

Trabajo Práctico Anual Integrador

**para**

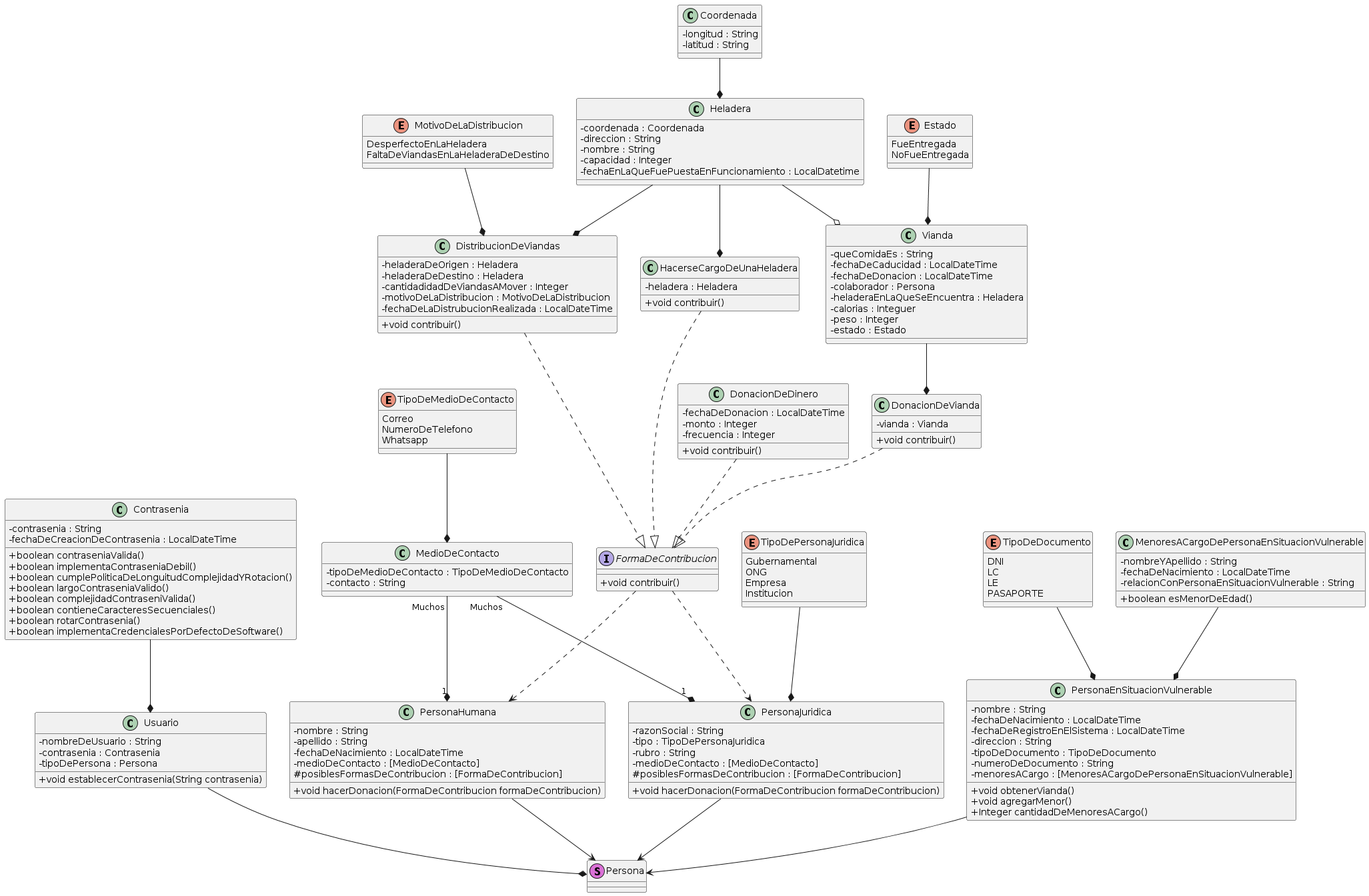
Diseño de Sistemas

**Preparado por**

|  |  |
| --- | --- |
| **GRUPO N° 14** | |
| **Nombre y Apellido** | **Legajo** |
| Alvarez Kalustian, Sahira Aylén. | 175.584-5 |
| Deza, Alexia. | 203.562-5 |
| Guardines, Brenda. | 172.658-4 |
| Juncos, Fernando. | 152.342-9 |
| Porcel, Gustavo. | 202.760-4 |

**ENTREGA 1: Modelado en Objetos Parte I**

**Diagrama de Clases:**



**Decisiones de Diseño:**

Elegimos crear una clase por donador (“PersonaHumana” y “PersonaJuridica”), ya que las mismas no comparten atributos. No implementamos una interfaz Donador, como pensamos inicialmente, ya que no compartían comportamiento. Ambas clases tienen el atributo “medioDeContacto”, que es una lista de clases bajo el mismo nombre, las mismas contienen un Enum de los “TipoDeMedioDeContacto” y su correspondiente contacto, decidimos desarrollarlo así, ya que esto nos permitiría un manejo más cómodo de los medios de contacto.

En la “PersonaJuridica” utilizamos un Enum para el tipo, ya que son opciones que no ejecutan nada y no son variables. El resto de los Enums se decidieron por los mismos motivos.

Creamos una clase Usuario, que contiene todos los datos de inicio de sesión, así como a su vez el tipo de persona a ingresar ya se un donador o una “PersonaEnSituacionVulnerable”, ya que ambos deben poder ingresar al sistema, cada uno para realizar su propia tarea. A su vez, el atributo “contrasenia” que pertenece al usuario es una clase “Contrasenia”, la cual tiene el String y la “fechaDeCreacionDeContrasenia” correspondiente a la misma, también posee el método “contraseniaValida”, que nos permite corroborar (al crear una nueva) que la misma sea apta. Decidimos unificar los recién mencionados atributos con el validador, para que la misma contraseña se valide y respetar las responsabilidades.

Para la “PersonaEnSituacionVulnerable” creamos una clase, y decidimos manejar la parte de los menores a cargo con una lista de “MenoresACargoDePersonaEnSituacionVulnerable” que es otra clase. La misma tiene los datos del menor (como “nombreYApellido”, “fechaDeNacimiento” y “relacionConPersonaEnSituacionVulnerable”, así mismo, incluimos un método “esMenorDeEdad” que se encarga de revisar lo propio, para que al modificar los menores a cargo se tenga en cuenta si lo son.

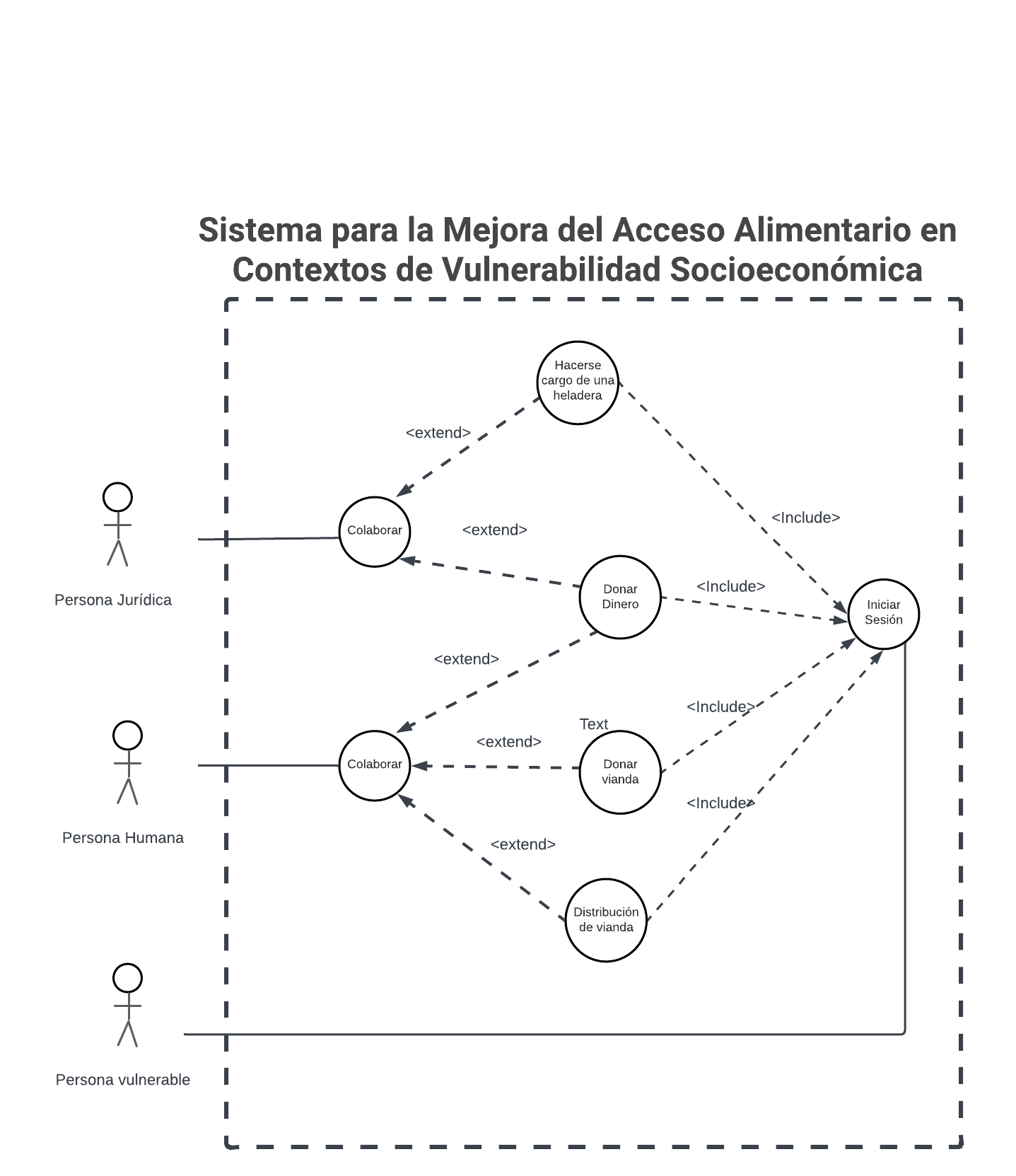
Para las clases “PersonaEnSituacionVulnerable”, “PersonaHumana” y “PersonaJuridica”, creamos un Stereotype con el objetivo de unificarlas para su uso en “tipoDePersona” dentro de Usuario. Elegimos esta entidad, ya que agrega información semántica adicional a los elementos en un modelo para indicar su rol o comportamiento específico.

Creamos una interfaz “FormaDeContribucion” para el uso del Patrón Strategy, ya que el mismo encapsula distintas formas de resolver el mismo problema en diferentes clases y eso es lo que estaríamos necesitando para manejar las donaciones. En cuanto a las “posiblesFormasDeContribucion” que maneja cada donador (ya que no todas coinciden) utilizamos una lista de “FormaDeContribucion” como atributo protegido en ambos casos. Y, dentro de la misma clase, majeamos las excepciones con dicha lista.

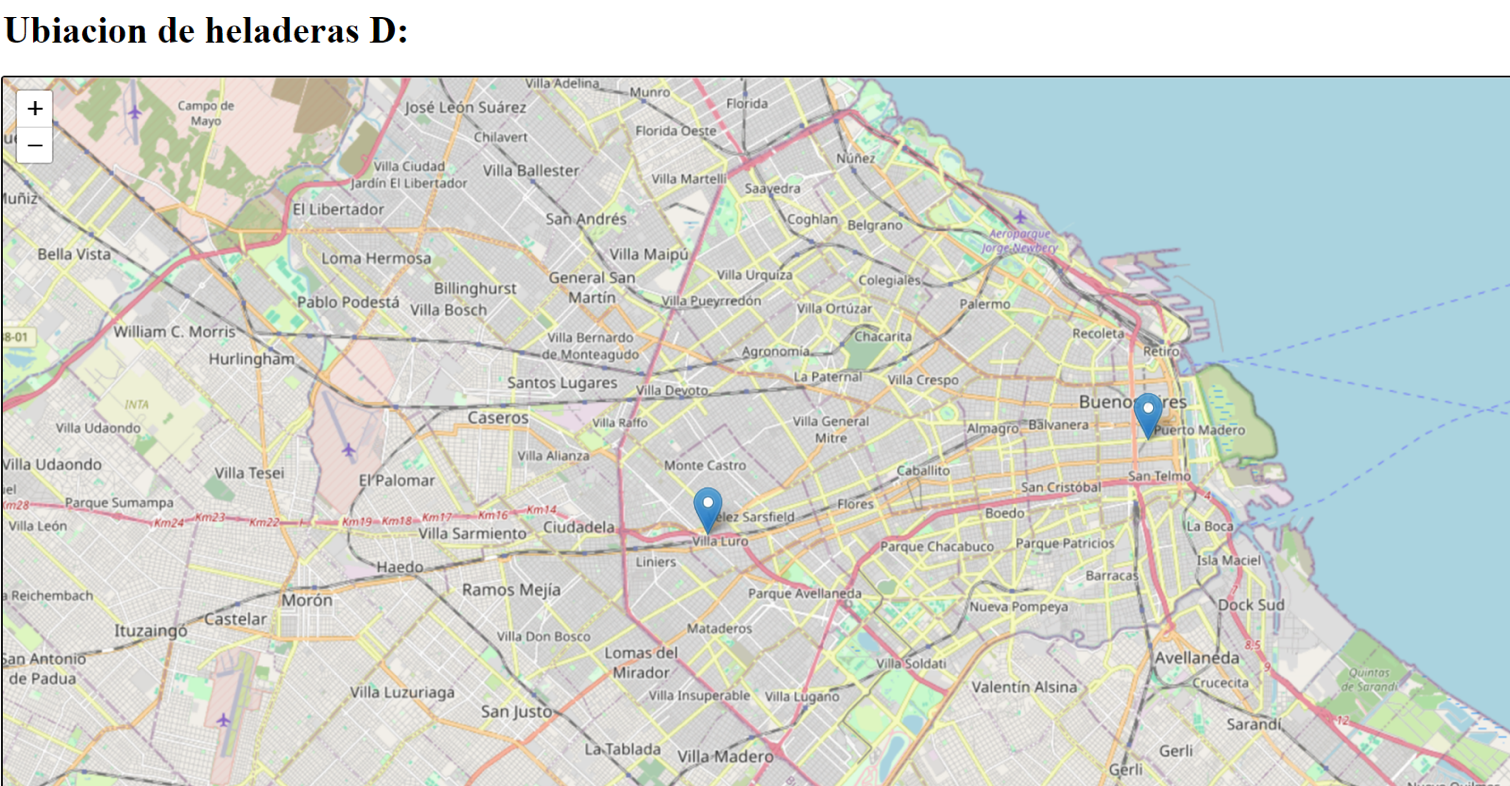
Respecto a la clase Vianda, la compusimos con DonacionDeVianda, ya que se la pasa como atributo y si no existiesen las viandas, no tendría sentido la donación de las mismas.

En cuanto a la clase Heladera, la agregamos a vianda, ya que consideramos que podrían haber heladeras vacías (por la existencia de DistribucionDeViandas, y sus motivoDeLaDistribucion). A su vez, creamos la clase Coordenada para la composición de las mismas en la Heladera.

**Diagrama de Casos de Uso:**



**Mapa Donde se Visualizan las Heladeras:**



Con respecto al mapa, investigamos las bibliotecas de JavaScript y, encontramos “Leaflet” la cual nos va a servir para incorporar los mapas interactivos para las futuras entregas utilizando coordenadas.

Dicha biblioteca permite tomar una sección del mapa de Google maps y mostrarlo mediante una etiqueta html en el centro de una pagina web.

Por medio de JavaScript, se utiliza un método propio de la biblioteca llamada “marker()” en el cual se ingresa las coordenadas de latitud y longitud, esto nos permite visualizarlo en el navegador.

Para futuras entregas planeamos utilizar el atributo “coordenada” de las heladeras para ubicar las mismas.

Como se puede ver arriba, utilizamos dos puntos aleatorios de ejemplo para demostrar que el mapa es funcional.