Decisiones de la entrega N°3:

El establecimiento ahora conoce la entidad a la que pertenece ya que necesitamos acceder a ese dato para poder generar uno de los rankings.

Los miembros solo pueden tener una comunidad
Esto es así porque de otra manera el miembro podría tener un rol distinto para cada comunidad (observador o afectado), teniendo que generar clases intermedias que consideren este caso resultando en una solución muy compleja

Los estados de los incidentes representan lo mismo que el estado de un servicio
Dado que se considera que si existe una incidencia sobre un servicio y esta se encuentra abierta, el servicio ya no se puede operar con normalidad, decidimos interpretar el estado del mismo, a través de la incidencia.

Las prestaciones de servicio ahora tienen un establecimiento
Esto lo decidimos dado que necesitábamos conocer la localización de un incidente. Para ello consultaremos la localización del establecimiento en el que se encuentra dicho incidente.

Las notificaciones se hacen a través de una clase Notificador que tiene la responsabilidad de revisar el medio y la estrategia de notificación del miembro y según eso notifica de esa forma. El notificador será instanciado en la capa de controladores y debería poder enviar una notificación de manera sincrónica o asincrónica ante un determinado evento.

Las clases MailSender y whatsappSender se integran con javaxMail y twillio respectivamente y serán utilizadas por el notificador como medio de envío para las notificaciones.

Creamos una interfaz común llamada Notificable para los distintos tipos de notificaciones. Esta interfaz común es la que implementaran los distintos eventos que tenemos hasta el momento, y tiene los métodos para construir un mensaje o un asunto dado el tipo de notificación.

Las notificaciones son 3 clases, que representan los eventos por los cuales el notificador envía la notificación a los miembros. Estas son: “NuevoIncidente”, “RevisarIncidente”, “CierreIncidente”. Son clases que tienen la responsabilidad de construir un mensaje y dárselo al notificador para que este se lo envíe de una manera u otra al receptor.

Al plantear las notificaciones de esta manera, implementando una interfaz común, ganamos en testeabilidad, cohesión, pues la responsabilidad de los distintos tipos de mensajes está en cada una y además delegamos responsabilidades que podrían tener erróneamente otra clase.

Las notificaciones sincrónicas se envían en el momento que sucede un evento.

Al abrir un nuevo incidente y este ser agregado en la comunidad del miembro notificante, se notificará a cada miembro de dicha comunidad.
Al cerrar un incidente, se notificará dicho cierre a cada miembro de la comunidad del miembro que notifica el cierre.
Y en caso de que un miembro cambie su localización, el sistema podría detectar que se encuentra cerca de un incidente, y en cuyo casa se le solicitará que vaya a revisarlo

Tomamos como que un miembro está cerca de la localización de un incidente, cuando la localización del miembro es literalmente la misma que la del incidente, esto lo hicimos ya que a fines prácticos simplifica muchas cosas. Si no, deberíamos meternos con coordenadas, distancias mínimas y cosas que no están tan buenas.

Los incidentes a la hora de reportarse, pueden tener una observación, que en este caso nosotros la implementamos como un atributo “descripción”, que sería la “general” del incidente, por decirlo de alguna manera. Por otro lado, tenemos lo que son las observaciones que realizan otros miembros sobre un incidente ya abierto. Esto no es más que un atributo, una lista de Observación (una clase que tiene un notificante, una descripción y una fecha), que se le asigna a ésta en forma de clase con el principal objetivo de no perder trazabilidad y formar una especie de historial de versiones de las distintas etapas por las que pasó un Incidente.