TPA Tercera Entrega
Equipo 10

UTN.BA
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DISEÑO DE SISTEMAS

Alcance

La entrega comprende:

- Apertura de Incidentes sobre Prestaciones de Servicios
- Cierre de Incidentes
- Envío de notificaciones
    - Por medios:
        - Email
        - WhatsApp
    - Por eventos de:
        - Apertura de incidentes
        - Cierre de incidentes
        - Sugerencia de revisión de incidentes
    - Según forma:
        - Cuando suceden (instantáneas)
        - Sin apuros (diferidas en el tiempo)
- Ranking de Incidentes según los tres criterios
- Informes para entidades prestadoras/organismos de control de Rankings
- Miembros Afectados/Observadores

Incidentes

Diseño del Incidente; apertura de incidentes; cierre de incidentes.

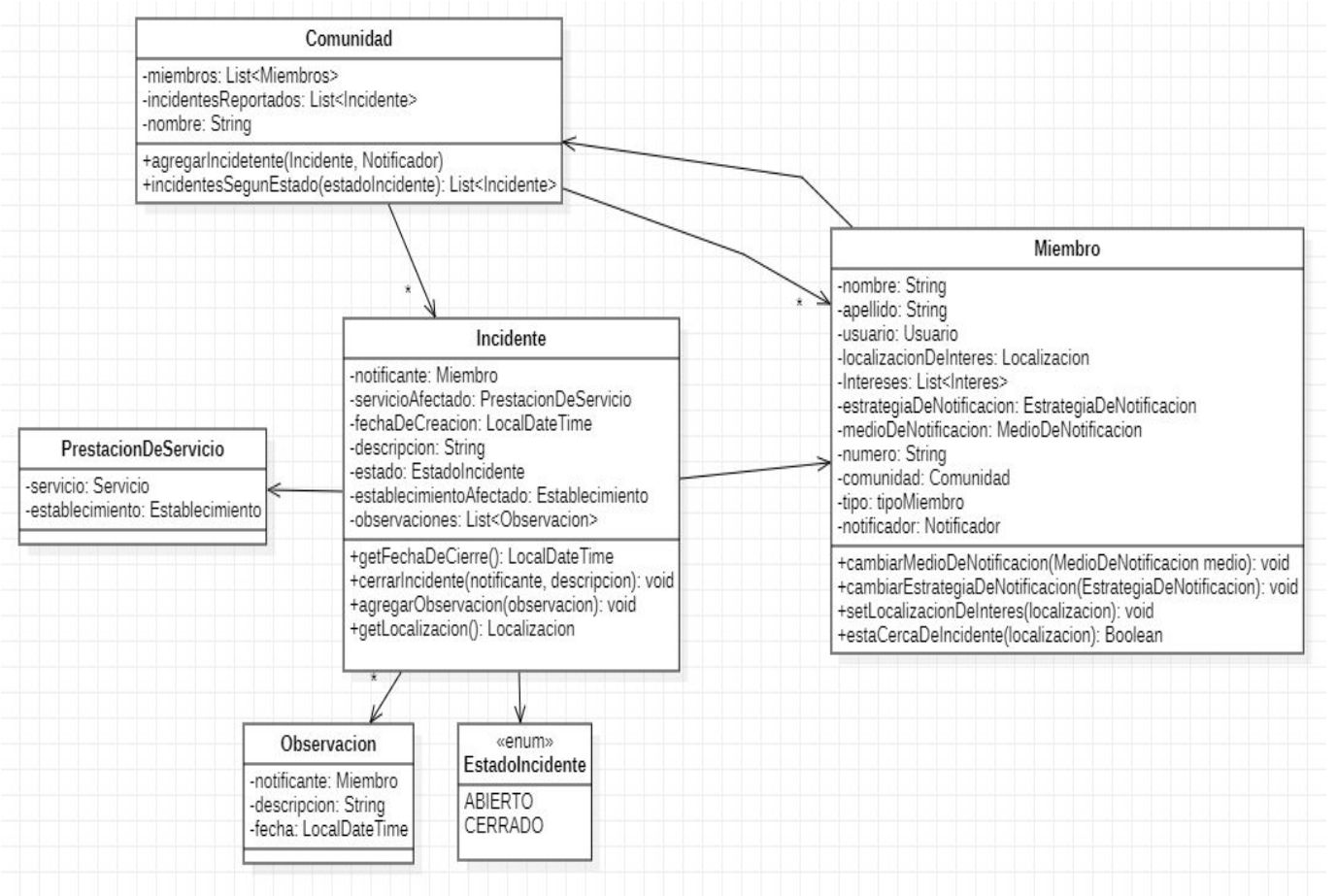
Incidentes – Diseño general

Los incidentes en el dominio se van a dar sobre una prestación de servicio, además tendrá un estado que representará el estado del servicio en el que se supone hay una incidencia. (Anteriormente teniamos un Boolean en la prestación como “disponible”).

Una comunidad va a tener multiples incidentes reportados que seran de interes para sus miembros, los cuales podrán consultar.

Además, para ganar trazabilidad, consideramos que el incidente tendra una lista de “Observacion”, una clase que agrupa:

- notificante: Miembro
- descripcion: String
- fecha: LocalDateTime



Incidentes – Apertura de incidentes

Todo miembro de alguna comunidad, será capaz de abrir un incidente. A la hora de hacerlo, el nuevo incidente será agregado a la lista de incidentes reportados de la comunidad del miembro notificante.

Pensando en como una persona, de manera fisica interactua con el sistema, pensamos en la apertura del incidente como un botón al que algún miembro al darle se instancia y solicita determinados datos para su armado.

En el momento de agregar el incidente en la lista de la comunidad, será cuando, en uso del notificador que discutiremos más adelante, se le notificará a todos sus miembros de esta novedad.

```
public void agregarIncidenteReportado(Incidente incidente, Notificador notificador) {
  this.incidentesReportados.add(incidente);
  NuevoIncidente nuevoIncidente = new NuevoIncidente();
  for (Miembro miembro: this.miembros){
    notificador.notificarAMiembro(miembro, nuevoIncidente, incidente);
  }
}
```

Incidentes – Cierre de incidentes

Para cerrar un incidente, como en la apertura, consideramos que será un botón que a la hora de presionarse para dar por cerrado un determinado incidente, se cargará una “Observación” (anteriormente mencionada) y se guardará en la lista de observaciones como última. De esta manera, podemos considerar la fecha de cierre como la fecha de última observación (Si el incidente se encuentra en estado CERRADO).

Además, nuevamente con el uso del notificador, se le enviará una notificación a cada uno de los miembros de la comunidad, a la que el miembro que está cerrando el incidente pertenezca.

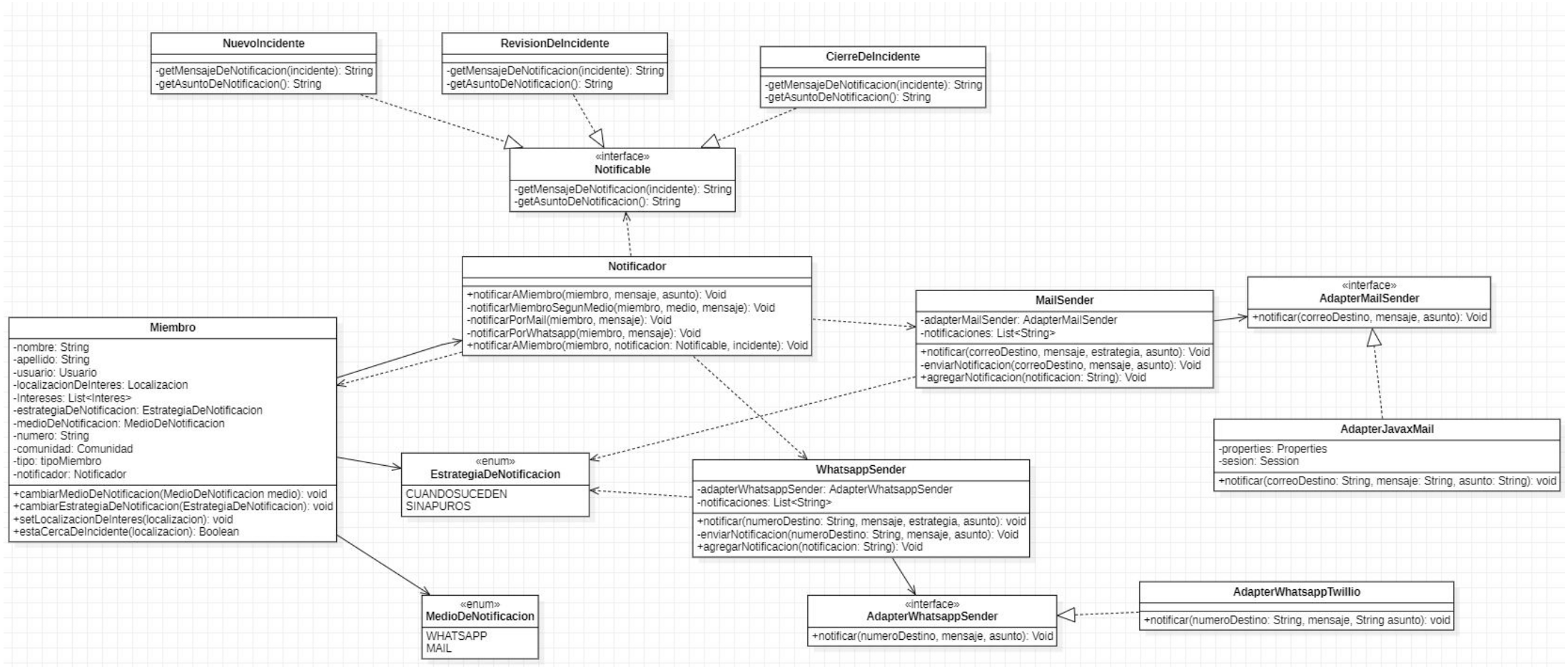
Incidente →

```
public void cerrarIncidente(Miembro notificante, String descripcion, Notificador notificador){
    this.estado = Estado.CERRADO;
    Observacion observacion = new Observacion(notificante, descripcion);
    agregarObservacion(observacion);
    for(Miembro miembro: notificante.getComunidad().getMiembros()){
        notificador.notificarAMiembro(miembro, new CierreIncidente(), incidente: this);
    }
}
```

Notificaciones

Diseño del Módulo de Notificaciones; notificaciones por los distintos medios; envío de notificaciones según eventos; envío de notificaciones en el momento y de forma diferida en el tiempo.

Notificaciones – Diseño general



Notificaciones – Medios de Notificaciones

El sistema cuenta con dos medios posibles para enviarle notificaciones a sus usuarios.

Mail y Whatsapp.

Para la representación de dicho medio, lo tomamos como un valor Enum que el Miembro podrá cambiar cuando sea que lo desee.

El notificador, clase responsable de notificar a los miembros, será el que consulte por el valor actual del miembro al que notificará, y allí decidirá que medio utilizar.

Conociendo el medio, utilizará la clase “MailSender” o “WhatsappSender” para realizar el envio de la notificacion. Clases que a su vez se adaptan con las bibliotecas “javaxMail” y “Twillio” respectivamente.

Notificaciones – Eventos que generan notificaciones

De momento el sistema cuenta con tres posibles eventos capaces de generar una notificacion.

Ante estos eventos decidimos crear una interfaz común llamada “Notificable”.

A su vez, tenemos un “Notificable” para cada evento.

- “NuevoIncidente”
- “CierreIncidente”
- “RevisionIncidente”

Para el caso de la Revision de un incidente, la notificación será enviada cuando el miembro cambie de localizacion. Allí el sistema revisará si ahora se encuentra cercano a un incidente, en cuyo caso le solicitara que vaya a revisarlo.

```
public void setLocalizacionDeInteres(Localizacion localizacion) {
  this.localizacionDeInteres = localizacion;
  if(this.comunidad != null){
    for (Incidente incidente : comunidad.incidentesSegunEstado(Estado.ABIERTO)) {
      if (this.estaCercaDeIncidente(incidente.getLocalizacion())) {
        notificador.notificarAMiembro( miembro: this, new RevisionIncidente(), incidente);
      }
    }
  }
}
```

Notificaciones – Envío sincrónico/asincrónico

Respecto a los eventos eventos sincrónicos, nosotros consideramos que sea el notificador que considere que estrategia tiene el miembro a la hora de notificarlo.

Existen dos estrategias. SinApuros y CuandoSuceden.

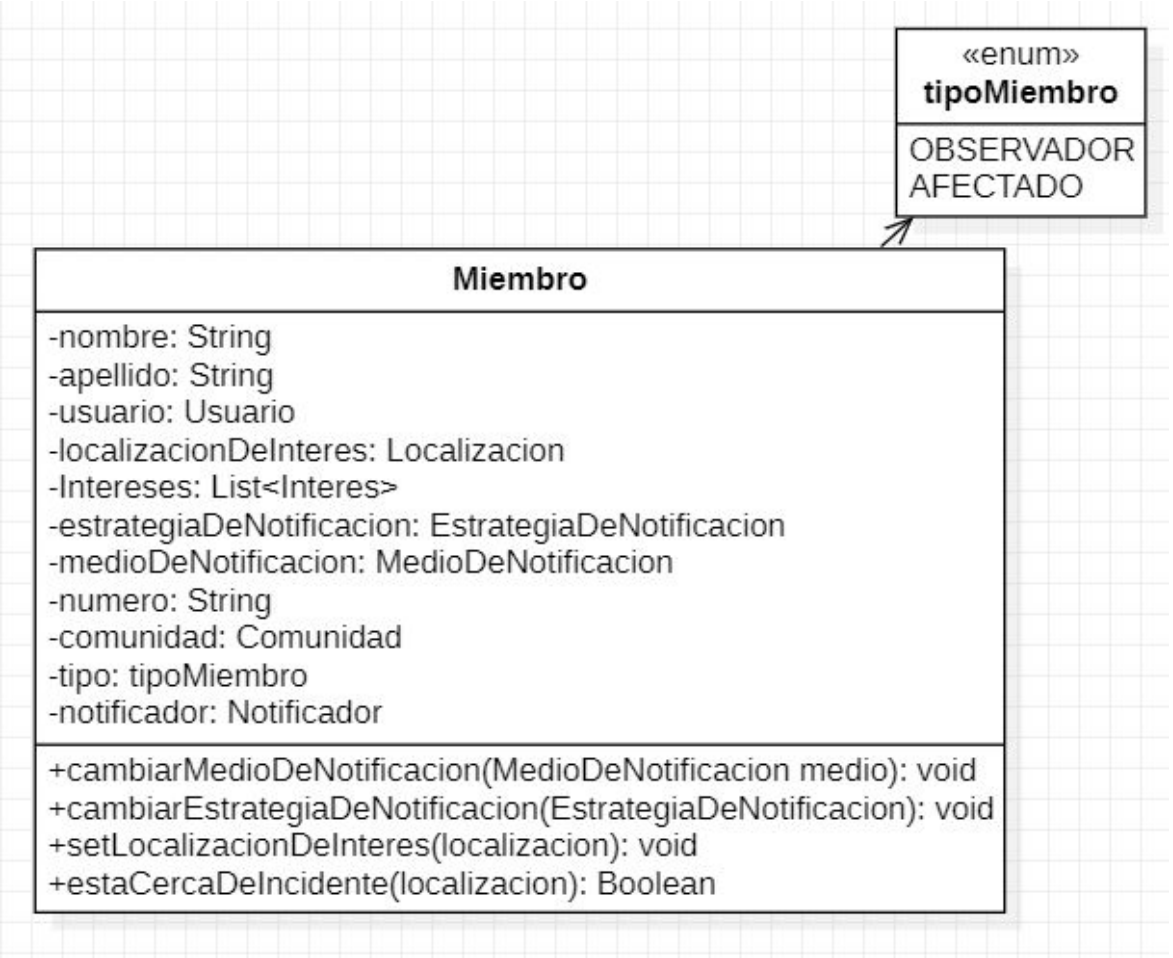
En caso de ser SinApuros, lo que pensamos fue que el notificador guarde en una lista la notificación que deba enviar, para que luego, cuando llegue el momento (un horario seteado por el miembro), a través de un cron, se envíe una única notificación con todas las notificaciones guardadas hasta ese momento.

Miembros Afectados/Observadores

Diseño de la condición de Afectado u Observador del Miembro

Afectado/Observador – Diseño general

Para simplificar el diseño decidimos que un usuario solo pueda ser Observador o Miembro para todos los servicios.



Ranking de Incidentes

Diseño del Módulo Rankeador de Incidentes; diferentes rankings solicitados.

Rankings – Diseño general

Es un componente que se ejecutará una vez a la semana que se encargará de generar las tablas de clasificaciones y las exportará utilizando el módulo exportador.

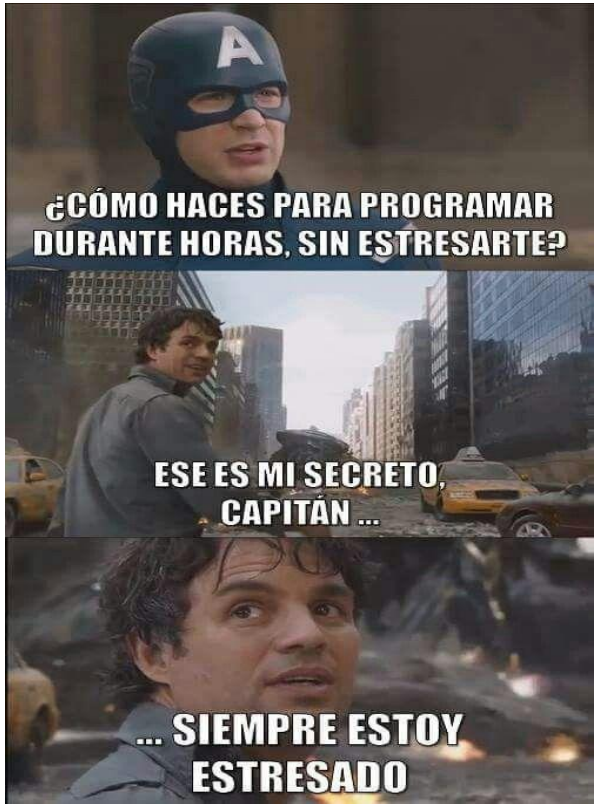
En una futura iteración, cuando nos integremos con una BBDD tendremos registro de todos los incidentes, con una consulta del tipo:
SELECT * FROM tabla WHERE fecha BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 WEEK)
AND DATE_SUB(CURDATE(), INTERVAL 1 SECOND);
Podremos obtener la lista de incidentes solo de la semana que deseemos generar el ranking.

Rankings – Entidades con mayor promedio de tiempo de cierre de incidentes

```
 public void generarRankingDeMayorPromedioDeCierre (RepositorioIncidentes repositorioIncidentes , Exportador exportador , RepositorioEntidades
repositorioEntidades , CalculadorDeTiempos calculadorDeTiempos) {
   Reporte reporte = new Reporte();
   List<Incidente> incidentesCerrados = repositorioIncidentes.getIncidentesCerrados()  ;
   List<Entidad> entidades = repositorioEntidades.getEntidades()  ;
   List<PromedioDuracionIncidente> promediosPorEntidad =  new ArrayList<>();
   for (Entidad entidad : entidades) {
     List<Long> horarios = incidentesCerrados.stream()
        .filter(incidente -> incidente.getEstablecimientoAfectado().getNombreEntidad() ==   entidad.getNombre())
        .map(incidente ->  calculadorDeTiempos .calcularTiempoEntreFechas(incidente.getFechaDeCierre() ,
incidente.getFechaDeCreacion()).toHours())
        .toList() ;
     Double suma = 0.0;
     for (Long horario : horarios) {
       suma += horario ;
     }
     Double promedio = suma / horarios.size() ;
     PromedioDuracionIncidente promedioDeEntidad =  new PromedioDuracionIncidente(entidad , promedio);
     promediosPorEntidad.add(promedioDeEntidad) ;
   }
   promediosPorEntidad.sort( new Comparator<PromedioDuracionIncidente>() {
     @Override
     public int compare(PromedioDuracionIncidente promedio1 , PromedioDuracionIncidente promedio2) {
       return Double.compare(promedio1.getPromedio() , promedio2.getPromedio()) ;
     }
   });
   for (PromedioDuracionIncidente promedio : promediosPorEntidad) {
     reporte.agregarDato( "El promedio de incidentes reportados en "  + promedio.getEntidad().getNombre() +  " fue de: " +
promedio.getPromedio()) ;
   }
   exportador.exportar(reporte , "EXCEL", "Ranking de mayor promedio de cierre" );
 }
}
```

Rankings – Entidades con mayor cantidad de incidentes reportados en la semana

:C



Rankings – Mayor grado de impacto de las problemáticas

Deprecado

Informes

Generación (y envío) de informes para Entidades Prestadoras y Organismos de Control

Informes – Diseño general

La clase GeneradorDeRankings será la encargada de crear el reporte (el Informe) y de enviarlo al componente Exportador para poder persistir el informe de manera que pueda ser visualizado por los organismos de control y por las entidades prestadoras.

La clase Exportador puede exportar los informes tanto en formato PDF como en formato excel.

```
exportador.exportar(reporte, "PDF", "RankingDeMayorPromedioDeCierre.pdf");
```

```
exportador.exportar(reporte, "EXCEL", "RankingDeMayorPromedioDeCierre.pdf");
```

[OPTATIVO] Otras decisiones y debates interesantes

Lo que se desee comentar..

Distribución de trabajo en el equipo

Mencionar, explícitamente, cómo trabajaron en conjunto y quién diseñó e implementó cada parte.

Fin y Gracias

¿Preguntas?

Equipo 10

Integrantes:

- Alborch, Felipe
- Gutson, Rodrigo
- Lucero, Fermin
- Piaggio, Facundo
- Vasquez, Carlos