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.