Ejercicio 1

Analizar para Gilded Rose (https://gitlab.com/tecnicas7510/7510-exercise-0/):

1. ¿Quiénes son los stakeholders?
2. ¿Qué objetivos tiene cada stakeholder?
3. ¿Cómo priorizaría dichos objetivos?

- Cliente
    - Usar el sistema para consultar inventario
    - Extender el sistema para un nuevo producto "Conjured Mana Cake"
- Usuario (Allison)
    - Usar el sistema para consultar inventario
    - Usar el sistema para actualizar inventario
- Goblin in the corner
    - Que no se modifique su propio código
- Developer (Yo)
    - Sobrevivir
    - Remuneración
    - Modificar fácilmente el sistema

1. Developer: Sobrevivir
2. Goblin: Que no se modifique su propio código
3. Allison/Cliente: Usar el sistema para consultar inventario
4. Allison: Usar el sistema para actualizar inventario
5. Ambas:
    a. Cliente: Extender el sistema para un nuevo producto "Conjured Mana Cake"
    b. Developer: Remuneración
6. Developer: Modificar fácilmente el sistema

Ejercicio 2

Se desea proponer una arquitectura para un editor local de diagramas UML, el cual debe poder usarse para operar sobre una representación gráfica del mismo.

1. ¿Qué requerimientos funcionales tiene el sistema?
2. ¿Qué requerimientos no funcionales tiene el sistema?
3. Proponga una arquitectura apropiada, indicando patrones aplicados si los hay.

Requerimientos funcionales:

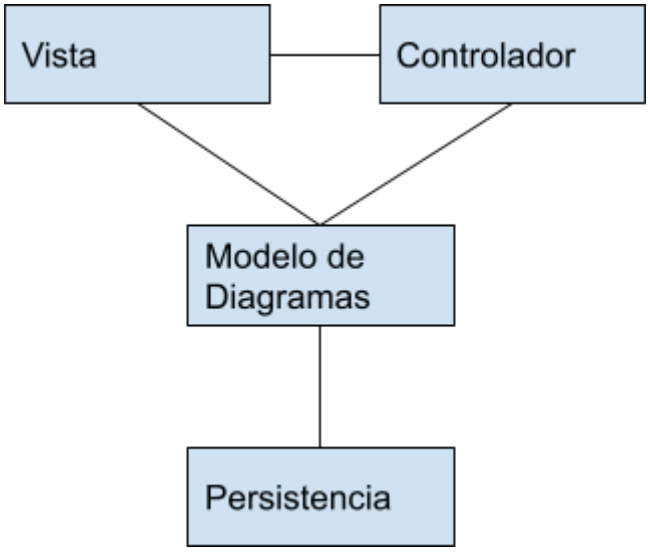
- Crear diagramas
- Establecer relaciones
- Cargar diagramas
- Guardar diagramas
- Compartir diagramas
- Exportar como imágenes

Requerimientos no funcionales:

- Usabilidad
- Extensibilidad (Tipos de diagramas)
- Performance
- Testeabilidad
- Disponibilidad
- Seguridad
- Portabilidad

Arquitectura:

- Capas (Presentación, Negocio, Persistencia)
- MVC (M: Objetos del diagrama, V: Imagen del diagrama, C: Reacción a dispositivos de entrada)



Ejercicio 3

Se busca proponer una arquitectura para un sistema de análisis estático de código. Este tipo de sistemas generan reportes sobre cuestiones como calidad de código y posibles bugs, a partir de reglas y cálculos estadísticos sobre el código fuente de una aplicación. Si bien estas reglas pueden ser aplicables a un solo lenguaje de programación, lo usual es que (dada una representación intermedia apropiada) generalicen a situaciones como "cualquier lenguaje orientado a objetos" o "cualquier lenguaje que usa SQL embebido como strings", cosa que se desea aprovechar para ampliar la audiencia de este sistema.

1. ¿Qué requerimientos funcionales tiene el sistema?
2. ¿Qué requerimientos no funcionales tiene el sistema?
3. Proponga una arquitectura apropiada, indicando patrones aplicados si los hay.

Restricciones:

- Debemos conocer a qué lenguajes aplica

Requerimientos funcionales:

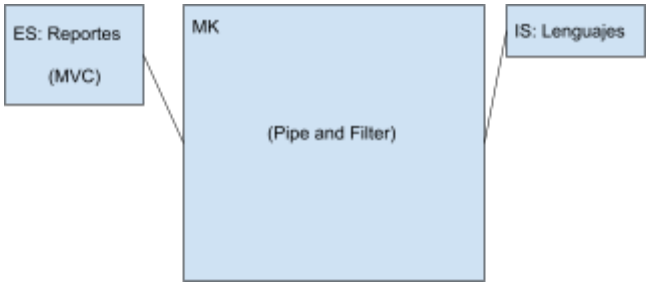
- Leer archivos de código
- Abstraer otros lenguajes a una representación intermedia
- Detectar si las reglas son aplicables al lenguaje del input
- Saber detectar bugs en código
- Generar reportes
    - Incluyendo algún resumen (score, métricas, etc)

Requerimientos no funcionales:

- Portabilidad
- Performance
- Usabilidad
- Modificabilidad/Extensibilidad
- Seguridad
- Interoperabilidad

Arquitectura:

- Pipe and filter: Texto extendido con estadísticas
- Microkernel (IS: Lenguajes, ES: Reportes, MK: Reglas y detección de bugs sobre representación intermedia)
- MVC (Reportes)


ES: Reportes
(MVC)
MK
(Pipe and Filter)
IS: Lenguajes

Ejercicio 4

Se desea proponer una arquitectura para un sistema de detección de abuso en una red social. El mismo debe poder analizar cada post y mensaje dentro de la red social antes de mostrarlo a otros usuarios, determinando si representa una violacion de los términos y condiciones.

Se dispone de una cantidad de componentes de software - paquetes, librerías, programas y APIs de terceros - que se combinarán para evaluar el contenido. Deberá ser posible adaptar el sistema en caso que se agreguen o remuevan componentes disponibles, y en caso que se descubra que otra combinación de componentes produce un mejor resultado.

1. ¿Qué requerimientos funcionales tiene el sistema?
2. ¿Qué requerimientos no funcionales tiene el sistema?
3. Proponga una arquitectura apropiada, indicando patrones aplicados si los hay.

Restricciones:

- Alguien más toma las decisiones de moderación basado en lo que detecta el sistema

Requerimientos funcionales:

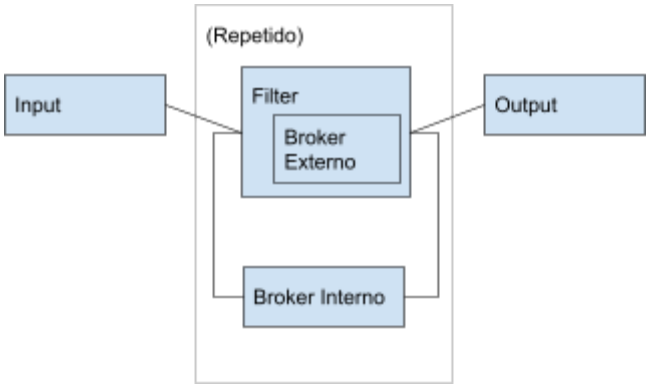
- Analizar contenido (Posts y mensajes)
    - Reconocer que contenido viola terminos y condiciones
    - X% del contenido negativos sean correctamente identificados
- Modificabilidad
    - Configurar nivel de certeza necesario para marcar contenido como negativo

Requerimientos no funcionales:

- Disponibilidad
- Interoperabilidad (APIs de terceros, programas usados para evaluar; interfaz con quien recibe las decisiones)
- Escalabilidad
- Performance
- Seguridad (Datos privados en mensajes)
- Monitoreabilidad
- Robustez (No haya cambios pequeños del algoritmo que causen grandes errores)

Arquitectura:

- Pipe and filter (Contenido aumentado con estadísticas)
- Broker (Unificar interfaces de componentes internos y APIs externas; como pipes)


(Repetido)
Input
Filter
Broker
Externo
Output
Broker Interno

Ejercicio 5

Se desea desarrollar una aplicación para la carga de datos de inspecciones de automotores para una empresa de seguros. La empresa posee una dotación de 100 agentes inspectores. Cada agente de seguros, portador de un teléfono o notebook que la empresa le entregará recibirá a través de la aplicación la agenda del día con los datos de los automotores a inspeccionar. Cada agenda está identificada con el número de empleado del agente el cual consta de veintiuno caracteres alfanuméricos, este será verificado al momento de ser recibida. La agenda posee los datos de cada automotor a inspeccionar y los de su propietario. Las agendas son diarias y cada una no podrá incluir más de 6 inspecciones, además la aplicación no podrá recibir una agenda si previamente no envió a la empresa todas las inspecciones de la agenda del día anterior. El agente se traslada al domicilio del propietario y realiza la inspección del automotor agendado para ser asegurado. Al realizar cada inspección el agente deberá registrar la marca, el modelo y los números de serie de cada componente (chasis, ruedas, motor, carrocería) y verificar y registrar su estado de conservación. Cada componente podrá estar en buen, regular o mal estado. En el caso de no poder ser identificado deberá marcarse como no legible. Las inspecciones podrán realizarse de una vez o podrán empezarse y continuarse para ser terminadas en otro momento. Las inspecciones podrán ser canceladas y también marcadas como de imposible realización registrando la razón. Una vez completada una inspección, ésta podrá ser enviada a la empresa o almacenada para ser enviada a posteriori. Las inspecciones no realizadas o canceladas no podrán ser enviadas a la empresa. Nos encargaron hacer un diseño preliminar y propuesta de arquitectura.

Al relever mejor surgieron nuevos requerimientos:

- Es prioritario que el inspector pueda trabajar esté donde esté, como ser zonas rurales o de poca o nula conexión.
- Se desea adjuntar fotos de las partes inspeccionadas.
- Los dispositivos de los inspectores no tienen capacidad infinita de almacenaje, se quiere priorizar el uso de espacio en los mismos de fotos e información de inspecciones.
- Se desea integrar con Aplicaciones de Mapas para obtener la ruta al domicilio de los propietarios o auto a inspeccionar.

Y también nos asignaron también el desarrollo de un backoffice:

- Surge la necesidad de implementar también el backoffice que podría ser un sistema Web para que un Rol Administrador pueda armar las agendas que se van a asignar a los inspectores.
- También va a poder ver las inspecciones realizadas, donde se podrán buscar y filtrar por datos del auto y titular, y ver el detalles de las mismas.

- Se va a permitir entonces cargar los pedidos de inspección y organizar nuevas agendas para los inspectores.
- Para el armado de agendas, el sistema debe proveer un mecanismo que permita indicar distancia entre las locaciones de los automotores, para que el administrador evite asignar locaciones muy lejanas entre sí en una misma agenda. Esto es solo orientativo, la lógica puntual de que inspecciones poner en una misma agenda son arbitrarias y las define el administrador.
- Las agendas completas se asignan en forma automática de forma diaria al final del día, y se asignan al primero que completa la agenda previa y está disponible para una nueva.
- Para la autenticación se quiere usar un sistema que la empresa ya tiene llamado KeyCloak que permite integrarse vía OAUTH para autenticarse y obtener un conjunto de permisos y/o roles.

- Se desea enviar notificaciones por email y/o SMS a los inspectores en casos como:
    - Cuando hay nuevas agendas listas y se requiere completen su agenda para poder recibir más inspecciones.
    - Cuando se le asigna una nueva agenda a un inspector.

1. ¿Qué requerimientos funcionales tiene el sistema?
2. ¿Qué requerimientos no funcionales tiene el sistema?
3. Proponga una arquitectura apropiada, indicando patrones aplicados si los hay.