 de 2019.

### Requerimientos No Funcionales

Se priorizan los siguientes caracteres de calidad para asegurar el éxito y la correcta adopción de la plataforma:

* **RNF1: Usabilidad (Prioridad: Alta):** La interfaz para el usuario debe ser extremadamente intuitiva y seguir un diseño mobile-first, dado que será utilizada en entornos operativos dinámicos. Como métrica de éxito, un nuevo usuario debe poder ser capaz de completar una tarea clave, como lo es publicar una nueva donación, en tiempo estimado de menos de 90 segundos durante su primer uso, sin necesidad de una capacitación formal.
* **Justificación:** La alta rotación de personal en el comercio minorista y el tiempo limitado de los voluntarios de las fundaciones hacen que una curva de aprendizaje mínima sea crucial para garantizar la adecuada adopción y el correcto uso consistente de la plataforma. Existe una tensión relacionada entre esta necesidad de simplicidad RNF1 y la necesidad de datos detallados para la generación de certificados RF6. Un formulario de publicación que exige demasiada información fiscal o logística puede crear que el usuario se aparte. Para resolver esto, se ha encontrado una solución del sistema que puede diseñarse para la captura de la información mínima y esencial en el momento de la publicación, para permitir que los detalles adicionales para el certificado sean completados posteriormente, ya sea por un usuario con rol de supervisor, desacoplando así la operación en tiempo real de la tarea administrativa.
* **RNF2: Fiabilidad (Prioridad: Alta):** El sistema debe garantizar una disponibilidad del 99.5%, excluyendo ventanas de mantenimiento programadas y comunicadas. Las transacciones críticas, como la creación de una donación y su asignación a un beneficiario, deben ser consistentes para prevenir errores críticos como la doble asignación de un mismo lote de alimentos.
* **Justificación:** La naturaleza pasajera de los productos donados significa que cualquier tiempo de inactividad del sistema puede resultar directamente en la pérdida de alimentos. Un fallo en la fiabilidad no solo tiene un costo económico, sino que también erosiona la confianza entre donantes y beneficiarios, pilar fundamental del ecosistema que AliMenta busca construir.
* **RNF3: Seguridad (Prioridad: Crítica):** El sistema debe ser diseñado para cumplir rigurosamente con la Ley Estatutaria 1581 de 2012 de Protección de Datos Personales de Colombia.Esto implica que toda la comunicación entre el cliente y el servidor debe estar cifrada mediante SSL/TLS. Las contraseñas de los usuarios deben almacenarse utilizando algoritmos de hashing y salting robustos. Además, se debe implementar un sistema de Control de Acceso Basado en Roles para asegurar que los usuarios solo puedan acceder a la información y funcionalidades estrictamente necesarias para su perfil.
* **Justificación:** El sistema manejará datos personales de los empleados de supermercados y miembros de fundaciones. Un fallo de seguridad no solo expondría a individuos, sino que también sometería al proyecto y a las organizaciones participantes a severas consecuencias legales y a un daño reputacional irreparable. La seguridad no es una característica, sino un pilar fundamental del diseño.

## 4. Patrones Aplicados

La selección de patrones de arquitectura y diseño es fundamental para construir un sistema robusto, mantenible y escalable. Las siguientes decisiones se han tomado para estructurar AliMenta de manera pragmática, equilibrando las necesidades actuales del proyecto con la visión a largo plazo.

### Patrón Arquitectural: Monolito en Capas (Layered Monolith)

Para la fase inicial del proyecto, se propone una arquitectura de **Monolito en Capas**. Este patrón organiza el código en capas lógicas y separadas (Presentación, Lógica de Negocio/Dominio, y Persistencia/Acceso a Datos) dentro de una única unidad de despliegue.

* **Justificación:**
  1. **Simplicidad de Desarrollo y Despliegue:** Para un equipo de desarrollo pequeño (3 miembros) y en la etapa de construcción de un Producto Mínimo Viable (MVP), la complejidad inherente a una arquitectura de microservicios (gestión de la comunicación entre servicios, descubrimiento de servicios, resiliencia distribuida, etc.) representa una sobrecarga técnica y operativa que podría retrasar significativamente el lanzamiento al mercado. Un monolito permite un desarrollo más rápido y un proceso de despliegue unificado y más sencillo.
  2. **Coherencia Transaccional:** Al operar con una única base de datos, se garantiza la consistencia transaccional de forma nativa. Esto es crítico para operaciones como la asignación de una donación, donde se debe asegurar que un lote de alimentos se asigne a un solo beneficiario sin riesgo de inconsistencias.
  3. **Alineación con el Stack Tecnológico:** El stack tecnológico propuesto, con Java (SpringBoot) en el backend, está optimizado para construir aplicaciones monolíticas en capas de manera eficiente y estructurada.

Esta elección arquitectónica también considera la estructura del equipo. Según la Ley de Conway, las organizaciones tienden a diseñar sistemas que reflejan su propia estructura de comunicación. Forzar una arquitectura de microservicios, que a menudo se beneficia de equipos independientes por servicio, en un equipo pequeño y cohesionado, crearía una fricción artificial. El monolito en capas se alinea de forma natural con un equipo que colabora estrechamente en una única base de código, optimizando así la productividad.

Para mitigar los riesgos de escalabilidad futuros, el monolito se diseñará de forma modular, con un bajo acoplamiento y alta cohesión entre los distintos dominios de negocio (Gestión de Usuarios, Gestión de Donaciones, Reportes). Esta separación lógica interna facilitará una eventual migración a una arquitectura de microservicios, donde cada módulo podría evolucionar para convertirse en un servicio independiente si el crecimiento del sistema lo justifica.

### Patrón de Diseño (Creacional): Factory Method

* **Aplicación:** Este patrón se aplicará en el módulo de Gestión de Usuarios (RF1) para la creación de objetos de usuario. Se definirá una interfaz User y clases concretas que la implementen (DonorUser, BeneficiaryUser, AdminUser). Una clase UserFactory contendrá un método createUser(role, data) que, dependiendo del parámetro role, instanciará y devolverá el objeto de la clase concreta correspondiente.
* **Explicación y Beneficios:** El patrón Factory Method desacopla el código cliente (que necesita un objeto de usuario) de la lógica de instanciación de las clases concretas. Esto adhiere al principio de Abierto/Cerrado: si en el futuro se introduce un nuevo rol de usuario, solo será necesario crear una nueva clase CarrierUser y actualizar la fábrica, sin modificar el código que invoca la creación. Esto aumenta la mantenibilidad y flexibilidad del sistema.

### Patrón de Diseño (Comportamental): Observer

* **Aplicación:** Este patrón es ideal para implementar el Sistema de Notificaciones en Tiempo Real (RF4).
* **Explicación y Beneficios:** El patrón Observer define una dependencia de uno a muchos entre objetos. Cuando un objeto (el "Sujeto") cambia de estado, todos sus dependientes (los "Observadores") son notificados automáticamente.
* **Sujeto:** El servicio de GestionDeDonaciones actuará como el generador de eventos. La creación de una nueva Donacion es el cambio de estado.
* **Observadores:** Las organizaciones Beneficiarias que se han suscrito para recibir alertas de donaciones en su área geográfica.  
    
  Flujo: Cuando un Donante publica una donación, el servicio de GestionDeDonaciones notifica a un “NotificationManager”, es decir, el Sujeto concreto. Este gestor recorre su lista de Beneficiarios suscritos, o sea, los Observadores y esto hace la invocación de su método update(), pasándoles la información de la nueva donación.  
    
  El principal beneficio es el bajo acoplamiento, porque el módulo de donaciones no necesita conocer los detalles de cómo se entregan las notificaciones, ya sea vía email, notificación push o un SMS. Se pueden agregar nuevos tipos de observadores, como, por ejemplo, un sistema de monitoreo logístico, sin la necesidad modificar el código del módulo de donaciones, esto permite mejorar la extensibilidad del sistema.

### Patrón de Idioma: Ejemplo de Factory Method en Java (SpringBoot)

El siguiente fragmento de código ilustra cómo se implementaría el patrón *Factory Method* para la creación de usuarios en el contexto de SpringBoot, utilizando la inyección de dependencias para gestionar la fábrica.

**Java**

// Interfaz común para todos los tipos de usuario  
public interface User {  
 String getRole();  
 // Otros métodos comunes  
}  
  
// Implementaciones concretas para cada rol  
public class DonorUser implements User {  
 @Override  
 public String getRole() {  
 return "DONOR";  
 }  
 // Atributos y métodos específicos para Donante  
}  
  
public class BeneficiaryUser implements User {  
 @Override  
 public String getRole() {  
 return "BENEFICIARY";  
 }  
 // Atributos y métodos específicos para Beneficiario  
}  
  
// La clase Factory, gestionada por Spring como un componente  
@Component  
public class UserFactory {  
 /\*\*  
 \* Crea una instancia de usuario basada en el rol proporcionado.  
 \* @param role El rol del usuario a crear ('DONOR', 'BENEFICIARY').  
 \* @return Una instancia de un objeto que implementa la interfaz User.  
 \* @throws IllegalArgumentException si el rol es desconocido.  
 \*/  
 public User createUser(String role) {  
 if ("DONOR".equalsIgnoreCase(role)) {  
 return new DonorUser();  
 } else if ("BENEFICIARY".equalsIgnoreCase(role)) {  
 return new BeneficiaryUser();  
 }  
 // Se podría añadir lógica para otros roles aquí  
 throw new IllegalArgumentException("Rol de usuario desconocido: " + role);  
 }  
}

## 5. Wireframes de la Interfaz de Usuario:

Los wireframes son bocetos son diseños, que definen la estructura visual y el flujo de interacción de la plataforma. A continuación, se describen los wireframes proporcionados y se realiza un análisis crítico para alinearlos con los objetivos funcionales del proyecto AliMenta.

### Descripción de los Wireframes Existentes

Los wireframes presentados en el documento Wireframes.pdf muestran tres pantallas principales:

1. **Pantalla 1 (Homepage):** Es la página de bienvenida. Su propósito es comunicar la misión de la plataforma ("Conectamos esperanza con acción"), presentar estadísticas impactantes sobre la desnutrición a nivel global y local, y ofrecer un llamado a la acción claro para que los nuevos usuarios se registren ("Crear Cuenta Gratuita").
2. **Pantalla 2 (Explorar Fundaciones):** Esta pantalla parece un dashboard para que un usuario explore y descubra diferentes fundaciones. Permite buscar y filtrar organizaciones, y muestra tarjetas de resumen con métricas como "Total recaudado", "Niños alimentados" y "Necesidades urgentes" en forma de proyectos que requieren financiación.
3. **Pantalla 3 (Detalle de Fundación y Proceso de Donación):** Esta vista detalla el perfil de una fundación específica y guía al usuario a través de un flujo para realizar una **donación monetaria**. Incluye elementos como selección de monto ($25, $50, $100), métodos de pago (tarjeta de crédito, transferencia) y la opción de hacer la donación mensual.

### Análisis Crítico y Propuesta de Adaptación

Se ha identificado una desconexión fundamental entre el flujo de usuario presentado en los wireframes y la misión central del proyecto AliMenta descrita en la documentación.1 Mientras que el proyecto se define como una plataforma logística para la

redistribución de excedentes de alimentos (donaciones en especie) de supermercados a fundaciones, los wireframes actuales están diseñados casi exclusivamente para facilitar donaciones monetarias de individuos a proyectos de fundaciones.

Para alinear la interfaz de usuario con los requerimientos funcionales del sistema, se propone la siguiente adaptación y rediseño de los flujos de usuario clave:

* **Flujo del Donante (Supermercado):**
  1. **Dashboard del Donante:** Al iniciar sesión, el usuario donante no vería una lista de fundaciones para donar dinero, sino un panel de control con sus donaciones de alimentos. Este dashboard mostraría una lista de sus ofertas activas (ej. "100 kg de plátanos - Disponible"), un historial de donaciones completadas y un botón prominente para "Publicar Nueva Donación de Alimentos".
  2. **Formulario de Publicación de Donación:** Al hacer clic en el botón anterior, se presentaría un formulario simple y rápido en línea con RNF1 para registrar una nueva donación. Los campos incluirían: descripción del alimento, cantidad estimada, kg, unidades, fecha de vencimiento y dirección/horario de recogida. Este flujo implementa directamente el RF2.
* **Flujo del Beneficiario (Fundación):**
  1. **Dashboard del Beneficiario:** Al iniciar sesión, el usuario beneficiario vería un mapa o una lista de donaciones de alimentos disponibles en su proximidad. La interfaz incluiría filtros por tipo de alimento, cantidad y distancia.
  2. **Vista de Detalle y Solicitud de Donación:** Al seleccionar una donación, el usuario vería los detalles completos y un botón para "Solicitar Recogida". Este flujo implementa directamente el RF3.

Esta adaptación asegura que la interfaz de usuario sirva directamente a los procesos de negocio definidos, conectando la oferta de excedentes de alimentos con la demanda de las fundaciones. La siguiente tabla mapea estos componentes de UI adaptados a los requerimientos y entidades del sistema.

**Tabla 2: Mapeo de Wireframes (Adaptados) a Requerimientos y Entidades**

|  |  |  |  |
| --- | --- | --- | --- |
| Componente UI (Adaptado) | Requerimiento Asociado | Entidades Involucradas | Flujo de Datos |
| **Botón "Publicar Nueva Donación"** (Dashboard Donante) | RF2: Publicación y Gestión de Donaciones | Donación, ItemAlimento | El usuario introduce datos que crean nuevas instancias de las entidades Donación e ItemAlimento en la base de datos. |
| **Lista de Donaciones Disponibles** (Dashboard Beneficiario) | RF3: Búsqueda y Adjudicación de Donaciones | Donación, Organización | El sistema consulta la base de datos para obtener las donaciones con estado 'Disponible' y las muestra al usuario, filtradas por su ubicación. |
| **Botón "Solicitar Recogida"** (Detalle de Donación) | RF3: Búsqueda y Adjudicación de Donaciones | Solicitud | La acción del usuario crea una nueva instancia de la entidad Solicitud, vinculando al Beneficiario con la Donación seleccionada. |
| **Notificación "Nueva Donación Cerca"** (UI Beneficiario) | RF4: Sistema de Notificaciones | Donación, Usuario | El sistema, a través del patrón Observer, empuja una alerta a la UI del beneficiario cuando se crea una nueva Donación que cumple sus criterios. |
| **Botón "Descargar Certificado"** (Historial de Donaciones) | RF6: Generación de Certificados de Donación | Donación, Reporte | El sistema genera un documento (ej. PDF) a partir de los datos de una donación completada, que el usuario Donante puede descargar. |

## 6. Conclusiones

Este informe ha detallado el diseño técnico y conceptual de la plataforma AliMenta, estableciendo una base sólida para su desarrollo e implementación. A través de un análisis exhaustivo del problema, la definición de un modelo de datos robusto, la especificación de requerimientos claros y la selección de patrones de arquitectura y diseño apropiados, se ha construido un plan coherente y justificado para la creación del sistema.

La integración de los elementos desarrollados prepara el proyecto de manera efectiva para el siguiente corte. La estrategia definida es la siguiente:

* El resumen del problema y el contexto (Sección 1) establecen una misión clara y validan la relevancia del proyecto, asegurando que el equipo de desarrollo comprenda el "porqué" detrás de su trabajo.
* El modelo conceptual de datos (Sección 2) proporciona el esquema de base de datos listo para ser implementado, sirviendo como una fuente única de verdad para la estructura de la información que manejará el sistema.
* Los requerimientos funcionales y no funcionales (Sección 3) actúan como una guía precisa para el desarrollo. Pueden ser directamente traducidos en historias de usuario y tareas en una metodología ágil, definiendo el alcance del Producto Mínimo Viable (MVP).
* Los patrones de arquitectura y diseño (Sección 4) ofrecen un marco probado para construir el sistema. La elección de un monolito en capas modular reduce la complejidad inicial sin sacrificar la escalabilidad futura, mientras que los patrones Factory Method y Observer aseguran un código limpio, mantenible y extensible para funcionalidades clave como la gestión de usuarios y las notificaciones.
* La adaptación de los wireframes (Sección 5) alinea la experiencia del usuario con los objetivos de negocio del sistema, asegurando que la interfaz no solo sea usable, sino que también resuelva el problema principal para el cual AliMenta fue concebido: la logística de donación de alimentos en especie.

En conjunto, estos entregables no son solo documentos aislados, sino componentes de un diseño de sistema integrado. El modelo de datos da estructura a los conceptos del problema; los requerimientos definen las operaciones sobre esos datos; los patrones dictan cómo implementar esas operaciones de manera eficiente; y los wireframes muestran cómo los usuarios interactuarán con el resultado final.

Esta documentación exhaustiva prepara al equipo para defender sus decisiones técnicas en el tercer corte, proporcionando la justificación detrás de cada elección arquitectónica y de diseño. Con esta base, el proyecto AliMenta está posicionado para pasar de la fase de diseño a la de implementación con una dirección clara, un alcance bien definido y una reducción significativa de la incertidumbre técnica.