Universidad Nacional de La Plata
Facultad de Informática

Tesina de grado

Componente Genérico de Auditoría para Monitorear Cambios en el Modelo de Objetos

Alumno: Javier Corvi
Directores: Javier Díaz – Claudia Queiruga

Agenda

Introducción

Objetivo

Marco Teórico

Análisis Funcional

Alcance de la librería

Diseño

Aplicación de la librería - Testing

Trabajos Futuros

Conclusión

Introducción

En las aplicaciones de software que manejan información crítica, la seguridad y la auditoría de la misma es un requerimiento obligatorio a la hora de realizar el desarrollo de software.

La información va modificandose constantemente durante su ciclo de vida . . . . . .
Dejar registro de estas modificaciones es un gran avance para mejorar la seguridad de la información dentro de un sistema.

Introducción

Monitorear los cambios realizados sobre los datos del sistema.

Objetivo

El objetivo de mi tesina está centrado en desarrollar un componente genérico que implemente el requerimiento de monitorear la información del sistema de forma simple y escalable.

Solución

Diseñar y desarrollar una librería de aspectos JAVA genérica que audite las altas, bajas y actualizaciones de las entidades del modelo de objetos en una aplicación que utilice la API de persistencia de Java (JPA) junto con el Framework Hibernate encargado de realizar el mapeo objeto-relacional.

A este desarrollo se le dará el nombre de AuditTrail

Agenda

Introducción

Objetivo

Marco Teórico

Análisis Funcional

Alcance de la librería

Diseño

Aplicación de la librería - Testing

Trabajos Futuros

Conclusión

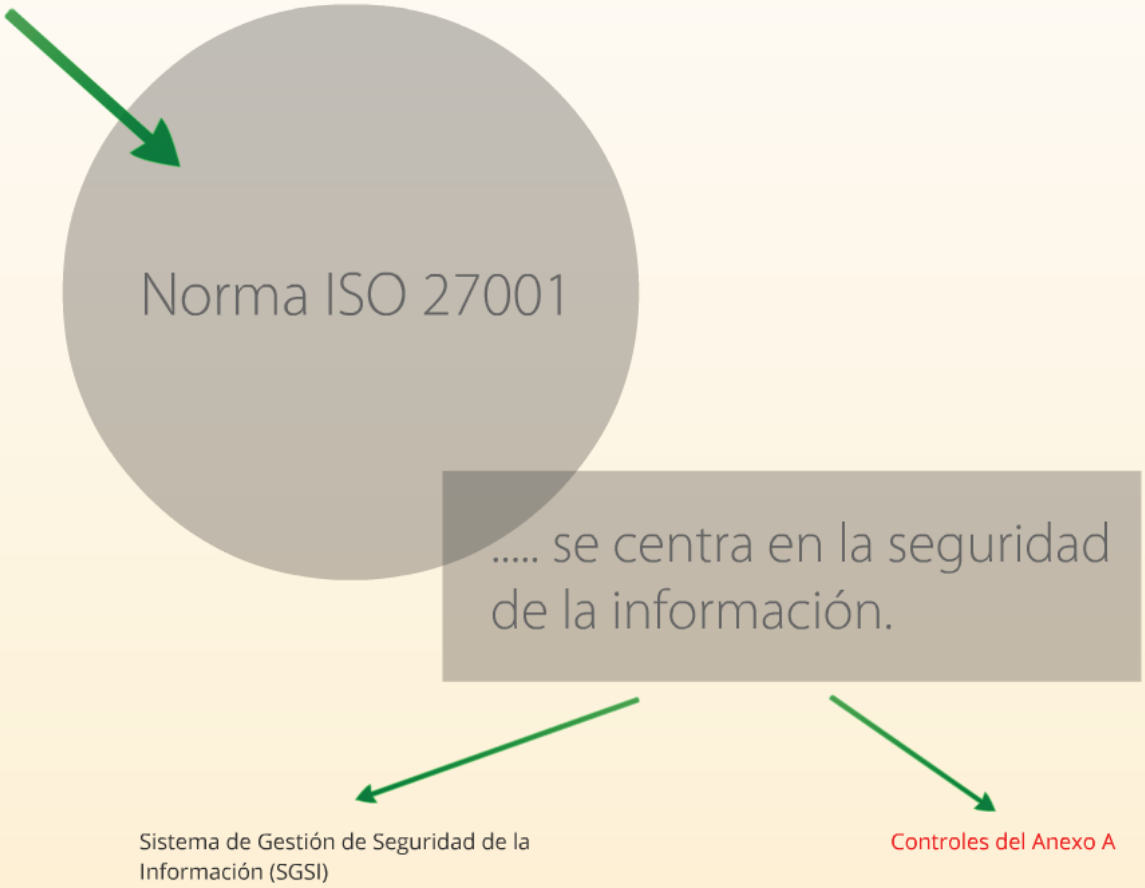
En muchas empresas, los activos más valiosos que poseen residen en la información que manejan.

Marco Teórico

La administración de los riesgos asociados a la información, así como el incremento de requerimientos para controlar la misma es esencial para la mejorar la calidad del negocio.

Marco Teórico

Las organizaciones buscan continuamente la calidad en el negocio, para ello muchas implementan las normas ISO 9000 para mejorar distintos aspectos dentro de la organización.



Norma ISO 27001

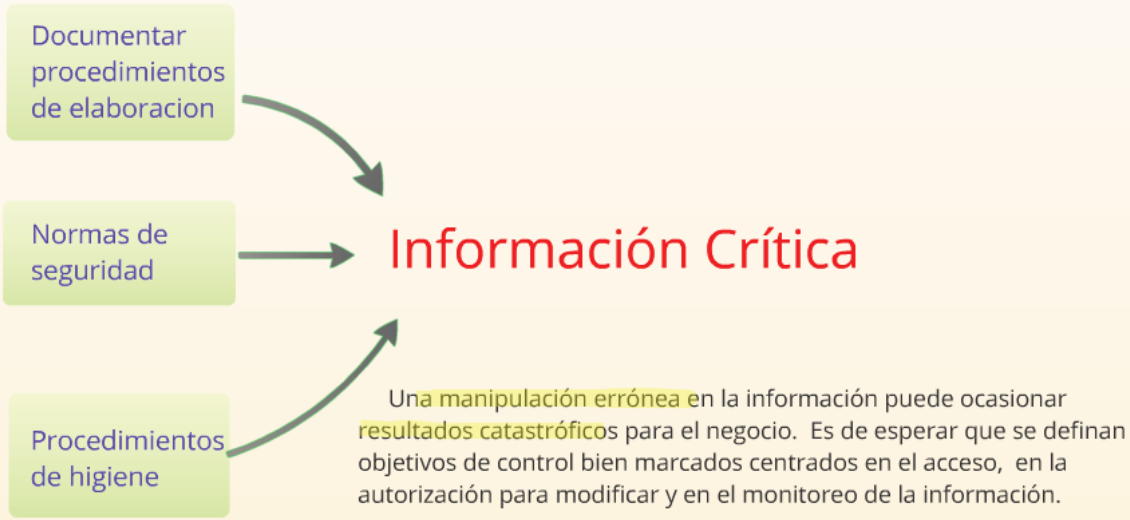
..... se centra en la seguridad de la información.

Sistema de Gestión de Seguridad de la Información (SGSI)

Controles del Anexo A

Caso de estudio

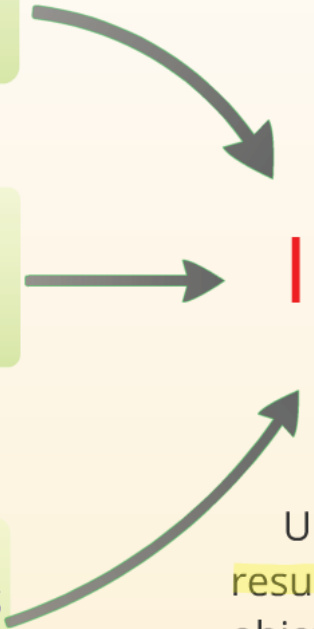
Laboratorio farmacéutico



Documentar procedimientos de elaboracion

Normas de seguridad

Procedimientos de higiene



Información Crítica

Una manipulación errónea en la información puede ocasionar resultados catastróficos para el negocio. Es de esperar que se definan objetivos de control bien marcados centrados en el acceso, en la autorización para modificar y en el monitoreo de la información.

Creación del procedimiento para elaborar el medicamento “Ibupirac 400”

Registro del AuditTrail

Entidad	Identificador	Usuario	Fecha	Acción
Procedimiento de elaboración	3	jcorvi	05/06/2010	Creación

Valores Auditados: creación, todos los atributos.			
Nombre	**Acción Terapéutica**	**Composición**	
Ibupirac 400	Analgésico, antiinflamatorio y Antifebril	Ibuprofeno	400 mg
		Aerosol	2 mg
		Lactosa	20 mg
		Avicel	2 mg
		Ac-Di-Sol	4 mg
		Est de magnesio	10 mg

Supongamos las siguientes situaciones

Modificación errónea

El laboratorio decide subir la dosis de Lactosa de 20 a 25 mg y la dosis de Avicel de 2 a 4 mg
La información del medicamento es actualizada en forma incorrecta, el responsable modifica la dosis de Lactosa de 20 a 30 mg.

Entidad	Identificador	Usuario	Fecha	Acción
Procedimiento de elaboración	3	Jcorvi	05/09/2010	Actualización

Valores Auditados: actualización, solo los atributos modificados		
Composición		
	Valor viejo	**Valor nuevo**
Lactosa	20 mg	30 mg
Avicel	2 mg	4 mg

Acción malintencionada

Usuarios internos o externos a la empresa, modificando o eliminando información

Entidad	Identificador	Usuario	Fecha	Acción
Procedimiento de elaboración	3	Hacker	08/10/2010	Eliminación

Modificación errónea

El laboratorio decide subir la dosis de Lactosa de 20 a 25 mg y la dosis de Avicel de 2 a 4 mg
La información del medicamento es actualizada en forma incorrecta, el responsable modifica la dosis de Lactosa de 20 a 30 mg.

Entidad	Identificador	Usuario	Fecha	Acción
Procedimiento de elaboración	3	Jcorvi	05/09/2010	Actualización

Valores Auditados: actualización, solo los atributos modificados		
Composición		
	Valor viejo	**Valor nuevo**
Lactosa	20 mg	30 mg
Avicel	2 mg	4 mg

Acción malintencionada

Usuarios internos o externos a la empresa, modificando o eliminando información

Entidad	Identificador	Usuario	Fecha	Acción
Procedimiento de elaboración	3	Hacker	08/10/2010	Eliminación

Información Inconsistente

- Detectar al responsable
- Momento exacto
- Análizar las causas
- Tomar acciones correctivas

Casos Reales

Estas situaciones pueden tornarse más complejas a nivel de negocio ...

Multinacional Mattel

Ocurrió en el 2007 cuando la multinacional de juguetes “Mattel” lanzó un aviso mundial “para retirar del mercado todos los ejemplares del muñeco Sarge, de la película de dibujos animados Cars, porque para fabricarlos se ha utilizado pintura contaminada con plomo”

En total se distribuyeron 18,2 millones de unidades afectadas a nivel mundial.

Pfizer S.R.L

El caso que se describió en el marco teórico, es idéntico al ocurrido en el 2010 cuando se retiraron del mercado 33 lotes alterados de Ibupirac, fabricados por la empresa Pfizer S.R.L.

"...según un comunicado de la ANMAT (Administración Nacional de Medicamentos, Alimentos y Tecnología Médica), se dispuso la quita del mercado de 33 lotes del analgésico y antifebril denominado Ibupirac, por presentar alteraciones en el olor y sabor..."

Agenda

Introducción

Objetivo

Marco Teórico

Análisis Funcional

Alcance de la librería

Diseño

Aplicación de la librería - Testing

Trabajos Futuros

Conclusión

Análisis Funcional

Requerimientos de control de la librería

- Constancia de las modificaciones que ocurrieron sobre la información
- Recuperar la información en algún momento de su ciclo de vida
- Obtener indicadores descriptivos de la información ..... Data Warehousing
- Separar los registros generados por el AuditTrail de la información del sistema

Constancia de las modificaciones que ocurrieron sobre la información

Constancia de las modificaciones que ocurrieron sobre la información

A.10.10.1 Monitoreo de actividades	Control Registrar las actividades del usuario, excepciones, y distintos eventos definidos por la organización. Los registros deberán conservarse durante un período determinado para ayudar en las investigaciones futuras y de supervisión del control de acceso.
A.10.10.2 Monitorear el uso del sistema	Control Procedimientos para monitorear el uso de procesamiento de la información deben ser establecidos y los resultados revisados regularmente.

Recuperar la información en algún momento de su ciclo de vida

Obtener indicadores descriptivos de la información .....
Data Warehousing

Separar los registros generados por el AuditTrail de la información del sistema

Separar los registros generados por el AuditTrail de la información del sistema

A.10.10.3 Protección de logs del monitoreo	Control Los logs del monitoreo de la información deben estar protegidos contra manipulación y acceso no autorizado.

Análisis Funcional

Requerimientos de control de la librería

- Constancia de las modificaciones que ocurrieron sobre la información
- Recuperar la información en algún momento de su ciclo de vida
- Obtener indicadores descriptivos de la información ..... Data Warehousing
- Separar los registros generados por el AuditTrail de la información del sistema

Documentación .....

- Enterprise Architect
- Embarcadero
- Javadoc

Agenda

Introducción

Objetivo

Marco Teórico

Análisis Funcional

Alcance de la librería

Diseño

Aplicación de la librería - Testing

Trabajos Futuros

Conclusión

Alcance de la librería

Paradigma Orientado a Objetos (POO).

Se aplica sobre el modelo de dominios
o modelo de objetos.

Aplicaciones desarrolladas en Java.

JPA junto con el Framework Hibernate.

Diferencias entre el mundo de bases de datos
relacionales y el orientado a objetos.

Agenda

Introducción

Objetivo

Marco Teórico

Análisis Funcional

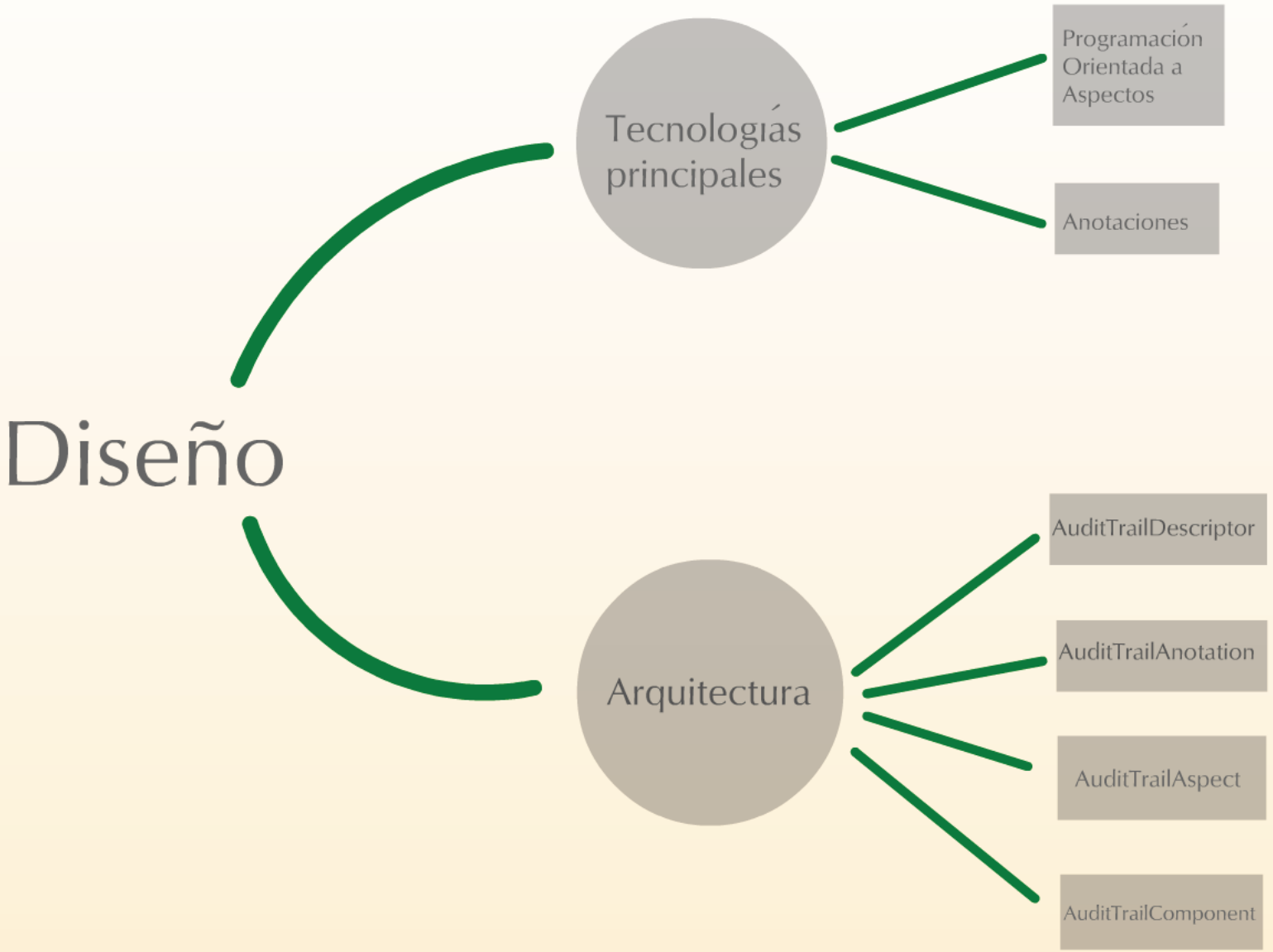
Alcance de la librería

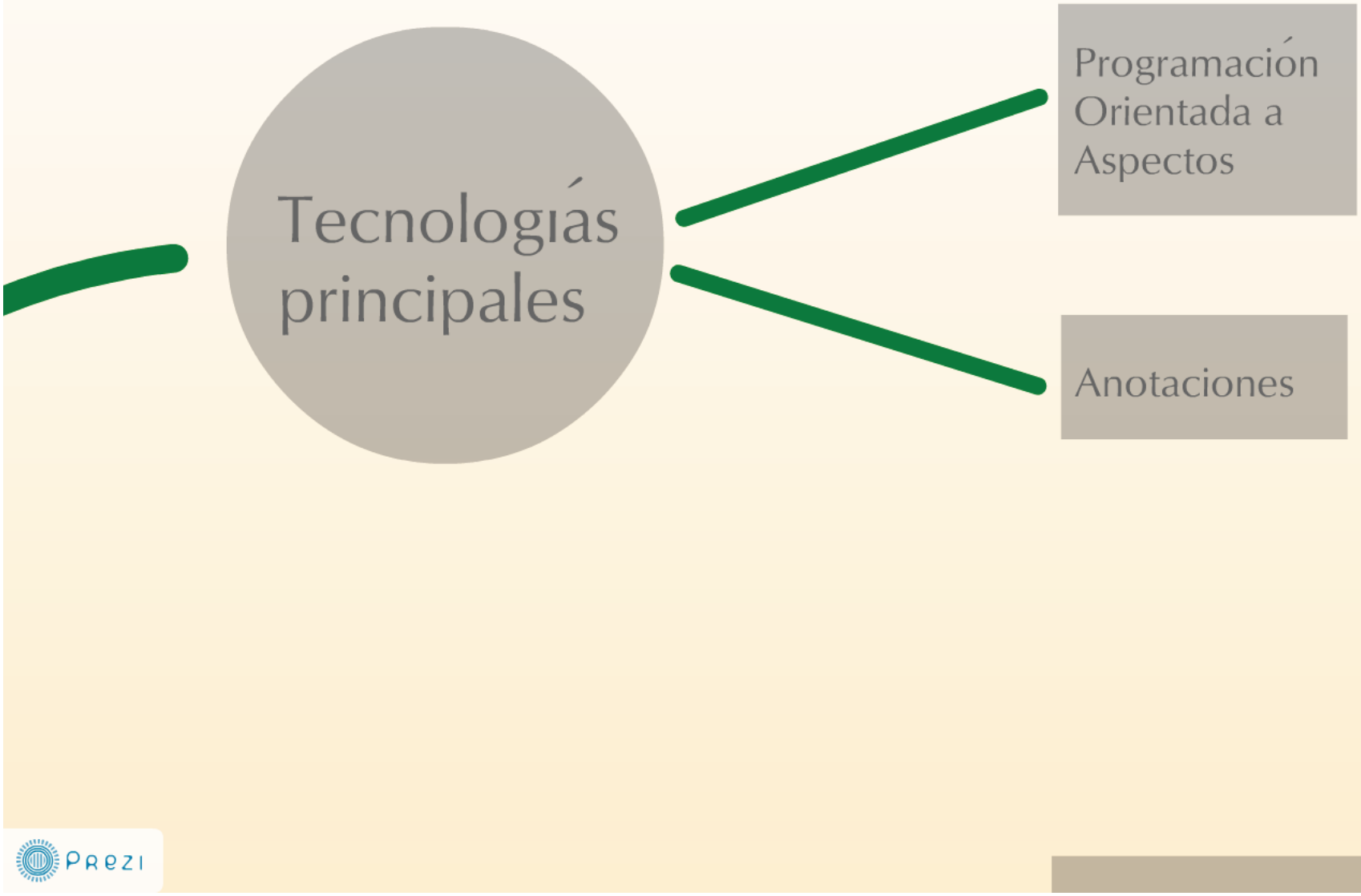
Diseño

Aplicación de la librería - Testing

Trabajos Futuros

Conclusión


Diseño
Tecnologías principales
Programación Orientada a Aspectos
Anotaciones
Arquitectura
AuditTrailDescriptor
AuditTrailAnotation
AuditTrailAspect
AuditTrailComponent


Tecnologías principales
Programación Orientada a Aspectos
Anotaciones

Programación Orientada a Aspectos

Requerimientos no funcionales:

- Auditoría de la información
- Seguridad
- Transacciones
- Excepciones
- Logeo
- Concurrencia

AOP - Requerimientos no funcionales

La Programación Orientada a Aspectos permite abstraer y modularizar los requerimientos no funcionales en unidades de software denominadas aspectos que luego se integrarán a la lógica de negocios.

Beneficios

Mayor grado de modularización

Reusabilidad de código

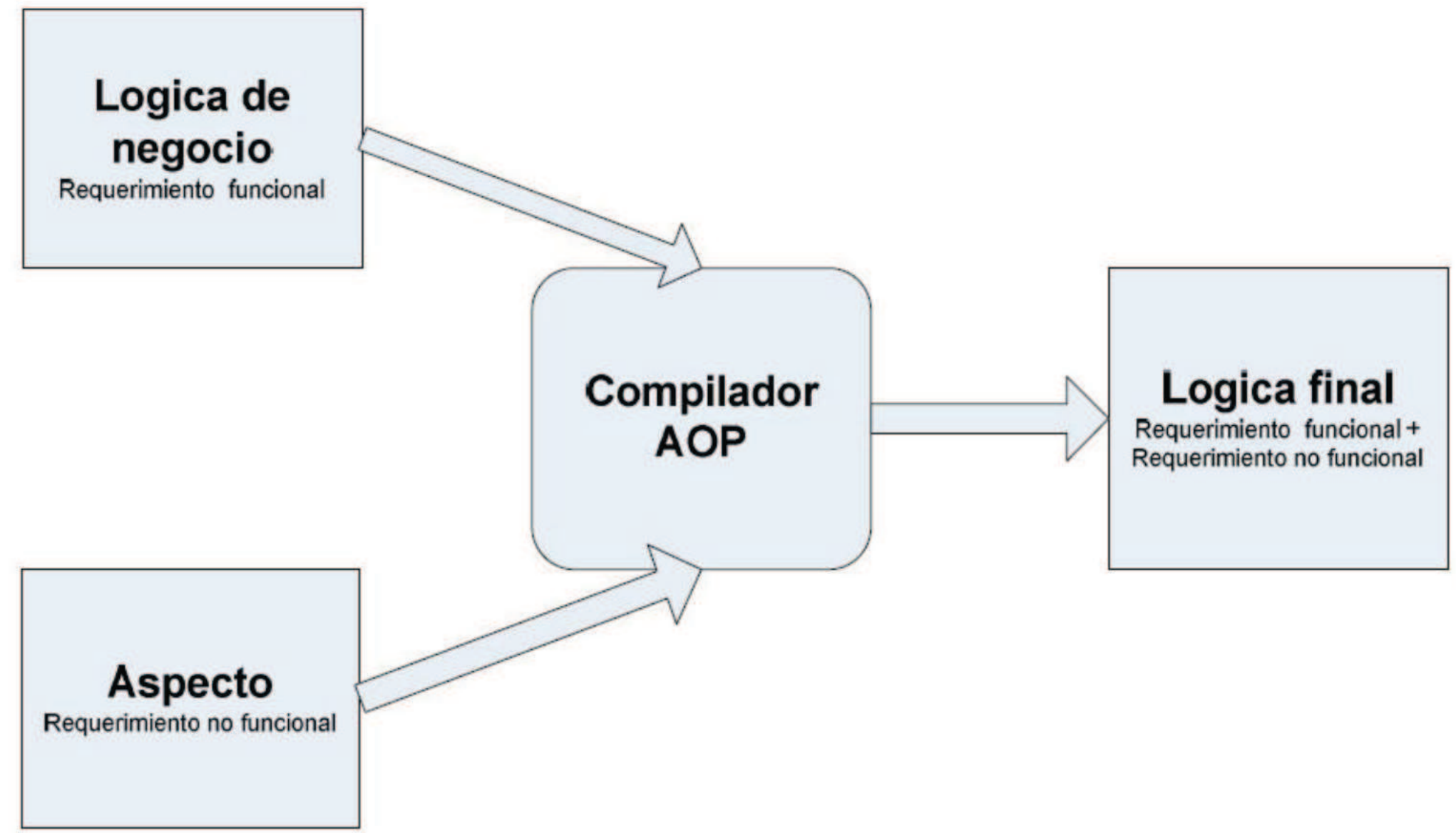
Código menos confuso

AspectJ

Extensión del lenguaje JAVA que agrega los conceptos básicos de AOP

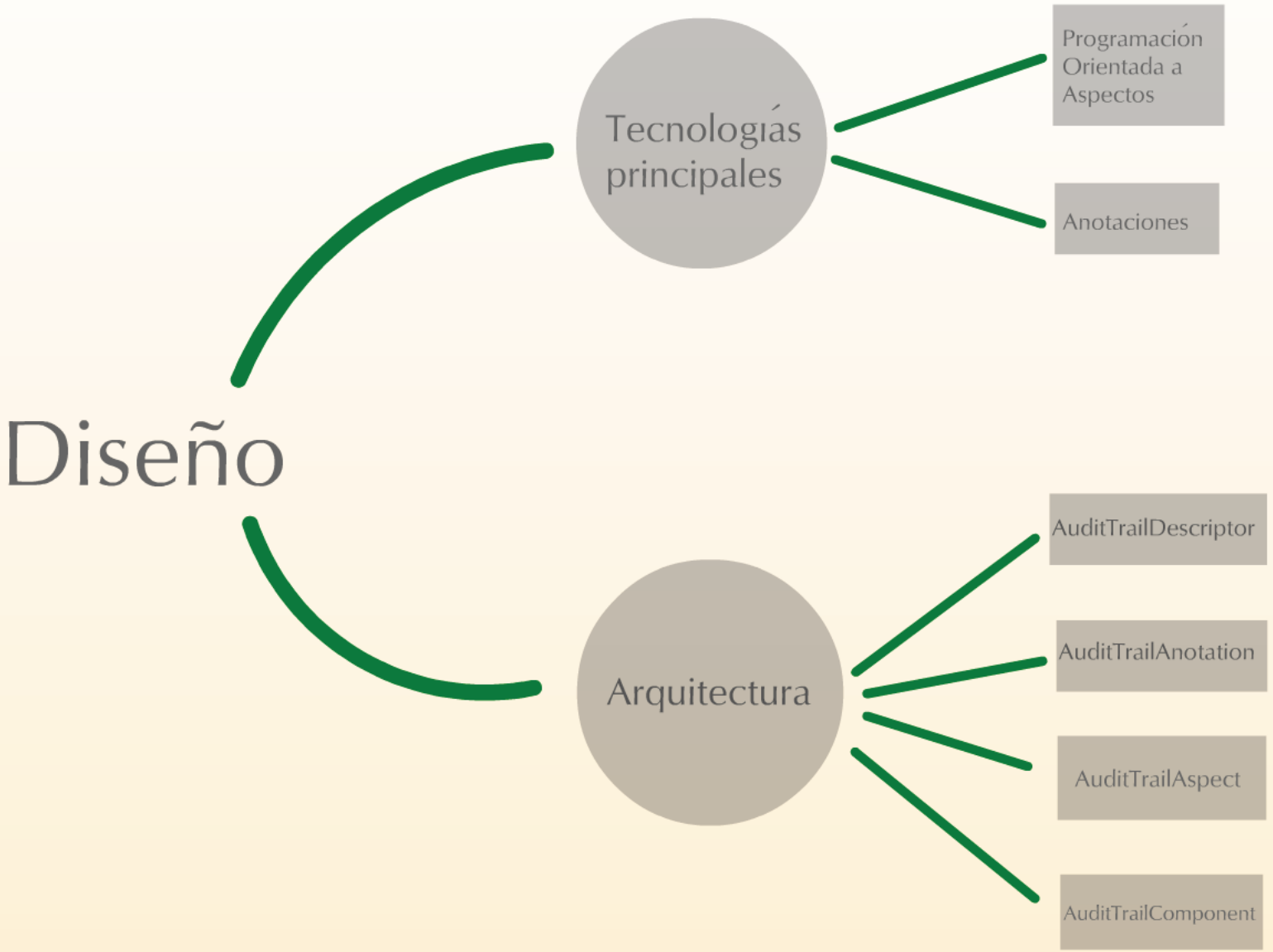
Conceptos básicos

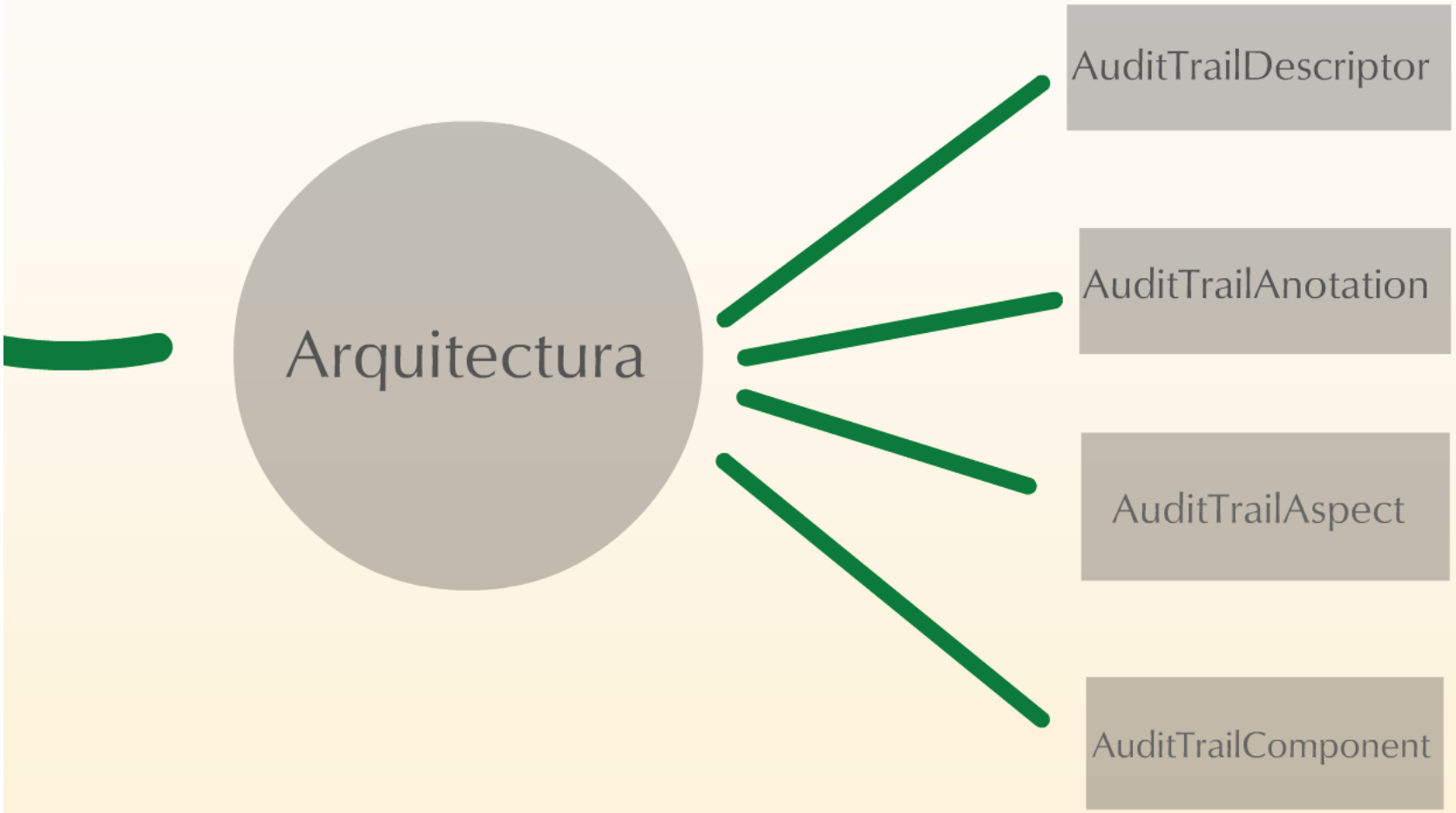
- Apect
- Join Points
- Pointcut
- Advice
- Weaving


Logica de negocio
Requerimiento funcional
Aspecto
Requerimiento no funcional
Compilador AOP
Logica final
Requerimiento funcional +
Requerimiento no funcional

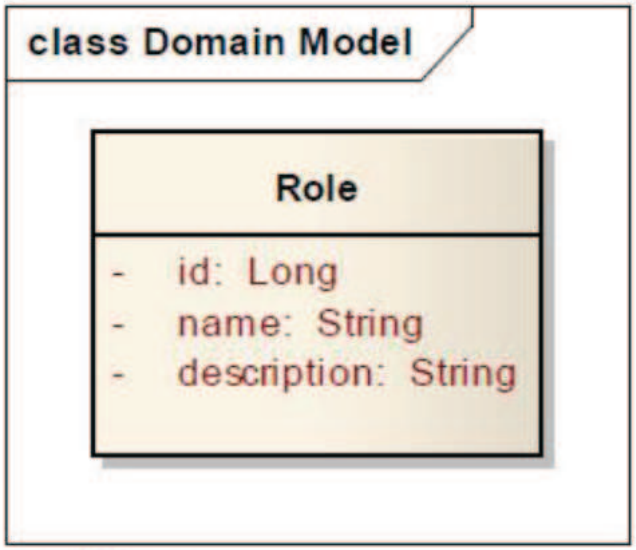
Anotaciones

- Forma simple de agregar metadatos al código fuente.
- Se usa como una alternativa a la tecnología XML.

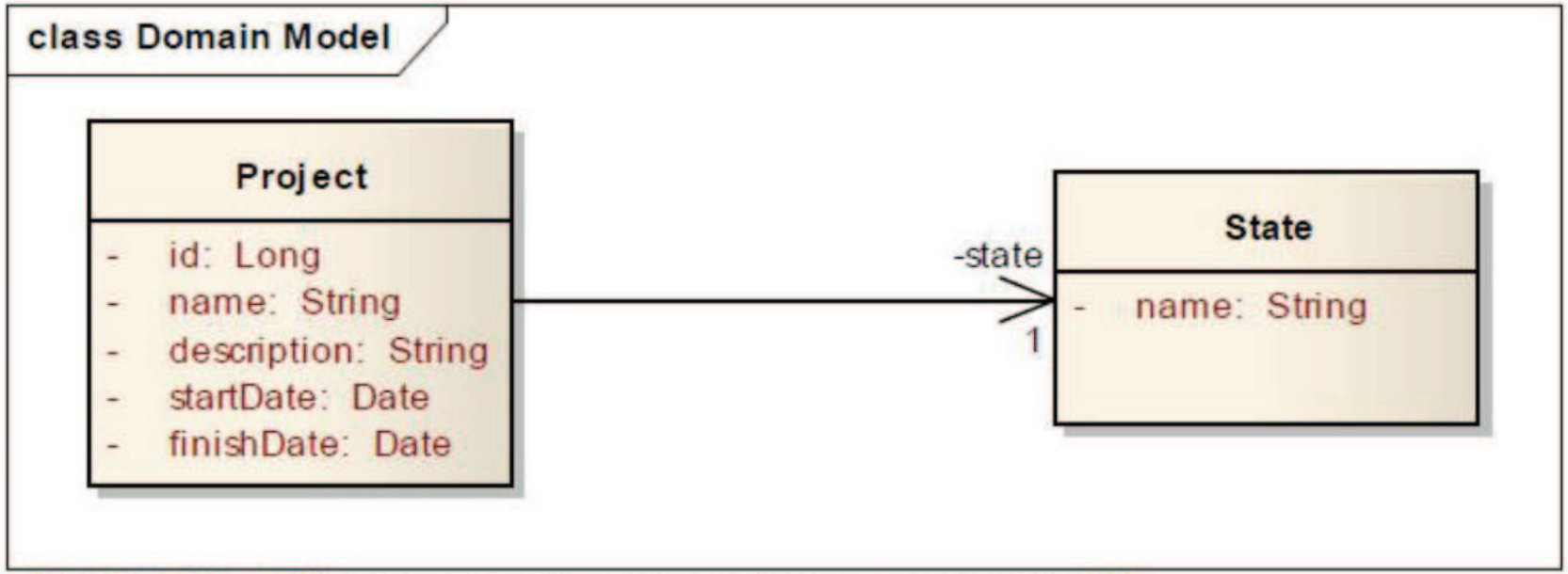

Diseño
Tecnologías principales
Programación Orientada a Aspectos
Anotaciones
Arquitectura
AuditTrailDescriptor
AuditTrailAnotation
AuditTrailAspect
AuditTrailComponent


Arquitectura
AuditTrailDescriptor
AuditTrailAnotation
AuditTrailAspect
AuditTrailComponent

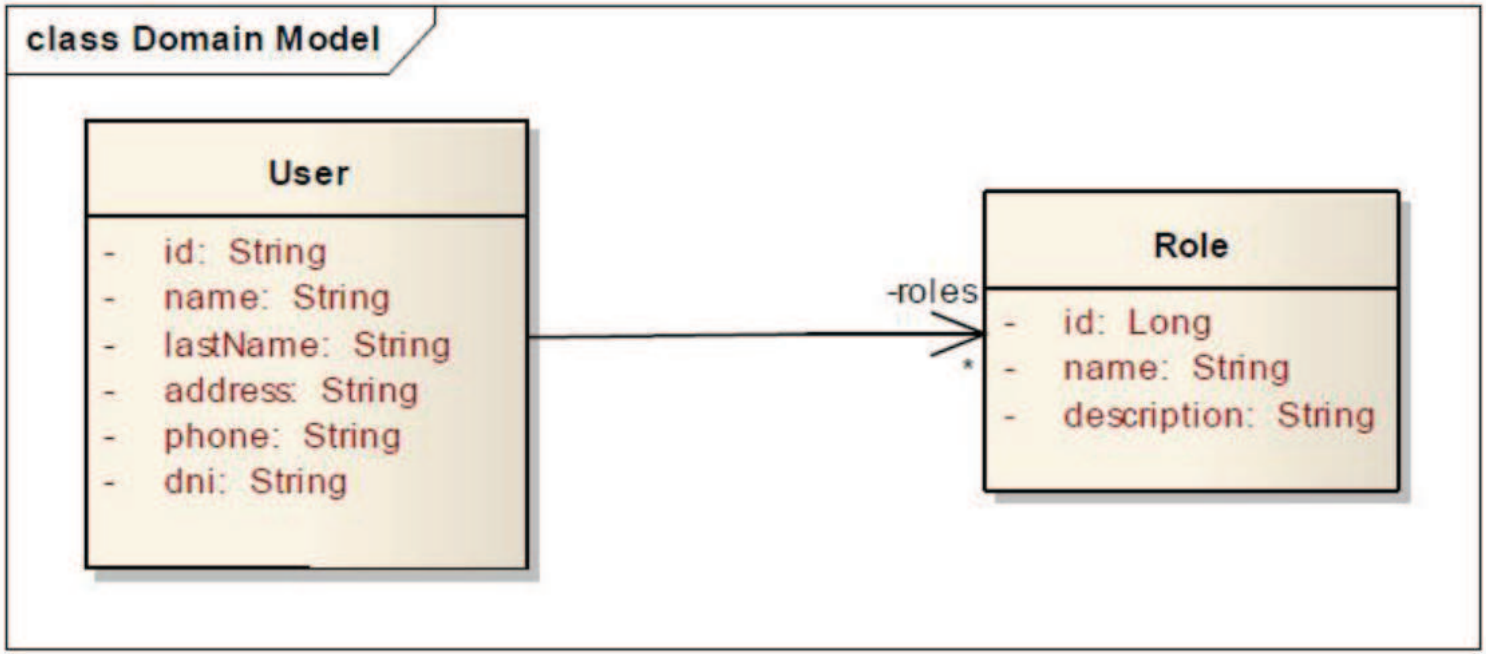
AuditTrailDescriptor




```
<entities>
    <!-- declaracion de la entidad a auditar, se agrega el nombre de la clase -
    <entity name="facultad.multimedia.staffing.model.Role">
        <identifier>id</identifier> <!-- indica el nombre del identificador -->
        <fields> <!-- coleccion de campos a auditar -->
            <field> <!--campo a auditar -->
                <!--nombre del atributo -->
                <name>name</name>
                <!--i18n del atributo (para visualizacion) -->
                <i18nname>comm.i18n.role.name</i18nname>
            </field>
            <field>
                <name>description</name>
                <i18nname>comm.i18n.role.description</i18nname>
            </field>
        </fields>
    </entity>
```



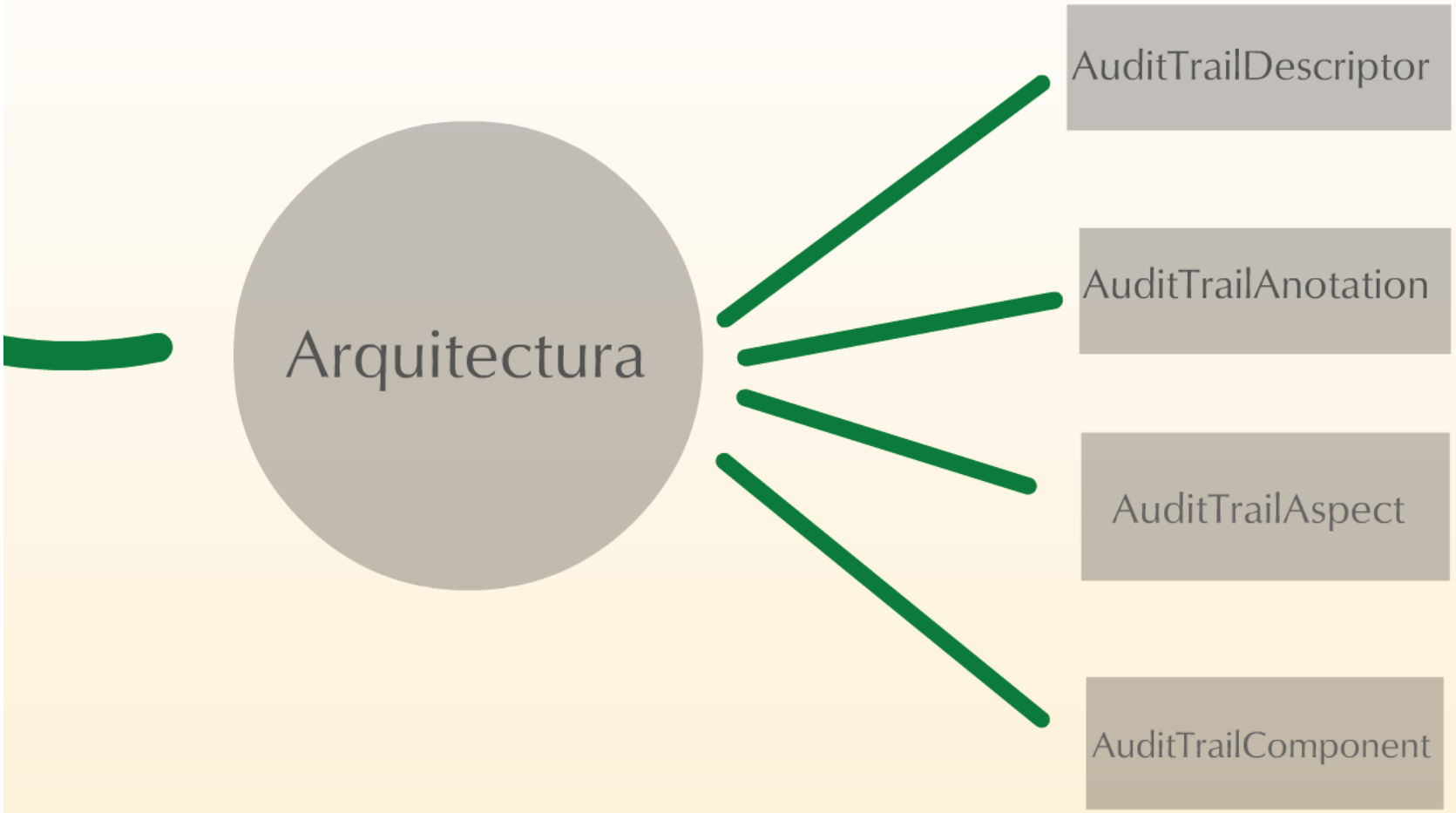

```
<entity name="facultad.multimedia.staffing.model.Project">
    <identifier>id</identifier>
    <fields>
        .....
        <field>
            <name>state</name>
            <i18nname>comm.i18n.project.state</i18nname>
            <!--si el campo es una entidad anidada -->
            <isEntity identifier="id" fieldValue="name"/>
            <!--id es el nombre del identificador de esa entidad -->
            <!-- fieldValue es el valor que se tomara de la entidad anidada
            para verificar el cambio -->
        </field>
        .....
    </fields>
</entity>
```




```
<entity name="facultad.multimedia.staffing.model.User">
    <identifier>id</identifier>
    <fields>
        ..............
        <field>
            <name>roles</name>
            <i18nname>comm.i18n.user.roles</i18nname>
            <!--si el campo es una coleccion -->
            <isCollection identifier="id" fieldValue="name"/>
            <!--id es el nombre del identificador de las entidades
            de la coleccion -->
            <!-- fieldValue es el valor que se tomara de las entidades de la
            coleccion para verificar el cambio -->
        </field>
    </fields>
</entity>
```

Beneficios de la configuración del audit-trail.xml

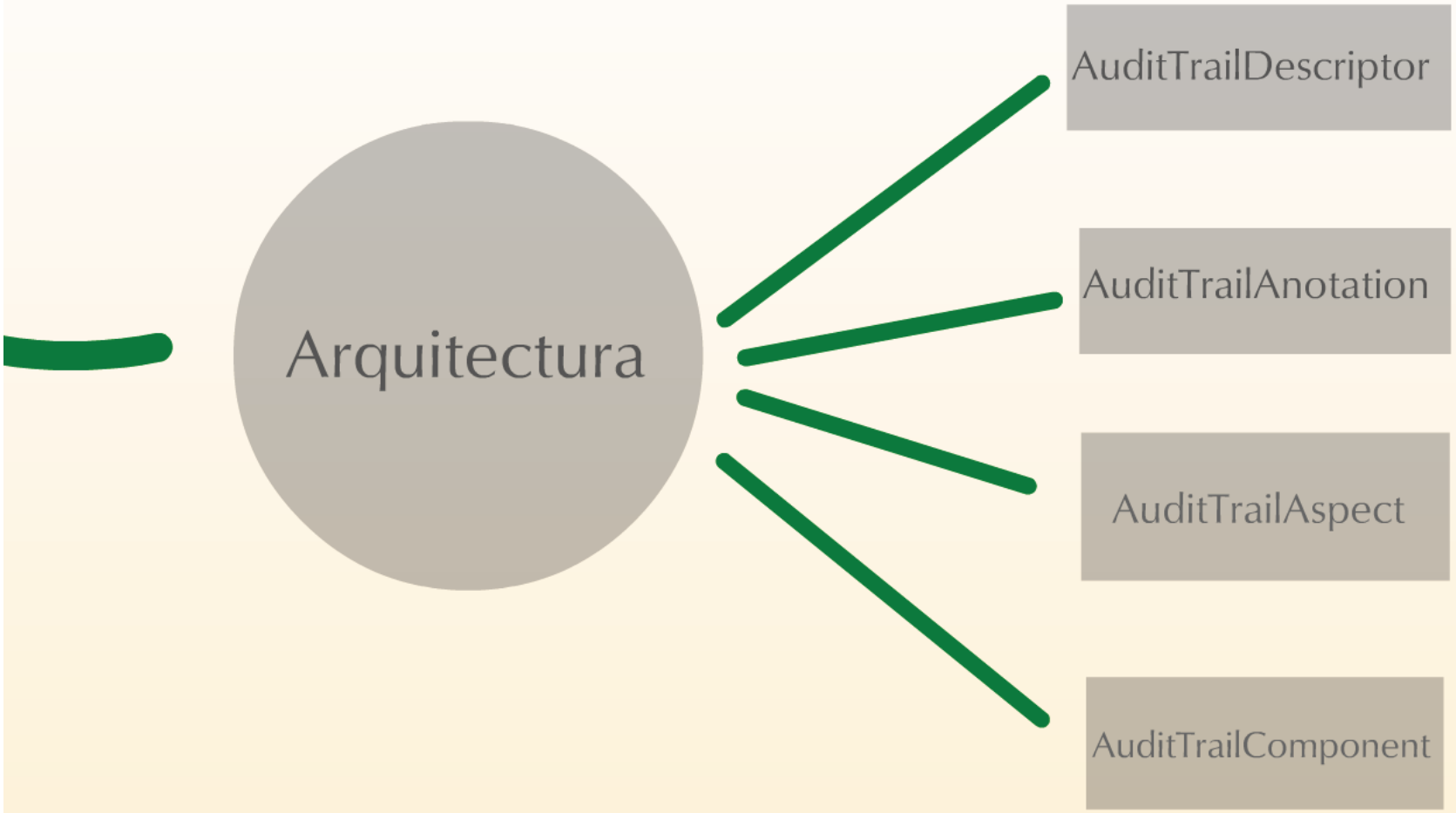
- Separado del código fuente
- Facilidad de configuración


Arquitectura
AuditTrailDescriptor
AuditTrailAnotation
AuditTrailAspect
AuditTrailComponent

AuditTrailAnotation

```
@AuditTrailAnnotation(auditedObject="user",
actionType=ActionType.CREATE,action="com.action.create")
public void createUser(User user){
        entityManager.getTransaction().begin(); //open transaction
        entityManager.persist(user);
        entityManager.getTransaction().commit(); //commit transaction
}
```

El objeto que debe ser auditado se describe en el elemento auditedObject, el cual hace referencia al nombre del parámetro a auditar, en este ejemplo "user".


Arquitectura
AuditTrailDescriptor
AuditTrailAnotation
AuditTrailAspect
AuditTrailComponent

AuditTrailAspect

```
package info.unlp.edu.ar.tesis.audit.aspect;
import .....
/**
* Aspecto que se encarga de modelar el requerimiento no funcional de
auditoría de las entidades
*/
public aspect AuditTrailAspect{
/**
* Pointcut definido para que intercepte los métodos que contengan
* el join point métodos con el Annotation @AuditTrailAnnotation
*/
pointcut auditTrail(Object object) : execution(@AuditTrailAnnotation * *(..)) && args(object);
/**
* Encargado de aplicar la funcionalidad de auditoría.
* Invoca al Facade AuditTrail, punto de entrada al componente.
*/
Object around(Object object) : auditTrail(object) {
.....
//me quedo con la entidad a auditar
.....
EntityChange entityChange = null;
try {
//invoco al componente para que realice la lógica de la
//auditoria, retorna los cambios a la entidad.
entityChange = auditTrail.audit(entity,annotation.actionType() ,annotation.action());
} catch (.....) {
.....
}
//procedo con el flujo normal del método
Object objectSaved = proceed(object);
//si la ejecución del método no tuvo inconvenientes, salvo los
//cambios encontrados a la entidad
try {
auditTrail.save(entityChange);
} catch (.....) {
.....
}
return objectSaved;
}
}
```

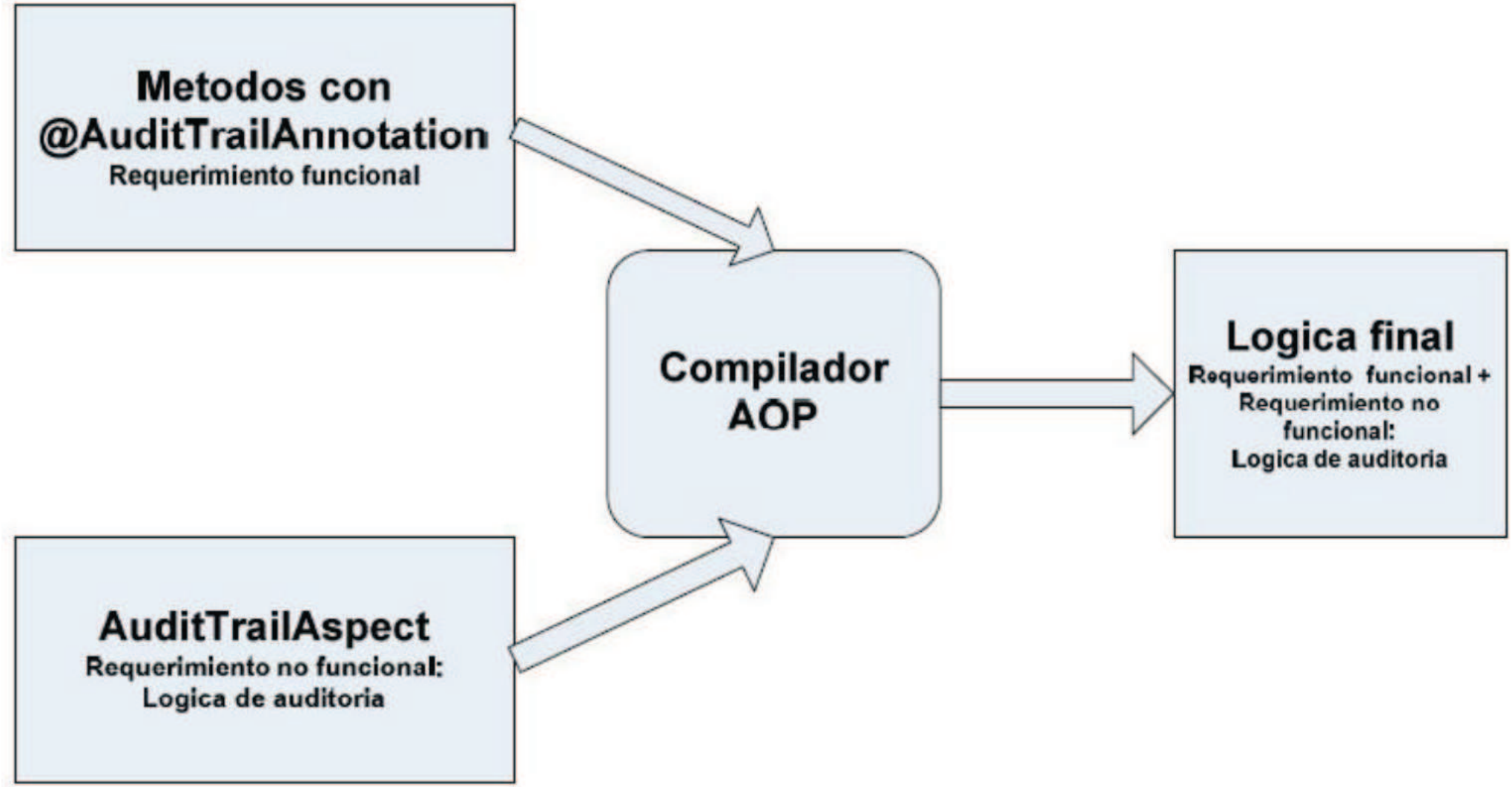
```
package info.unlp.edu.ar.tesis.audit.aspect;
import .....
/**
* Aspecto que se encarga de modelar el requerimiento no funcional de
auditoría de las entidades
*/
public aspect AuditTrailAspect{
/**
* Pointcut definido para que intercepte los métodos que contengan
* el join point métodos con el Annotation @AuditTrailAnnotation
*/
pointcut auditTrail(Object object) : execution(@AuditTrailAnnotation * *(..)) && args(object);
/**
* Encargado de aplicar la funcionalidad de auditoría.
* Invoca al Facade AuditTrail, punto de entrada al componente.
*/
Object around(Object object) : auditTrail(object) {
```

```
pointcut auditTrail(Object object) : execution(@AuditTrailAnnotation * *(..)) && args(object);
/**
* Encargado de aplicar la funcionalidad de auditoría.
* Invoca al Facade AuditTrail, punto de entrada al componente.
*/
Object around(Object object) : auditTrail(object) {
.....
//me quedo con la entidad a auditar
.....
EntityChange entityChange = null;
try {
//invoco al componente para que realice la lógica de la
//auditoria, retorna los cambios a la entidad.
entityChange = auditTrail.audit(entity,annotation.actionType() ,annotation.action());
} catch (.....) {
.....
}
//procedo con el flujo normal del método
Object objectSaved = proceed(object);
//si la ejecución del método no tuvo inconvenientes, salvo los
//cambios encontrados a la entidad
try {
auditTrail.save(entityChange);
} catch (.....) {
.....
}
return objectSaved;
}
}
```

```
@AuditTrailAnnotation(auditedObject="user",
actionType=ActionType.CREATE,action="com.action.create")
public void createUser(User user){
        entityManager.getTransaction().begin(); //open transaction
        entityManager.persist(user);
        entityManager.getTransaction().commit(); //commit transaction
}
```

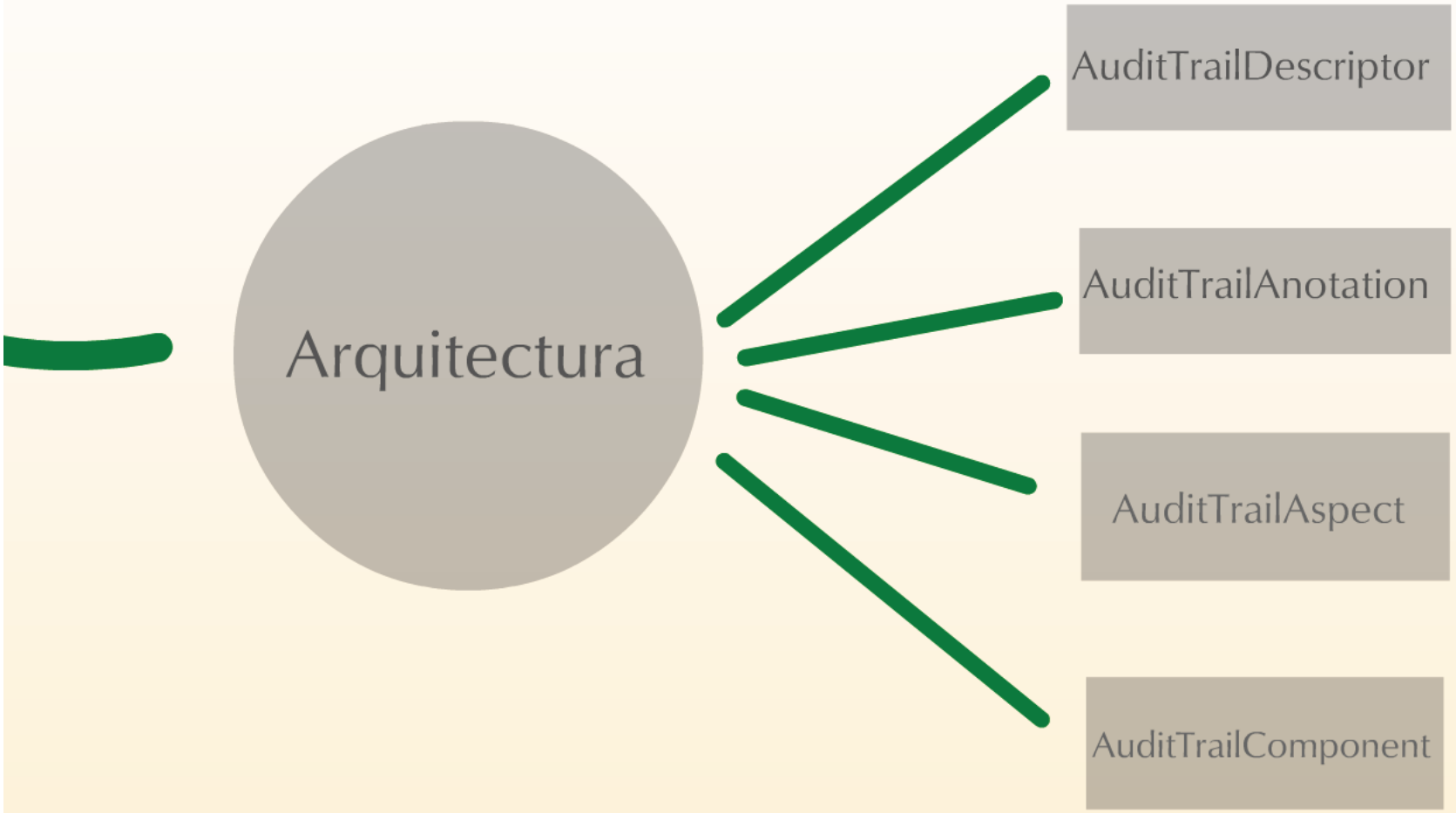
El objeto que debe ser auditado se describe en el elemento auditedObject, el cual hace referencia al nombre del parámetro a auditar.

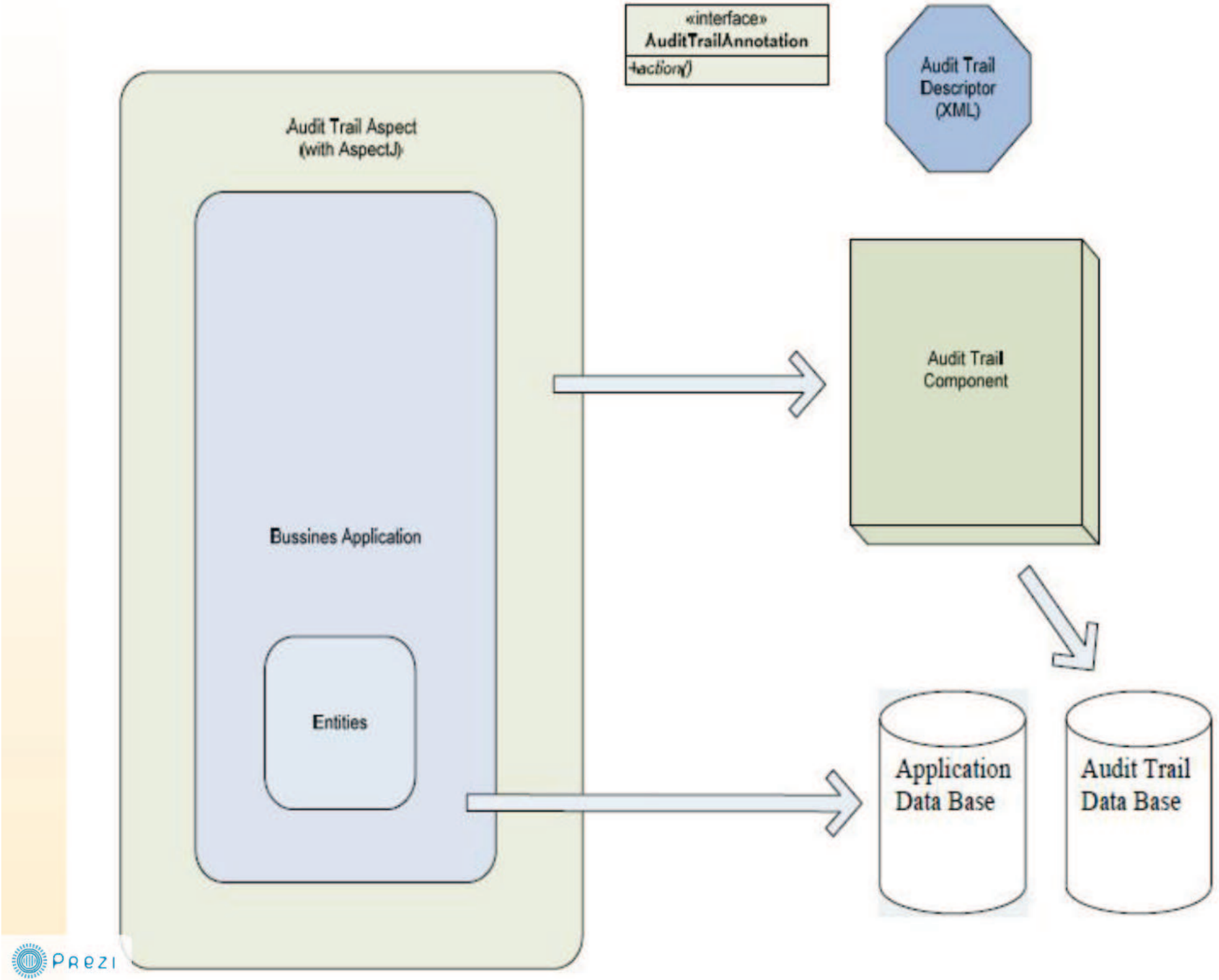
Que pasa cuando realizamos la integración ...


Metodos con
@AuditTrailAnnotation
Requerimiento funcional
AuditTrailAspect
Requerimiento no funcional:
Logica de auditoria
Compilador
AOP
Logica final
Requerimiento funcional +
Requerimiento no
funcional:
Logica de auditoria

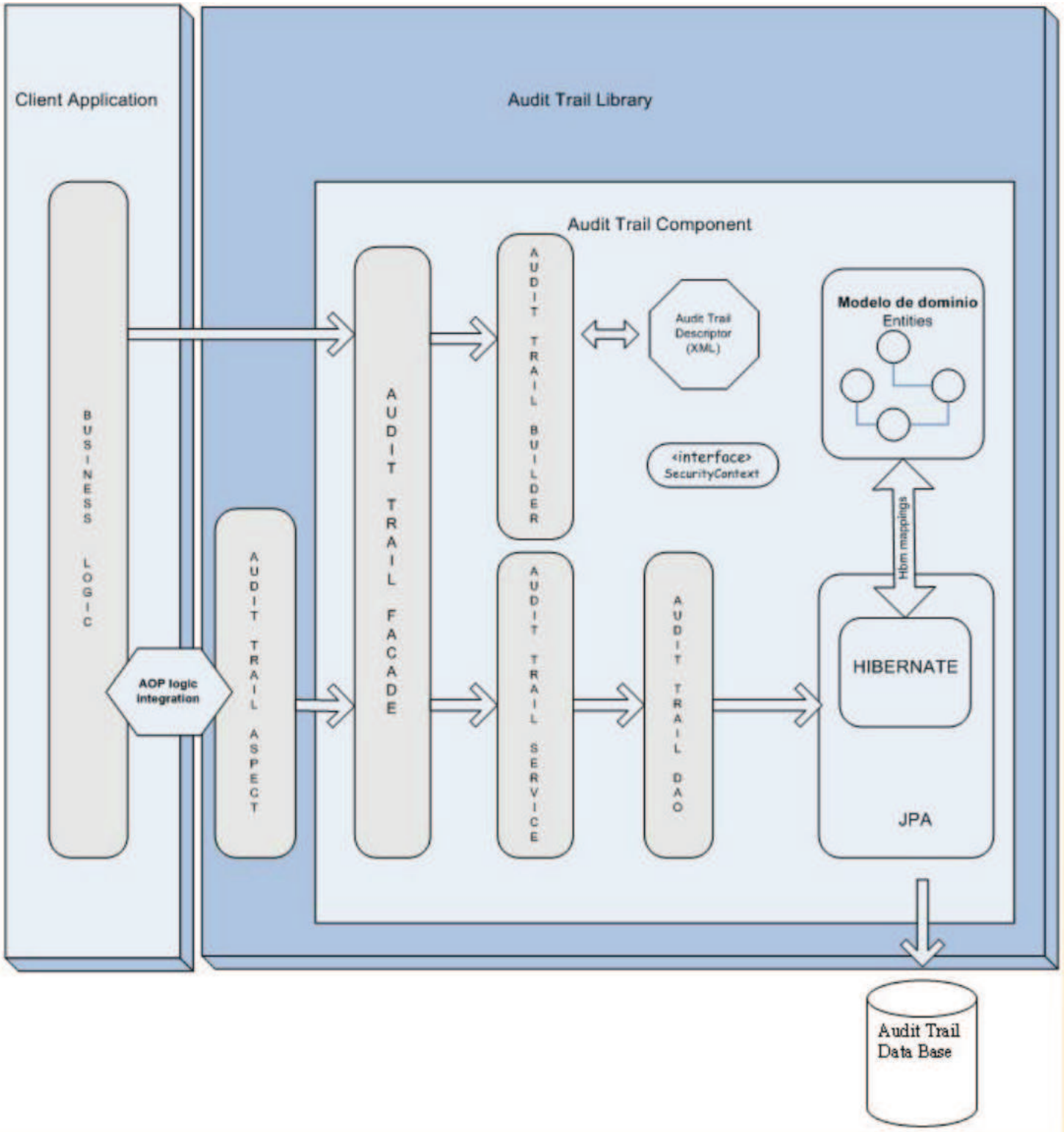
Que pasa cuando realizamos la integración ...

```
public void createUser(User user){

    Object entity=getAuditedObject(); //entity = user
    entityChange = auditTrail.audit(entity, annotation.actionType() ,annotation.action());

    entityManager.getTransaction().begin(); //open transaction
    entityManager.persist(user);
    entityManager.getTransaction().commit(); //commit transaction

    //sino ocurre ningún error, se persiste el AuditTrail
    auditTrail.save(entityChange);

}
```


Arquitectura
AuditTrailDescriptor
AuditTrailAnotation
AuditTrailAspect
AuditTrailComponent


«interface»
AuditTrailAnnotation
+action()
Audit Trail
Descriptor
(XML)
Audit Trail Aspect
(with AspectJ)
Bussines Application
Entities
Audit Trail
Component
Application
Data Base
Audit Trail
Data Base

AuditTrailComponent


Client Application
Audit Trail Library
Audit Trail Component
BUSINESS LOGIC
AOP logic integration
AUDIT TRAIL ASPECT
AUDIT TRAIL FACADE
AUDIT TRAIL BUILDER
AUDIT TRAIL SERVICE
AUDIT TRAIL DAO
Audit Trail Descriptor (XML)
<interface> SecurityContext
Modelo de dominio
Entities
Hbm mappings
HIBERNATE
JPA
Audit Trail Data Base

Agenda

Introducción

Objetivo

Marco Teórico

Análisis Funcional

Alcance de la librería

Diseño

Aplicación de la librería - Testing

Trabajos Futuros

Conclusión

Aplicación de la librería - Testing

Proyecto StaffingProject

LOYAL 5

Aplicación orientada al manejo de la calidad

La misma posee dos módulos principales:

- DMS (Document Management System)
- QMS (Quality Management System)

Rol de la librería AuditTrail ....

Sistema de reglas interactivo

Utiliza un conjunto de reglas que es definido por el cliente, para luego tomar decisiones ....

Por ejemplo aprobar o desaprobar préstamos a personas reales o jurídicas.

Rol de la librería AuditTrail ....

Agenda

Introducción

Objetivo

Marco Teórico

Análisis Funcional

Alcance de la librería

Diseño

Aplicación de la librería - Testing

Trabajos Futuros

Conclusión

Trabajos Futuros

Registrar las veces que un usuario ingreso al sistema

Registrar los accesos a entidades

Monitorear las modificaciones en atributos binarios, en especial documentos de texto

Conclusión

La tecnología orientada a aspectos es la más adecuada para desarrollar los requerimientos no funcionales de los sistemas.

En cuanto al marco teórico, el desarrollo y testeo de la librería AuditTrail cumple con los objetivos de control dispuestos en la norma ISO 27001.

Profesionalmente, el desarrollo de mi tesina me sirvió para implementar soluciones de software innovadoras.

Agenda

Introducción

Objetivo

Marco Teórico

Análisis Funcional

Alcance de la librería

Diseño

Aplicación de la librería - Testing

Trabajos Futuros

Conclusión

Preguntas

¿?

Gracias

UNLP