

ReCircular

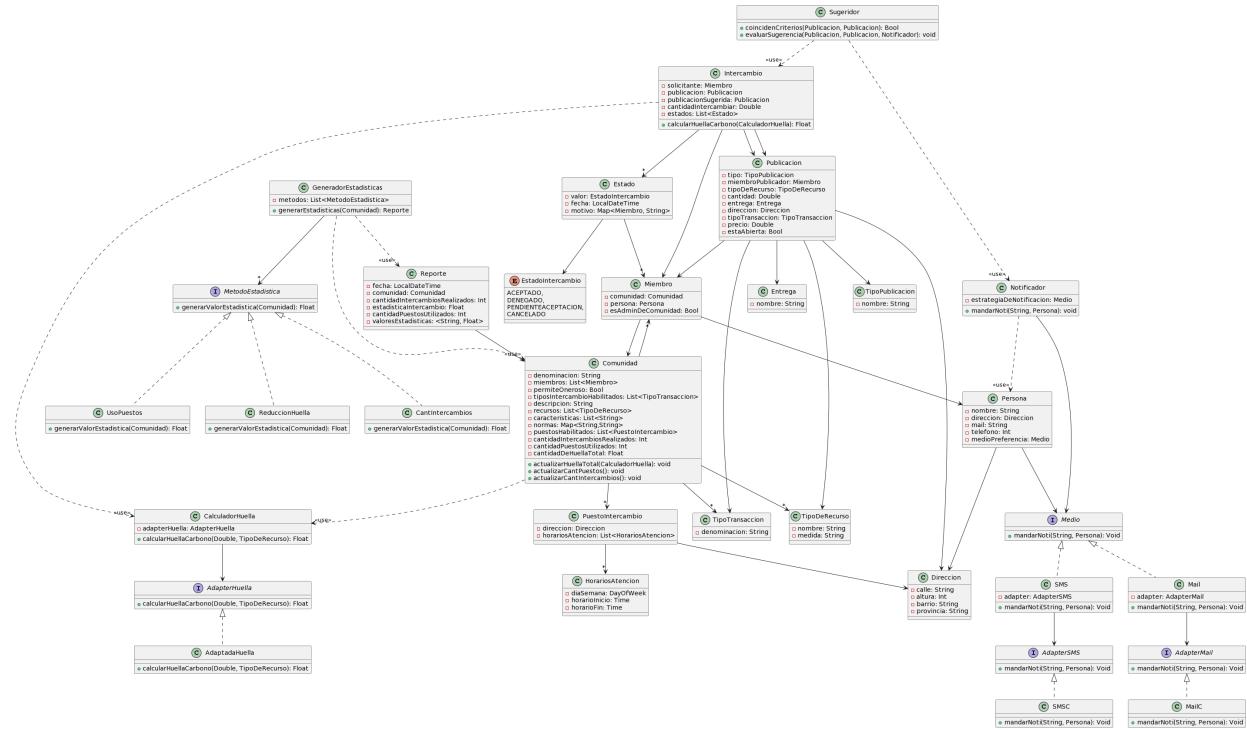
[Final DDS 20221217.pdf](#)

Clase de consulta: no hay

Hecho por: Facundo Piaggio y Chabela Lamas

Modelado de objetos

Diagrama de clases



Decisiones de diseño

Por lo que respecta a los patrones, en el presente diagrama se puede encontrar el Adapter y el Strategy. El primero, se puede observar en la implementación del medio de notificación y en el calculador de la huella de carbono (debido a que debemos integrarnos a una API REST). De esta manera, se gana cohesión en las clases adaptadas, y se delega la responsabilidad en objetos particulares, aumentando de esta manera la mantenibilidad al no tener acoplamiento entre la clase adaptada y el cliente. Luego, el patron Strategy se puede encontrar en la implementación del notificador y su medio como atributo, así como también en los métodos de generador de estadística. Por consiguiente, se gana cohesión, se mejora la mantenibilidad y se permite la extensibilidad para incorporar nuevos algoritmos o formas de realizar la notificación a la personas / estadísticas.

La clase Intercambio posee un atributo solicitante y publicacionSugerida que pueden ser nulleables. Por ejemplo, cuando se trata de una sugerencia, el solicitante se setea como null y cuando se trata de una instancia creada por un miembro respecto una publicacion, la publicacionSugerida va en null.

Las clases: tipoPublicacion y tipoRecurso, fueron creadas para ganar consistencia de datos, permitiendo aumentar la mantenibilidad y extensibilidad del diseño. Así como

también la clase Dirección, que permite solucionar el tema de los filtros de búsqueda por cercanía. Asimismo, esta entidad es de suma relevancia ya que tiene información que se debe tomar en cuenta para la entrega de los recursos.

Cabe aclarar que, la ejecución de los filtros de búsqueda se realiza en la capa de presentación. Por ello, a nivel objetos la publicación debe contar con toda esa información requerida para los filtros (Tipo Recurso, cantidad, dirección) tal como se mencionó anteriormente.

Por otro lado, la clase Publicación tiene un TipoTransaccion que setea el valor de precio en null si es gratuito o si es gratuito pero con envío a cargo del receptor, con el valor correspondiente del mismo una vez que se intercambie. De la misma manera, si es oneroso, se fija el atributo con el valor deseado. Con la dirección de entrega, sucede algo similar ya que si se realiza en el domicilio de quien publica, se setea con su respectiva dirección. Si se entrega en un cierto puesto fijo de intercambio, se seta con la dirección del puesto de intercambio y si es a convenir, se deja en null hasta que se llegue a un acuerdo. No obstante, en caso de que se quiera buscar los horarios de atención de los puestos fijos, esta información se podrá encontrar en alguna vista de la comunidad, tal como lo hace Mercado Libre. Es decir, que a nivel de la publicación solo se indicara cuál es el puesto que posee la dirección indicada y, luego, si la persona desea, puede acceder a los horarios de atención en una vista del front que enumere los puestos disponibles para la comunidad.

En adición, se busca lograr trazabilidad con los cambios de estados en los intercambios. Cada estado tiene un valor que es un enumerado, una fecha y un motivo que se completará en caso de que se cancele la entrega.

Por lo que refiere al sugeridor, en el siguiente punto se desarrolla su funcionamiento pero, algo para destacar es que esta con el Notificador, el Calculador de huella y el Generador de estadística, fueron generadas para aumentar la cohesión y disminuir el acoplamiento entre las clases. La creación de las instancias de las mismas es realizada en la capa de controladores.

Por último, el generador estadística crea un reporte con las estadísticas y los valores registrados (los que se encontraban calculados en la comunidad, los cuales se utilizaron para obtener la estadística) de huella de carbono, uso de puestos y cantidad de intercambios. Para realizar los cálculos, se obtienen los reportes anteriores del repositorio y se realizan las diferencias entre los valores de las respectivas fechas. Cabe aclarar que, se considera que los valores que se utilizan de la comunidad para

este componente se encuentran precalculados en la entidad para mejorar la performance de esta función. De esta manera, se actualizan mediante el uso de una Cron Task.

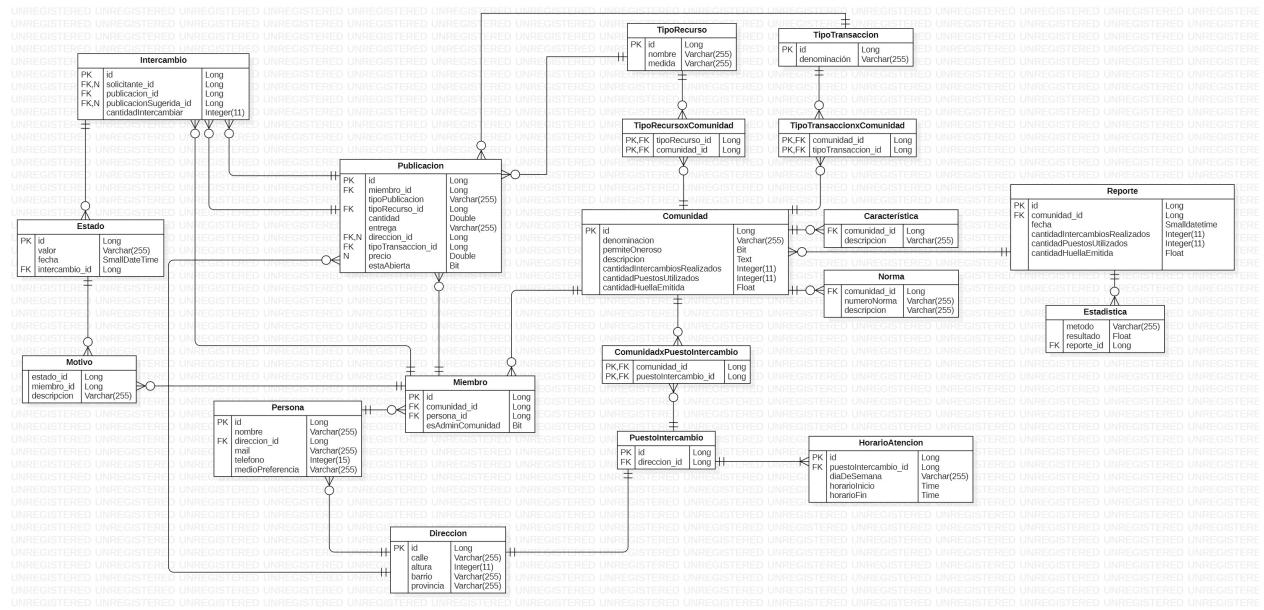
Nosotros consideramos que las estadísticas son los porcentajes de aumento o disminución de los atributos mencionados frente a los del último reporte. A nivel de interfaz se podría recopilar los datos del repositorio de reportes y mostrar la información de manera más estética y usable. Cabe aclarar que, hicimos un map de <Tipo de método utilizado pasado a string, Resultado>. Se convierte el método de la estadística generada a un string. Esto permite que si en un futuro se quieren sumar nuevos métodos, se pueda lograrlo de una manera muy fácil.

Justificar en forma detallada cómo resuelve el siguiente requerimiento: “el sistema debe notificar automáticamente a las personas con publicaciones abiertas que coincidan (total o parcialmente) con los criterios de otra. Cuando una persona inicia y culmina exitosamente un intercambio con otra que haya sido sugerida, las dos publicaciones serán cerradas a la par”.

Se puede utilizar una Cron Task que, cada cierto tiempo, utilice el sugeridor, analizando las publicaciones existentes activas de cada comunidad. El sugeridor, en caso de que coincidan dos publicaciones, llama al notificador y este envía un mensaje al medio de preferencia de la persona. De esta manera, se crea una instancia de la clase Intercambio donde se setea el estado a PENDIENTEACEPTACION. En caso de que ambas partes acepten dicha propuesta, el estado pasa a ACEPTADO, permitiendo que en la capa de controllers se modifique el boolean de estaAbierta de ambas publicaciones a False.

Persistencia

Modelo de datos



Decisiones de diseño

En primer lugar, se utiliza un converter para el tipo de entrega de la publicación, el medio de preferencia de la persona y el valor del estado debido a que consideramos que no son entidades que requieran ser persistidas en una tabla.

En segundo lugar, se realiza un element collection para las características y normas de una comunidad, así como también para los motivos de un estado.

Entidades necesarias a persistir

- Dirección
- PuestoIntercambio
- HorarioAtención
- Persona
- Miembro
- Comunidad
- Publicación
- Estado
- Intercambio

- Reporte
- TipoTransacción
- TipoRecurso

Impedance mismatch

Tipo de dato objetos	Tipo de dato relacional
LocalDateTime	SMALLDATETIME
String	VARCHAR(255)
Int	INTEGER(11)
Bool	BIT
String	TEXT
DayOfWeek	VARCHAR(255)

Respecto al impedance mismatch de Identidad, tuvimos que agregar un campo "id" (clave subrogada) para identificar únicamente a cada una de nuestras entidades.

Modelo desnormalizado

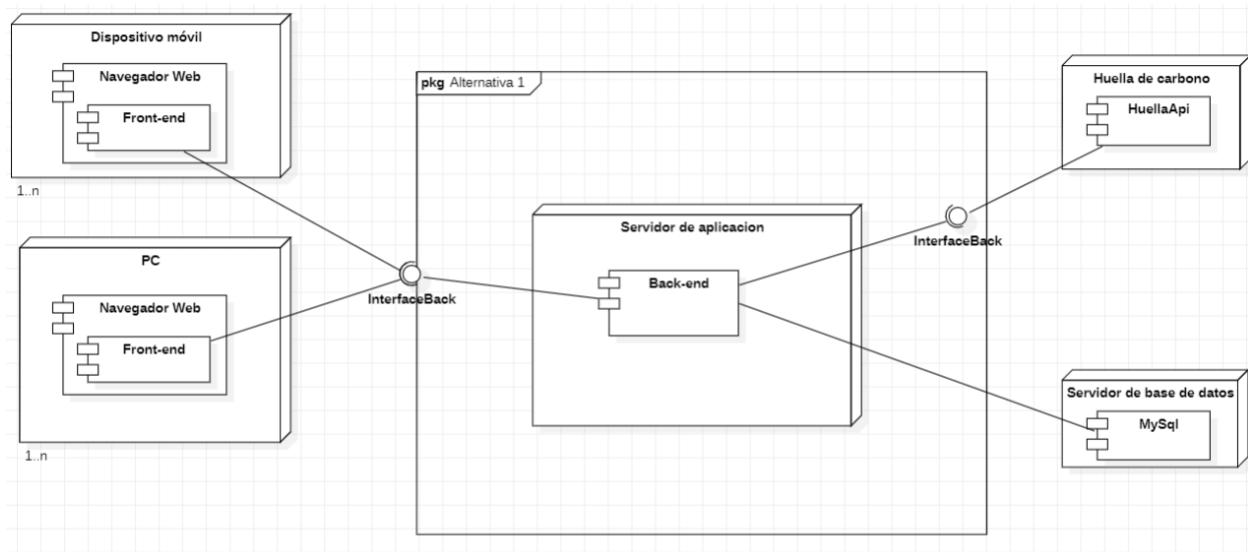
Se decide desnormalizar los valores utilizados para la estadística en el reporte para mantener la consistencia de datos.

Arquitectura

Realizar una propuesta de arquitectura y documentarla utilizando un diagrama de despliegue, que dé respuesta a cada uno de los siguientes requerimientos:

1.RECircular debe poder ser accedido tanto desde dispositivos móviles recientes como computadoras de escritorio, privilegiando una experiencia fluida y una interfaz usable. Además, debe minimizar la transferencia de datos, para que la aplicación funcione aún con baja conectividad.

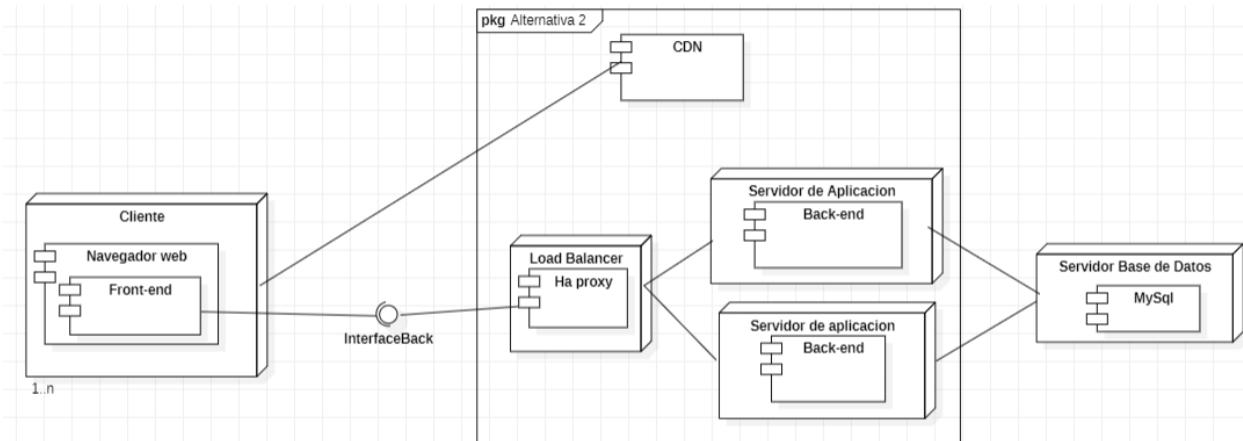
Debido a que se busca priorizar la experiencia fluida, permitiendo que la aplicación funcione aún con baja conectividad, se decide usar un SPA, un tipo de aplicación web donde todas las pantallas las muestras en la misma página, sin recargar el navegador. La primera request es la más pesada. De la misma manera, los dispositivos móviles y las computadoras pueden acceder al ReCircular mediante un navegador web. Por último, se implementa un cliente pesado, logrando que la lógica quede del lado del cliente y que no exista una gran cantidad de consumo de procesador y de memoria RAM del servidor.



2.RECircular debe estar disponible las 24 horas del día, los 365 días del año, minimizando las variaciones de tiempos de respuesta en función de la carga del sistema.

Se utiliza un cliente pesado, no por lo que enuncia el requerimiento sino porque consideramos que es más seguro al usar tokens, el más performante, al no requerir la misma cantidad de consumo de procesador y de memoria RAM del servidor al usar JSON y tener la lógica del lado del cliente. Asimismo, para cumplir con el enunciado, se opta por usar una CDN, que permite mejorar los tiempos de carga del sitio web, aumentando la disponibilidad de contenido y de redundancia, al tener un grupo de servidores distribuidos geográficamente que trabaja juntos para ofrecer una entrega rápida de contenido de Internet. Por último, también decidimos escalar horizontalmente,

implementando un Load Balancer, priorizando más la disponibilidad y performance del sistema en cuestión.



3. Sabemos que el API del sistema de cálculo de huella de carbono puede demorar bastante en responder, o incluso estar periódicamente fuera de servicio. Sin embargo, esta situación no debe afectar a los tiempos de respuesta ni a la disponibilidad de RECircular.

Debido que se busca evitar que se genere un cuello de botella con el servicio integrado del cálculo de la huella de carbono, se opta por utilizar una cola de mensajes con el fin de no saturar el sistema. Por consiguiente, se realiza el proceso de manera asíncrona, donde se coloquen en una cola los intercambios los cuales se deben calcular la huella de carbono emitida y donde luego, se la retire de la misma cuando el servicio se encuentre disponible.

