UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Diseño de Sistemas

Nombre del trabajo: "TPA": Entrega 2
Grupo: N°11 - SiMeAl
Curso: K3101 - K3001
Profesor: Ezequiel Escobar
Fecha de entrega: 28/05/2024.

Nombre, Apellido	Mail	Legajo
Tomás Pauza Sager	tpauzasager@frba.utn.edu.ar	2091306
Juan Manuel Prividera	jprividera@frba.utn.edu.ar	2096535
Elías Martín Mouesca	emouesca@frba.utn.edu.ar	2096389
Felipe Russo	ferusso@frba.utn.edu.ar	2088228
Francisco Mosquera	fmosquera@frba.utn.edu.ar	2038870

Colaboradores

Usuarios

- Contexto
    - Al finalizar la carga masiva de datos se nos pide que enviemos a los colaboradores las credenciales de acceso, si es que no tenían ya.
- Conclusión
    - Eso nos generó la clase usuario que tiene un nombre de usuario y contraseña. Además cada usuario tiene un rol que de momento serían colaborador jurídico, colaborador humano y administrador, y cada uno tendrá diferentes permisos dependiendo de las acciones que pueda hacer en el sistema.

Puntos de Reconocimiento

- Contexto
    - Los colaboradores reciben puntos cada vez que colaboran y cada colaboración da puntos de una forma distinta, cada una tiene un método que calcula los puntos que da al colaborador.
- Propuestas
    - Crear una clase Calculador de Reconocimiento, esta clase tendría un método que dado un colaborador devuelve los puntos que tiene en función de todas las colaboraciones que haya hecho, tendría que acceder a los datos de las colaboraciones persistidas.
    - A medida que se crean las colaboraciones se suman al colaborador los puntos que esta genera.
- Conclusión
    - Decidimos implementar la primera opción ya que la colaboración de adherir heladera aporta puntos de reconocimiento en función de los meses que estuvo activa. Hicimos que el colaborador tenga guardados los puntos que fue gastando así cada vez que se quiera saber los puntos que tiene se calculan los totales en ese momento y le restamos los gastados.

Medios de Contacto

- Contexto
    - Contamos con colaboradores los cuales tienen uno o varios métodos de contacto en caso de que se necesite contactarlos, por ejemplo en el caso de la carga masiva de datos. Además contamos con técnicos de los cuales también conocemos sus métodos de contacto
- Conclusión
    - Representamos el medio de contacto tanto para el colaborador como para el técnico por medio del patrón de diseño Strategy ya que para cada uno de los medios de contacto tenemos una estrategia distinta de notificación/aviso.

Tipo de colaborador

- Contexto
    - Remodelando los requerimientos de la primera entrega nos encontramos con el problema del modelado del tipo de colaborador y el acceso a cada colaboración que le corresponde a cada uno. Para esto planteamos:
- Propuestas:
    - El tipo de colaborador se representa con un enum, JURIDICO O HUMANO, y este es atributo de la clase Colaborador. A partir de este se decide a cuáles colaboraciones tiene acceso el Colaborador.
    - El tipo de colaborador se representa con un enum, HUMANO, GUBERNAMENTAL, EMPRESA, ONG o INSTITUCION, y este es atributo de la clase Colaborador. A partir de este se decide a cuáles colaboraciones tiene acceso el Colaborador, en el caso del jurídico se verifica que no sea HUMANO
    - El tipo de colaborador se representa a partir del rol del Usuario correspondiente a la clase Colaborador y además se cuenta con un atributo tipoJuridico que corresponde a un enum, GUBERNAMENTAL, EMPRESA, ONG o INSTITUCION. A partir del rol se decide a cuáles colaboraciones tiene acceso el Colaborador.
- Conclusión
    - La solución elegida es la última ya que presenta adecuación funcional al cubrir el requerimiento de representar el tipo de jurídico y presenta extensibilidad al permitir incorporar nuevos roles instanciando la clase, sin modificar el código.

Colaboraciones

Contexto General

Cada colaboración contiene los datos del colaborador que la generó, y la fecha en la que se generó. El colaboracionController se encarga de sumar los puntos de reconocimiento al mismo tiempo que genera la colaboración. Implementando el patrón Strategy, la forma de resolver el cálculo del reconocimiento de cada colaboración, es propio de cada una, esto se aprovecha para tratarlo de forma polimórfica. Además, cada colaboración tiene un factor de reconocimiento, que modifica el reconocimiento que agrega al colaborador.

Dar de alta persona vulnerable

El colaboracionController, luego de instanciar la persona vulnerable, se encarga de settear el atributo ‘personaVulnerable’ de la tarjeta, en función de los datos utilizados para instanciar la colaboración.

Distribuir vianda

El colaboracionController se encarga de modificar el atributo ‘heladera’ de las viandas afectadas, en función de los atributos de la instancia de la colaboración. La cantidad de viandas que se distribuyen se conoce a través del método cantViandasAfectadas.

Ofertas

No se modelan los productos o servicios ofrecidos por los colaboradores, porque nuestro dominio no contempla estos objetos. En cambio, se utiliza la imagen para mostrar la información pertinente de la oferta.

Heladeras

Estado

- Contexto
    - Las heladeras pueden estar activas o inactivas, de momento, dependiendo de si sufren algún desperfecto que haga que dejen de funcionar correctamente.
- Conclusión
    - Implementamos un patrón state para el estado de la heladera, en el cual cada estado tiene el método notificar, que notifica a los colaboradores de que una heladera está inactiva o si ya vuelve a estar activa. Esto también nos da extensibilidad a si en algún futuro las heladeras pasan a tener otro estado como podría ser “En reparación”.

Sensor

- Contexto
    - Las heladeras cuentan con sensores, de temperatura y de movimiento, y estos deben generar alertas si tienen una medición fuera de los parámetros establecidos.
- Conclusión
    - Dado que los sensores son un dispositivo físico en las heladeras nos limitamos a modelar las mediciones que estos generan y cada medición tiene un tipo que corresponde al sensor que la generó, además de otros datos como la fecha, el valor de la medición y su gravedad.

Administrador de alertas

- Contexto
    - Se requiere cambiar el estado de la heladera así como registrar el desperfecto correspondiente al fallo.
- Conclusión
    - Contamos con una clase AdministradorDeAlertas que se encarga de responder a la alerta en caso de que se genere una Medición la cual no está dentro de los parámetros esperados.

Modelo

- Contexto
    - Las heladeras tienen un modelo en el que se definen algunos parámetros propios del mismo como la temperatura máxima y mínima aceptables y la capacidad que tiene la heladera.
- Conclusión
    - Creamos una clase modelo y cada heladera tiene un modelo.

Vianda

Heladera

Definimos que la vianda conozca la heladera en la que está que contiene. Esto facilita la implementación de los retiros, y ganamos cohesión porque es la vianda la encargada de retirarse o ingresarse a las heladeras.

Mover

Este métodos funciona como setter del atributo heladera de la vianda. Si se ingresa a una heladera cambia el valor de la heladera, a la que fue ingresada. Mientras que, para retirar, se elimina dicha vianda del sistema. El RetiroController se encarga de liberar dicho espacio en memoria (posteriormente se encargará de eliminarla de la base de datos).

Persona vulnerable

Tarjeta

- Contexto
    - Las tarjetas le van a permitir a las personas vulnerables retirar viandas de la heladera. Estos tienen un uso límite dependiendo de la cantidad de hijos menores de edad que tenga. También vinculan a una persona vulnerable con el colaborador que la dio de alta en el sistema.
- Conclusión
    - Tiene un método limiteDeUsoDiario que se encarga de calcular la cantidad máxima de retiros que puede realizar la persona en el mismo día (teniendo en cuenta los atributos ‘limiteDeUsoDiario y retirosAdicionalesPorHijo). Este límite diario es luego utilizado por puedeRetirar para, en función de los retiros realizados en la fecha, definir si la tarjeta puede o no realizar más retiros durante ese día. Esto permite proteger al sistema contra errores del usuario ante un posible intento de retiro no autorizado.

Retiro

El retiro almacena consigo la fecha de cuando se realizó, la heladera de donde se sacan las viandas y la tarjeta que efectuó el retiro. Nos da trazabilidad sobre los retiros que realice una persona vulnerable.
RetiroController se encargará de modificar el atributo ‘heladera’ de las viandas afectadas por el retiro y de agregar la fecha del retiro a ‘fechasRetiros’ de la tarjeta.

Carga masiva de datos

- Contexto
    - Dado un archivo .csv con los datos de colaboraciones del antiguo sistema tenemos que leerlo y guardar esos datos.
- Conclusión
    - Lo resolvimos haciendo una clase LectorCSV que se encargaría de esta funcionalidad garantizando la cohesión respeto a la carga de datos. Esta clase utiliza una librería externa OpenCSV que nos facilita la lectura del archivo de una manera más eficiente que si la implemtaramos nosotros. También utilizamos una clase ColaboracionBuilder para crear colaboraciones de forma polimórfica a medida que leemos el archivo. Esta clase tiene un método que recibe el tipo de colaboración y algunos parámetros y nos devuelve una colaboración. De esta forma está desacoplada la lectura del archivo de las distintas colaboraciones que tengamos en el sistema.

Consumo API

- Contexto
    - Necesitamos consumir una API Rest que nos de algunos puntos geográficos recomendados para colocar heladeras en base a parámetros como la densidad demográfica, movimiento circulatorio, etc.
- Conclusión
    - Como el desarrollo de la API lo hace alguien externo a nuestro sistema vamos a implementar un patrón adapter que nos permite avanzar con nuestro diseño sin conocer la implementación final de la API. De esta forma nos desacoplamos de la implementación de la API haciendo más modular nuestro sistema.