**Justificaciones del diagrama**

Las contribuidores, que están diferenciados por persona humana y persona jurídica, decidimos agruparlos en una clase **Persona** y manejar las diferencias por composición ya que los 2 tipos de persona tienen el mismo comportamiento, agregan contribuciones, compran productos, se suscriben a notificaciones, etc.

Las **contribuciones** son clases y sus atributos son los datos solicitados al momento de realizar la contribución. De esta forma, cuando se contribuya, se instancia un nuevo objeto con la información ingresada. Además, cada contribución conoce al colaborador que la realizó, de esta forma podemos separar si la contribución es humana o jurídica, delimitando el tipo de persona que la puede realizar. También, hay una **interfaz Contribución**, la cual tiene el común el método para el cálculo de puntos.

Para cuando una persona jurídica quiera conocer los **puntos recomendados para la colocación de una heladera** decidimos utilizar el patrón “adapter”, ya que utilizamos una API externa. De esta forma ganamos más desacoplamiento de la implementación de la API, desligándonos de esta.

Las personas vulnerables tienen todos sus tipos de datos correspondientes, y, además, los menores a cargo son una lista de PersonaVulnerable, ya que estos menores también son personas vulnerables. Para sacarlas de esta lista, existe el método para calcular la edad dada su fecha de nacimiento y otra para comprobar si es mayor, y cuando sea mayor de edad se lo extrae.

La **tarjeta** posee un código de identificación único, para ello usamos una librería de Java que nos proporciona un UUID. Está asociada a una personaVulnerable, por esa razón conoce a su dueño. Luego tiene su cantidad de usos y un método asociado para ir calculando este valor, ya que varía dependiendo de la persona relacionada con esta. Además, la tarjeta misma tiene una lista de todos los registros, los cuales son todas veces que se haya utilizado para sacar una vianda y de esta forma tener constancia de todo.

La **heladera** tiene sus datos asociados, como su capacidad, su locación con un puntoGeografico, fechaDeFuncionamiento, el estado para saber si esta activa o no. Cada heladera tiene asociada 2 sensores que la conocen y envían a un receptor los datos que obtienen, esto para separar la lógica propia de cada sensor y de sus validaciones de la heladera.

Luego generamos la clase **Direccion** para tener la información de forma más precisa y detallada, de esta forma ganamos más consistencia de datos y flexibilidad. Siguiendo por la misma línea, realizamos lo mismo para el **PuntoEstrategico, Comida, AreaCobertura.**

Los **medios de contactos** los diseñamos como una clase que indique tipo, para que se pueda saber qué tipo de medio de contacto es, y el medio de contacto en sí. Se decidió modelarlo de esta forma para que puedan agregarse nuevos medios de contacto permitidos de forma dinámica, sin tener que definir un atributo para cada tipo de medio de contacto, y para que el sistema sepa qué medio de contacto está recibiendo.

Los **formularios** son un conjunto de **preguntas** con un título, mientras que el **formulario respondido** es un conjunto de **respuestas** que hacen referencia a un formulario, nos parece una buena abstracción de lo que es un formulario y nos permite crear y modificar formularios de forma dinámica. En cuanto a las preguntas y respuestas decidimos guardar sus valores en un string, pero poder identificar el tipo de valor para procesarlo, sabemos si la respuesta es un número, un string, un color o lo que sea que necesitemos analizar.

El cálculo de puntos se lo delegamos a una **CalculadoraDePuntos** para que sea más cohesivo, lo único que hace la calculadora es preguntar y sumar los puntos de las contribuciones, de esta forma, la calculadora se desliga de la lógica de cálculo de puntos de las contribuciones, haciendo a nuestro sistema de puntos más extensible.

Entrega 3:

* Las tarjetas de los colaboradores las representamos mediante **TarjetaPersona**. Similar a las tarjetas de las personas vulnerables, las tarjetas de los colaboradores conocen al colaborador asociado y guardan registro de sus usos en un **RegistroAperturaHeladera**.
* Las solicitudes de apertura de las heladeras se guardan en las heladeras en **SolicitudSobreHeladera** para que maneje los permisos del colaborador al intentar abrir una heladera, usando el colaborador que dio la solicitud y a la que la dio.
* Decidimos que los incidentes los reportes los receptores de los sensores, encargados de recibir y analizar la información de los sensores. Al ser los métodos de evaluación distintos, pero los pasos que le siguen a la detección de un incidente iguales, decidimos dejarle la evaluación a cada receptor y delegarle el procesamiento del incidente a un **ProcesadorDeIncidentes** que se encarga de crear el incidente y notificar al técnico correspondiente.
* Los técnicos son avisados cuando se crea un incidente usando un **Notificador**, responsable de enviar mensajes a través de distintos medios, y un **RadarDeTecnicos**, que devuelve un técnico cercano dada una heladera. Decidimos que los técnicos sean responsables de registrar sus visitas en un **RegistroVisitaTecnico** y que estas sepan, además de la información obligatoria de la visita, el técnico que hizo la visita, si soluciono el incidente, y que sepa decir si soluciono un incidente en concreto. De esta forma podemos saber que técnico soluciono que incidente y también si un técnico soluciono o no un incidente en específico.
* Pensamos en que los colaboradores se suscriban a las heladeras a través de una **Suscripción**. Decidimos también separar los tipos de notificaciones a las que puede suscribirse el colaborador en clases que definan como y cuando se debe activar la notificación para esa suscripción. Creemos que es una solución cohesiva y extensible.
* Los reportes decidimos generarlos a partir de un “singleton” **Reporte** que se encargue de utilizar toda la información recolectada en la semana para generar los reportes utilizando un cronjob. Para que el reporte no ande preguntando todo el tiempo a los componentes que tienen los datos o que conozca su funcionamiento interno, decidimos que posea métodos que reciban la información que necesita el reporte y que se vaya cargando de la información que necesita de esa semana, así cada vez que sucede algo que debería considerar el reporte, se llama al reporte para darle esa información.
* Para que una persona se pueda suscribirse a cierta heladera primero se hace el chequeo con suscribirseA(heladera). Este simplemente corrobora que la persona y la heladera sean de la misma localidad
* Las suscripciones están modeladas con una composición y dependiendo del tipo de suscripción el método cumpleCondcion(heladera), tiene un comportamiento u otro. En todos los casos devuelve un booleano. También tiene un método el cual es el encargado de llamar al Notificador, en caso de que se cumpla la condición en la heladera para activarse.
* Luego, en la heladera tiene tres métodos, un setter de estado y otros dos, para aumentar y disminuir las viandas. Siempre que se llame a alguno de estos luego de que se settean los nuevos valores, se llama al método cumpleCondcion, el cual recorre la lista de Suscripciones que tiene la heladera y llama al Notificador para informar, en los casos que se cumplan las condiciones solicitadas de esa suscripción.
* El notificador conoce a los tres enviadores actuales, el de mails, de Telegram y Whatsapp. Todos son singletons y están desacoplados lo mayor posible para que cualquier componente los pueda usar. El notificador tiene un método notificar(medioContacto, mensaje) el cual tiene un switch con las tres formas de contacto posibles según nuestro dominio. Y dependiendo de cual se elige se utiliza un enviador u otro.
* Para el enviador de Whatsapp se necesita un número destino y el mensaje. Para el de mails settear el asunto, mail destino y mensaje. Por último, el de Telegram es un poco distinto, en este se le debe solicitar primero al usuario que inicie una conserva con nuestro bot: @TPA\_DDS\_2024\_GRUPO\_12\_BOT, este cuando inicie la conversación va a proporcionar un chatID, esa va a ser el lugar destino que se enviaran los mensajes. Entonces para Telegram usamos ese chatID y el mensaje correspondiente