Ambito de decision (Objetos / Arquitectura / Persistencia / Otro)	Componente/s impactado/s	Decisión	Otras Alternativas	Justificación de la decisión
Objetos	Medio de contacto	Utilizamos el patrón Strategy para los distintos medios de contacto.	Utilizar enum para los tipos de medios de contacto.	Permite extensibilidad y mantenibilidad a la hora de agregar nuevos medios de contacto.
Objetos	Persona vulnerable	No incorporamos un atributo explícito para saber si la persona vulnerable se encuentra en situación de calle.	Tener ambos atributos dirección y enSituaciónDeCalle(Bool).	Se puede calcular a partir de la nulleabilidad del atributo dirección.
Objetos	Dirección	Incorporar la clase dirección	Incluir los distintos atributos de una dirección en las entidades que la requieran (colaborador, heladera, etc.)	Permite almacenar toda la informacion posible de una direccion de forma estandarizada.
Objetos	Donacion monetaria	Tener una única entidad donaciónDeDinero con el atributo periodicidad como Bool o bien como entero cuyo valor 0 representaría una donación de dinero esporádica (no periódica)	Implementar dos entidades: una para la donación periódica y otra para las donaciones.	En las contribuciones de tipo Periodica (de dinero) al dejarlas en el historial, se deja constancia de que al cumplirse un intervalo se realizo una donacion de Dinero, creando un nuevo objeto de la clase DonacionMonetaria. Esto da la posibilidad de que en caso de desear hacer una unica contribucion monetaria, solo se representa con una donacion Periodica de intervalo 0 siendo este medido en dias. Aunque esto ocasiona la necesidad de que el sistema tambien se "encarge" de realizar efectivizar la donacion. Ante esto, se decidio optar por la simplificacion y tener unicamante la Donacion Monetaria
Objetos	Enums	Creación de enums.	Como string o booleano.	En general, permite estandarizar descripciones.
Objetos	Persona Juridica	No incorporar el atributo heladera/s en una persona jurídica.	Incluir el atributo heladeras como lista.	PersonaJuridica no tiene un atributo de heladeras ya que tiene una lista de formasDeContribuccion. Tenerlo haría que se deba modificar en 2 lugares al mismo tiempo (Heladera y persona juridica), es redundante y puede traer problemas en la base de datos.
Objetos	Persona vulnerable	No incorporamos un atributo explícito para saber si la persona vulnerable tiene menores a cargo.	Tener una entidad hijo con los datos de éste. Tener los atributos en personaVulnerable: tieneMenoresACargo (Bool) y cantidadMenores. Ademas de los que veamos oportunos (DNI, nombre, etc)	No es necesario tener un atributo para saber si tiene menores y otro para la cantidad. Si "cantidadDeMenores = 0" => no tiene No es necesario crear una entidad para los hijos, ya que estos, de momento, carecen de funcionalidad en el dominio del sistema (por ser menores)

Objetos	Vianda	Implementar una entidad "viandaEstandar" que tiene las dimensiones estándar de una vianda	Tener las dimensiones en Vianda o no contemplarlas (asumir que no cambiarán las dimensiones)	Sabemos que las viandas tienen un tamaño estándar, poniendo las dimensiones en una entidad podemos modificarlas a futuro.
Otros	Validador de contraseñas	Almacenar las contraseñas más comunes en una variable.	Leer el archivo de contraseñas cada vez que se usa el validador.	Si almacenamos las contraseñas más comunes en una variable, cualquier cambio al archivo de texto no se produciría hasta reiniciar el programa. Sin embargo, consideramos que las modificaciones que pueda sufrir el archivo es ampliamente menor a la cantidad de veces para leer, y la lectura de una variable es más eficiente que el costo de memoria que causa leer el archivo cada vez que se use el validador, por eso tomamos la decisión mencionada.
Objetos	Validador de contraseñas	Implementar cada política de contraseña como una clase diferente.	Implementar una única clase con diferentes métodos para cada política.	Se decidió implementar una clase por política pensando en que a futuro se puede exigir que para determinados usuarios se utilicen X políticas y a otros no. Además, facilita el mantenimieno a largo plazo de cada política frente a los cambios.
Objetos	Tarjeta	Se ubica el atributo "responsable" en tarjeta, como el colaborador que registra a un usuario vulnerable.	Incorporar el "responsable" del registro de una personaVulnerable, como atributo en esta última entidad, indicando por quién fue registrado.	El atributo en la personaVulnerable, genera acomplamiento entre esta y la lógica de negocio con respecto al registro, lo que podría dificultar la modificación y mantenimiento del código en el futuro.
Objetos	Heladera	Colocar los atributos temperatura min y max en la heladera.	Crear una nueva clase "modelo Heladera" para indicar según éste la temperatura min y max.	Es un atributo configurable por el usuario, por lo tanto, un mismo modelo de heladera puede tener distintas temperaturas límites.
Objetos	SensorTemperatura	Llevar un registro de todas las temperaturas pasadas de una heladera y obtener la actual buscando la última fecha.	Restringir al conocimiento del último y actual valor de temperatura de la heladera.	Consideramos que podría ser necesario obtener reportes sobre este atributo para conocer el funcionamiento de la heladera.
Objetos	SensorTemperatura - SensorMovimiento	Utilizar una clase para cada tipo de sensor y que cada uno lleve el registro de sus activaciones.	Reducir a una única clase "sensor" con su tipo y llevar un registro global de activaciones de los distintos sensores.	LLevar un registro único de activaciones de los sensores podría volverse difícil de gestionar a medida que aumenta el número y la diversidad de sensores, lo que podría afectar negativamente el rendimiento y la escalabilidad del sistema. Separando en cada clase, simplifica aumenta la cohesividad de los componentes, la mantenibilidad, y la extensibilidad

Objetos	Documento - PersonaVulnerable	Crear clase Documento	Agregar los atributos en la clase que requiera su uso, por ejemplo, PersonaVulnerable	Consideramos que es necesaria la existencia de una clase de Documento donde se almacena toda la informacion de un "Documento de Identidad" para poder normalizar el uso de estos datos y brindar flexibilidad al sistema en caso de que se requieran realizar mas operaciones con los documentos o agregar nuevos atributos.
Objetos	Premio	El premio puede ser reclamado por un colaborador. Metodo premio. reclamar(colaborador)	El colaborador puede reclamar premios. Metodo colaborador.reclamar(premio)	Consideramos que es responsabilidad del Premio conocer la lógica de las restricciones que permite o no para ser reclamado por un Colaborador. Si estuviera en el Colaborador, rompería el encapsulamiento.
Objetos	Premio	Premio posee la fecha de Acreditacion	No tener fecha en los premios	La existencia del atributo fecha hace que podamos a futuro realizar un historico de en que fechas se reclamaron premios pudiendo explotar esta informacion para provecho de las empresas
Objetos	FormaImportacion	El metodo importar recibe un string	El metodo importar recibe un conjunto de byte[]	Al utilizar un formato de string es posible que a futuro se pueda ademas de pasar un archivo (codificado por ejemplo en Base64) se pueda pasar una URL para obtener los datos de las colaboraciones
Objetos	Score	Los coeficientes se leen por archivo de configuracion.	Tener los parámetros en una clase de configuraciones del sistema o para los coeficientes requeridos.	Al leer los coeficientes desde un archivo de configuración, podemos modificarlos fácilmente. Aunque estas modificaciones requieran reiniciar el sistema, como esto sucede, a lo sumo, una vez por mes, consideramos que es un riesgo que podemos asumir.
Objetos	Score	Cada forma de contribución calcula sus puntos y cuando el colaborador contribuye se llama a la clase Score para agregar los puntos a ese colaborador. Se suman los puntos cada vez se realiza una contribución.	Que el colaborador calcule sus puntos en base a la lista de contribuciones. Sumar los puntos cuando el colaborador quiera ver sus puntos o cuando se haga una colaboración masiva, a partir de las cantidades indicadas en el CSV.	Al separar la lógica de cálculo en cada forma de contribución y la asignación de puntos en la clase Score se flexibiliza para que si a futuro se desea imponer nuevas reglas al momento de calcular solo se tenga que modificar la clase score, por ejemplo un límite máximo de puntos.

Objetos	Heladera - Modelo Heladera	Crear una clase que contenga los atributos temperatura min y max en la heladera que representen los limites predeterminados por el modelo de heladera; y los atributos de temperatura min y max configurables por el usuario en la heladera	Omitir la clase modelo heladera e incorporar los atributos duplicados en la heladera o solo dejar los configurados por el usuario en la heladera y validar los limites de temperatura acordes al modelo dentro del constructor.	Es un atributo configurable por el usuario, por lo tanto, un mismo modelo de heladera puede tener distintas temperaturas límites, pero siempre entre las establecidas del modelo.
Objetos	Clase abstracta sensor	Utilizar clase abstracta sensor	Utilizar Interfaz sensor	La clase otorga métodos de identificación de tipos, conveniente para el envio de notificaciones, ante una variación detectada por el sensor, hacia la heladera
Objetos	Premio	Los premios son únicos	Template premio para dar de alta varios premios de un mismo tipo	Los colaboradores ofrecen premios únicos y diferentes entre ellos.
Objetos	Premio	Atributo ReclamadoPor: Colaborador	Lista de premiosReclamados en Colaborador	Responsabilidad y encapsulamiento. Identificar quién reclama el premio desde esta entidad.
Objetos	AutorizacionManipulacionHeladera	Cantidad de usos de la autorización	Que la autorizacion pueda utilizarse solo una vez	Una autorización puede ser usada múltiples veces dentro del rango horario entre fechaCreacion y fechaExpiracion. Esto permite que la heladera pueda abrirse y cerrarse varias veces mientras se ingresan o retiran viandas. Pensado para escenarios en los que deban ingresarse cantidades grandes de viandas y por algún motivo la heladera se cierra, la autorización deberia ser válida todavía
Objetos	AutorizacionManipulacionHeladera	Omitir el tipo de autorizacion (retiro o ingreso de vianda)	Guardar el tipo de autorizacion para permitir acciones especificas en una tarjeta	Por simplicidad, la autorizacion es para abrir la heladera unicamente. Ya que en los requerimientos se especifica que se chequea el permiso al momento de abrir la heladera, momento en el cual no se sabe si se va a hacer un ingreso o un retiro, por lo que si se agregase ese atributo, estaria en desuso
Objetos	AccesoHeladera	Relacion con autorizacion = null para tarjetas de consumo	Excluir a las tarjetas de consumo del historial que se crea en esta clase, y a su vez, crear otra clase que registre unicamente los accesos de las tarjetas de consumo	Las personas que usen tarjetas de consumo no necesitan autorizacion para abrir la heladera, por lo que, si bien se registra el acceso dentro de la misma clase (AccesoHeladera), la autorizacion se setea como un valor null

Objetos	Persona - Rol	Se utilizan dos clases para diferenciar los tipos de persona (juridica o humana) y el rol de negocio que pueden ocupar (persona vulnerable, tecnico, colaborador, usuario). Entre ellos el rol tiene la persona y una persona la lista de roles adquiridos	Una clase persona que de esta se desprendan colaboradores, personas vulnerables y técnicos. De los colaboradores se desprenden personas fisicas y jurídicas. No se relaciona técnicos ni personas vulnerables con personas fisicas o juridicas, como si lo puede ser en la realidad.	Acerca el modelo a la realidad. Además, permite flexibilidad y extensibilidad para que una persona pueda cambiar de rol o tener varios; facilita la combinación entre tipo de persona y un rol, para no restringir el rol a un tipo de persona (física o jurídica)
Otro	Operaciones	Se decide separar el comportamiento de las funcionalidades del sistema en una capa de operaciones donde sus clases tienen la estructura del patron Command con su Comando, Handle y Validaciones en caso de ser requeridas	Incorporar las funcionalidades dentro de las clases de dominio	Las clases de dominio se mantienen enfocadas en contener los datos y relaciones, mientras que las clases de operaciones gestionan la lógica de negocio y los casos de uso, promoviendo la cohesión, la mantenibilidad, la extensibilidad y testeabilidad del sistema.
Otro	INotificacionBuilder - IReporteBuilder	Implementar dos patrones Builder, uno para la notificacion de suscripciones y el registro de usuario, y otro para la creación de reportes	Utilizar un único patrón.	Permite construir las distintas notificaciones con sus respectivos mensajes e información, sea de suscripciones hacia los colaboradores o la creación de usuario, de forma que es polimórfico y en el futuro es modificable fácilmente. Además de esta forma queda desacoplado respecto de los medios de contacto a los cuales se envia el mensaje. Como los reportes son conceptos sustancialmente distintos y tienen distinta frecuencia a las notificaciones, es necesario separar la creacion de estos con la de suscripciones.
Arquitectura	Sistema para la Mejora del Acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica	Los estereotipos << Base de datos >> se relacionan con el componente central "Sistema para la mejora del acceso alimentario en contextos de vulnerabilidad socioeconómica"	Los estereotipos << Base de datos >> se relacionan con los módulos más particulares que lo requieren. Por ejemplo: "<< Base de datos >> Usuarios del Sistema" con componente "Validador de contraseñas"	El acceso a las bases de datos queda controlado por el componente central "Sistema para la Mejora del Acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica", el cual se encargará de buscar los datos necesarios para el procesamiento de las distintas funcionalidades evitando que los componentes accedan a datos los cuales no deberian ver.

Arquitectura	<<Base de datos >> - Persona - Usuarios del sistema - Personas vulnerables - Técnicos - Colaboradores	Plantear una base de datos persona y una base de datos para cada rol.	Única base de datos de persona o solamente las bases de datos por roles.	Ciertos roles tienen mayor volumen de operaciones que el resto, por lo tanto tener bases de datos separadas permite distribuir mejor la carga sin afectar el rendimiento de consultas para otros. Además facilita la evolución independiente de cada rol. La base de datos persona solo guardará los atributos compartidos y es la que se auditará.
Arquitectura	Gestor de incidentes	Incorporacion de un componente para toda la gestion de los incidentes (reportar incidentes, si este fue resuelto o no).	En el Sistema para la mejora del acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica se reporten de los incidentes, poder saber si este fue resuelto o no.	Permite ingresar incidentes, observar los sensores y generar las notificaciones para enviarse.
Arquitectura	Gestor de suscripciones	Incorporacion de un componente para toda la gestion de las suscripciones de los colaboradores para que estos sean notificados.	Implementar en el Sistema para la mejora del acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica las suscripciones de los colaboradores.	Permite a los colaboradores optar por suscribirse a las heladeras en zonas donde frecuentan y emitir las notificaciones a los suscriptos.
Arquitectura	Emisor de notificaciones	Incorporar un componente especializado en el control del ciclo de vida de las notificaciones a traves de diferentes medios	Implementar en el sistema el control de envio y trazabilidad de la notificaciones	Delegar la lógica de las notificaciones en un componente separado nos permitiría tercierizar estas acciones a un proveedor externo
Arquitectura	Calculador de puntos de colaboraciones	Incorporar un componente cuyo único objetivo sea a partir de unas colaboraciones iniciales calcular los puntos de cada una.	Implementar en el sistema el calculo de los puntos de cada colaboracion	Delegar la logica en un componente especializado nos permite que a futuro con solo cambiar dicho componente sea mas facil la forma de calulcar puntos por cada una de las colaboraciones
Arquitectura	Generador de reporte	Incorporar un componente especializado en la creación de reportes.	Implementar en el sistema de forma integral la logica de creacion de reportes	Al permitir cambiar el componente de generacion de reportes se puede cambiar el reporte generado para la misma solicitud dependiendo los requerimientos
Arquitectura	Importador de Colaboradores	Incorporar un componente separado del sistema central para la importación de colaboradores.	La importación de colaboradores queda a cargo del Sistema para la Mejora del Acceso Alimentario en Contextos de Vulnerabilidad socioeconómica.	Separar el componente otorga mayor flexibilidad para incorporar nuevas formas de importación y complejizar el mismo, sin afectar otras funcionalidades del sistema. Además deja la posibilidad a delegar a futuro esta funcionalidad en proveedores externos.

Arquitectura	- Validador de contraseñas - Validador de accesos a heladeras	Las validaciones se encontrarán a cargo de componentes particulares.	Incorporar la lógica de las validaciones dentro del Sistema para la mejora del acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica.	Debido a su alta probabilidad de modificaciones sobre las validaciones, decidimos delegar en un componente específico.
Arquitectura	- Sistema de recomendación de puntos estratégicos para colocación de heladeras - Selector de técnicos	Componentes específicos para la implementación de las funcionalidades involucradas.	Incorporar las funcionalidades mencionadas dentro del Sistema para la mejora del acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica.	La complejidad de estas funcionalidades y la posibilidad de extensión de las mismas, ameritan la delegación en componentes específicos.
Arquitectura	Cargador de contribuciones	Componente para que los colaboradores cargen las contribuciones que hayan realizado.	Implementar en el sistema las funcionalidades para la carga de las contribuciones que fueron realizadas por los colaboradores.	La complejidad de las funcionalidades de la carga de contribuciones y la posibilidad de extensión de las mismas ameritan la delegación en componentes específicos.
Arquitectura	Selector de técnico	Incorporar un componente separado del sistema central para la selección de los técnicos para que estos resuelvan los incidentes por cercanía.	La selección de los técnicos queda a cargo del Sistema para la Mejora del Acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica.	La complejidad de las funcionalidades de la selección de los técnicos para que estos resuelvan los incidentes por cercanía y la posibilidad de extensión de las mismas, ameritan la delegación en componentes específicos.
Arquitectura	Localizador	Incorporar un componente separado del sistema central para la localización de los técnicos que servirá en la selección de los técnicos en la resolución de los incidentes.	La localizacón del técnico queda a cargo del Sistema para la Mejora del Acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica.	La complejidad de las funcionalidades de la localización de los técnicos para la selección de estos últimos para que resulvan los incidentes ameritan la delegación en componentes específicos.
Otro	Operaciones	Se decide separar el comportamiento de las funcionalidades del sistema en una capa de operaciones donde sus clases tienen la estructura del patron Command con su Comando, Handle y Validaciones en caso de ser requeridas	Incorporar las funcionalidades dentro de las clases de dominio	Las clases de dominio se mantienen enfocadas en contener los datos y relaciones, mientras que las clases de operaciones gestionan la lógica de negocio y los casos de uso, promoviendo la cohesión, la mantenibilidad, la extensibilidad y testeabilidad del sistema.