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

Copia.me

Ya se trate de código, literatura o artículos académicos, a nadie le gusta que la gente se copie (bueno, al menos no sin citar al autor). Por eso nos han pedido construir un servicio en la nube para detectarlas.

El problema

Copia.me será un sistema en el que los usuarios subirán documentos de distintos tipos con el fin de detectar si estos son plagios. Por ahora soportaremos documentos de texto, programas y partituras pero se espera incorporar más. También se contará con revisores freelancer que se encargarán de analizar manualmente los documentos para darles un análisis más inteligente.

Copia.me estará orientado a docentes, que cargarán lotes de documentos y recibirán un reporte que indique cuales son copias entre sí.

Los usuarios de *Copia.me* podrán elegir entre tres calidades de servicio distintas; cada una una realiza distintos análisis y por supuesto **esto afectará en el precio de la revisión**. Las calidades de servicios con los que contamos son:

- **Bronce***:* sólo realiza *detección automática*, usando diferentes algoritmos, dependiendo del tipo de contenido:
  - Los documentos legales, literarios y académicos se comparan usando **Distancia de Levenshtein**
  - Los fragmentos de código se analizan usando **Detección de clones basado en árboles sintácticos**
  - Las partituras musicales se analizan usando **Acustic fingerprint**
- **Plata:** además de la detección automática, envía un cierto porcentaje (configurable por el usuario, esto influencia en el costo) de los documentos a analizar contra revisores freelancers que realizan una *revisión manual*. Cada revisor recibe un email con cada par de documentos asignados.
- **Oro:** además de la detección automática, envía todos los documentos a los revisores, y además hace una *revisión cruzada*: un cierto porcentaje de los mismos (nuevamente, configurable por el usuario, esto influencia en el costo) los envía a más de un revisor.

Cuando un análisis (detección automática, revisión manual o revisión cruzada) termina, genera un índice de copia para cada documento, entre 0 y 1. Y cuando todos los análisis correspondientes para una calidad de servicio terminan, se marca a cada documento afectado como *revisado*. Tené en cuenta **estos análisis son asincrónicos**, por lo que pueden terminar en cualquier orden.

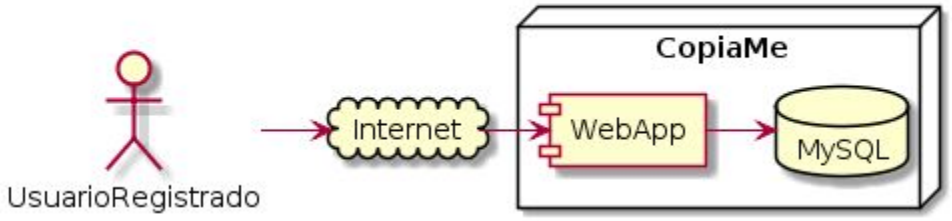
Obviamente, contar con un gran equipo de revisores freelancers no es tarea fácil, así vamos a asumir algunas cosas:

1. De cada freelancer sabemos cuántos documentos puede revisar por máximo cada mes, y su email.
2. Siempre habrá revisores disponibles.
3. Los revisores ingresarán al sistema y cargarán los resultados de sus análisis, siempre en el plazo máximo de 1 día.
4. Para la comunicación con los revisores, ya contamos con un componente que nos permite enviar emails usando SMTP (aunque en un futuro tal vez contratemos un servicio externo).

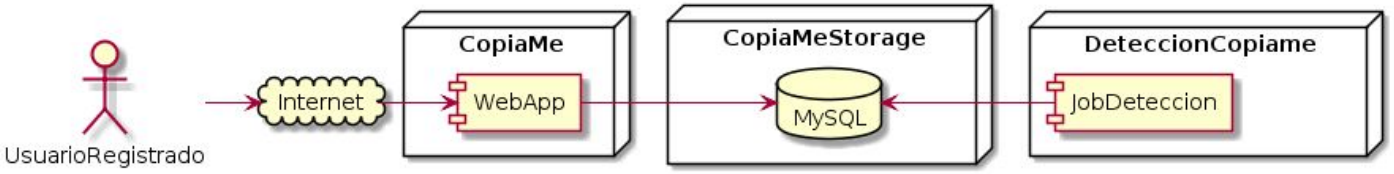
Punto 1: Arquitectura (30 Puntos)

Compará las siguientes arquitecturas indicando ventajas y desventajas de cada una. ¿Cuál creés que es la más apropiada para el dominio presentado? ¿Por qué?

Opción 1) Una instancia de aplicación web.



Opción 2) Una instancia de la aplicación web y otro nodo capaz de procesar la detección automática de copias.



Punto 2: Modelado en Objetos (40 Puntos)

Modelar el dominio presentado, que se adecue a la arquitectura elegida, utilizando el paradigma orientado a objetos, comunicando su solución mediante un diagrama de clases (**obligatorio**), código, pseudocódigo, otros diagramas y/o prosa (complementario).

Tiene que quedar claro cómo cargar un servicio nuevo, realizar el análisis automático, y notificar a los revisores freelance si fuera necesario, cómo consultar el costo de un servicio y cómo calcular el nivel el índice de copia de un documento. **Anotá todas las suposiciones que consideres necesarias**.

Punto 3: Persistencia (30 Puntos)

Utilizando un DER (**obligatorio**), código anotado y/o prosa (complementario), se pide explicar cómo persistir el modelo de objetos del punto anterior en una base de datos relacional, indicando claramente:

- Qué elementos del modelo es necesario persistir.
- Las claves primarias, las foráneas y las restricciones (constraints) según corresponda.
- Estrategias de mapeo de herencia utilizadas, si fueran necesarias.
- Las estructuras de datos que deban ser desnormalizadas, si corresponde.
- Justificaciones sobre las decisiones de diseño tomadas anteriormente.

Explicar supuestos y justificar decisiones de diseño.
Condiciones de aprobación: Para aprobar debe sumar como mínimo 60 puntos y no menos del 50 % en cada sección: (A) otorga 30 puntos, (B) 40 puntos y (C) 30 puntos.