Justificaciones de Diseño Entrega 2

Grupo 07

Sistema para la Mejora del Acceso Alimentario en Contextos de Vulnerabilidad Socioeconómica

Trabajo Práctico Anual Integrador

-2024-

Requerimientos detallados

1. Se debe permitir que los colaboradores registren a las personas en situación vulnerable.
2. Se debe permitir asegurar trazabilidad y auditoría en el manejo de las diferentes tarjetas.
3. Se debe permitir conocer el estado en tiempo real de cada heladera.
4. Se debe permitir el alta,baja, modificación de técnicos.
5. Se debe permitir solicitar la recomendación de puntos de colocación de las heladeras.
6. Se debe permitir calcular el puntaje de los colaboradores y realizar el canje por los productos y/o servicios disponibles.
7. Se debe permitir que las empresas ofrezcan productos y/o servicios como una nueva forma de colaboración.
8. Se debe permitir la carga masiva de colaboraciones.

Justificaciones

1. Para cumplir con el requerimiento en donde los colaboradores puedan registrar a personas en situación vulnerable, se decidió implementar la clase AltaPersonaVulnerable de tipo colaboración, la cual contiene a la PersonaVulnerable que se está dando de alta, la Tarjeta que se le es entregada a esta persona y la fecha en la que se realizó la transacción. Ver Tarjeta

2. Para asegurar la trazabilidad en el manejo de las diferentes Tarjetas se implementó la fecha de adjudicación (fechaAdjudicacion) en la clase Tarjeta permite rastrear cuándo se asignó la tarjeta a una persona vulnerable. El atributo colaborador en PersonaVulnerable permite identificar qué colaborador realizó la asignación de la tarjeta, proporcionando trazabilidad sobre quién realizó la acción. Los atributos frecuenciaUso y cantidadUsosDia en la clase Tarjeta permiten monitorear y auditar la frecuencia de uso de cada tarjeta, ayudando a detectar patrones de uso inusuales. La fecha de uso (fechaUso) en UsoTarjeta permite registrar cada instancia en la que se usó la tarjeta, proporcionando un historial de uso.

   Por otro lado, en términos de auditoría, utilizando los atributos de frecuencia y cantidad de usos, podemos generar reportes para auditar el uso de las tarjetas, asegurando que no se excedan los límites establecidos. La fechaUso en UsoTarjeta permite generar reportes sobre cuándo y con qué frecuencia se utilizan las tarjetas. Además, fechaRegistro en PersonaVulnerable y fechaAdjudicacion en Tarjeta permiten auditar el proceso de asignación y registro de las tarjetas.

3. Para conocer en tiempo real el estado de una Heladera, teniendo en cuenta los Sensores de Movimiento y Temperatura que las mismas poseen, se decidió implementar dos clases:
    - SensorTemperatura que cada 5 minutos efectúa el método registrarTemperatura(temp: Float) , seteando de esta forma el atributo ultimaTempRegistrada para que el sistema la conozca
    - SensorMovimiento: con un método alertarRobo() que emite una alerta avisando que la heladera está siendo robada.

    Además, se le agregaron los atributos tempMin y tempMax a Heladera, para que estas sean configurables por el usuario

4. Para el alta, baja, modificación de técnicos se decidió incorporar la clase Técnico, con su respectivo nombre, apellido, documento, etc., sus medios de contacto, en la clase medioDeContacto, la cual posee un CanalContacto (enum) y el área de cobertura de cada técnico, basada en una Dirección (punto de referencia) y un radioDeCoberturaEnKM en base a este punto.

5. Para solicitar la recomendación de puntos de colocación de las heladeras desarrollamos la clase RecomendadorDePuntosDeColocacion el cual recomendarUbicacion() en base a un punto (una ubicacion) y un radio deseado.

6. Cada vez que un colaborador realiza una Colaboración, la misma utiliza la interfaz CalculadorDePuntos en base al tipo de colaboración que se efectuó y, estos puntos, se le suman al Colaborador correspondiente.

    Con estos puntos sumados, el colaborador será capaz de realizar el canje por los productos y/o servicios disponibles, estos productos se encuentran disponibles en OfertaProducto, una clase de tipo Colaboración en donde las empresas/colaboradores pueden ofrecer sus productos, de distintas categorías, a cambio de puntos ganados. El canje se incorporó en la clase CanjeProducto, el cual tiene un Colaborador quien es el que quiere canjear la oferta, una fecha y una OfertaProducto, en donde se consultará si los puntos del colaborador son suficientes para canjear el producto.

7. Es implementada y está explicada en el punto anterior.

8. Para la carga masiva de colaboradores mediante la importación de archivos CSV para simplificar la carga de datos se decidió implementar el patrón de diseño Adapter, el cual

por ejemplo permite que el sistema existente, que puede no haber sido diseñado para leer archivos CSV, sea compatible con esta nueva funcionalidad sin necesidad de modificar su estructura interna (Compatibiidad). Utilizando el Adapter ganamos en Flexibilidad a la hora de cambiar de la biblioteca o método utilizado para la lectura de archivos CSV sin afectar al resto del sistema, por ejemplo si cambia el tipo de archivo a leer. Para esto se incorporó la interfaz CSVReaderAdapter para leer los datos desde un archivo CSV y cargar las colaboraciones (readCSV al cual se le pasa la ubicación de un archivo por parámetro).

Teniendo en cuenta que una vez procesado el archivo se le deberá enviar un mail a aquellos colaboradores que no tenían usuario en el sistema, también se decidió desarrollar el patrón de diseño Adapter para el envío de los mails ya que similar al caso del CSV, el Adapter permite integrar diferentes servicios de envío de correos electrónicos (en nuestro caso utilizamos el servicio de SendGrid) con el sistema sin necesidad de cambiar su lógica interna (Compatibilidad). Por otro lado, si en el futuro decidimos cambiar de proveedor de servicios de correos electrónicos solo necesitaremos cambiar el adapter y no todo el código relacionado con el envío de correos (Flexibilidad). Por último se podría decir que facilita el mantenimiento y pruebas del sistema, ya que la lógica de envío de correos está encapsulada y puede ser fácilmente intercambiada o simulada durante las pruebas, en nuestro caso utilizando Mockito.
Todo esto se encuentra implementado en la interfaz MailSenderAdapter.