Tesis final - UNLP

Título: Arquitectura de aplicaciones Enterprise

Alumnos: Matias Butti – Alfredo Fidani

Director: Dr. Gustavo H. Rossi

Temario

- Introducción
  - Problema
  - Definición de arquitectura
  - Actividades del equipo de arquitectura
- Aplicaciones Enterprise
- Aportes más importantes
  - Metodología propuesta
  - Técnico: Definción e implementación de la Arquitectura propuesta para el caso de estudio

Introducción

- Problema
  - Es frecuente en la industria del software, que el alcance de la arquitectura se restrinja a la definición de tecnologías a utilizar
  - De esta manera, los responsables de la arquitectura de software suelen trabajan solo en etapas previas al comienzo del desarrollo y con una definición vaga de la arquitectura
  - ¿Por qué consideramos que esta visión de la arquitectura no es adecuada para aplicaciones de tipo Enterprise? Porque:
    - La definición de la arquitectura no solo debe abarcar tecnología sino también definiciones conceptuales y, principalmente, pautas de uso de la arquitectura y de los frameworks elegidos
    - El trabajo del equipo de arquitectura no concluye en la defínción, sino que incluye una implementación y más aún, acompañamiento a lo largo del desarrollo
  - El efecto negativo que produce este problema en aplicaciones pequeñas puede no ser alto, pero en aplicaciones Enterprise puede ser la causa del fracaso del proyecto: poco reuso, baja productividad, alto acomplamiento a tecnologías, dificultad de mantenimiento y extensión, entre otros
  - Sobre la base de este problema, formulamos una definición de arquitectura sobre la que basamos nuestra tesis :"Soporte necesario para la construcción de los casos de uso, que ayude a garantizar el cumplimiento de los requerimientos no funcionales, asegure la calidad y maximice la productividad"

Actividades del equipo de arquitectura

- Tareas que consideramos, deben ser responsabilidad de un equipo de arquitectura (la metodología ordenará estas actividades)
  - Definición conceptual de la arquitectura
  - Evaluación de frameworks y herramientas existentes en la comunidad que puedan ser de utilidad en el proyecto
  - Pautas de diseño y desarrollo sobre la arquitectura, que garanticen el cumplimiento de los requerimientos no funcionales, aseguren la calidad del producto y permita conseguir buena productividad en el desarrollo.
  - Diseño y desarrollo
    - …de frameworks que resuelven problemas comunes (alineadas con la arquitectura que se haya definido)
    - …de frameworks que abstraen funcionalidad común
    - …de herramientas que mejoren la productividad del desarrollo
  - Actualizaciones tecnológicas
  - Soporte técnico a los desarrolladores
  - Capacitación al equipo de desarrollo
  - Resolución de problemas de performance

Aplicaciones Enterprise

- ¿Qué es una Aplicación Enterprise?
  - *Herramienta de soporte para las organizaciones*
  - *Relacionan información*
  - Ayudan a cumplir los objetivos de la Organización

- ¿Cuál es el objetivo de una Aplicación Enterprise?
  - *Automatización de los procesos de negocio de la organización*
  - *Brindar soporte funcional a las organizaciones con el objetivo de mejorar la productividad y la eficiencia de las actividades de la misma*

- Centramos el estudio en las aplicaciones que modelan procesos administrativos

Aplicaciones Enterprise
Requerimientos funcionales de alto nivel

- *La información debe estar integrada*
  - Definición de los procesos en forma centralizada - centralización normativa
  - Estandarización de la información de uso común
- *Descentralización operativa*
- *Multimoneda*
- *Manejo de objetos del negocio de gran tamaño*
- *Log de operaciones*
  - Auditorias, controles y seguimientos
- *Definición de perfiles y niveles de información*
  - Perfiles y dominios de información

Aplicaciones Enterprise
Requerimientos no funcionales

- Performance
  - Tiempo de respuesta
  - Tiempo de respuesta al usuario
  - Tiempo de latencia
  - Throughput
- Capacidad y Escalabilidad
- Usabilidad
  - *Categorías de usuarios*
- Adaptabilidad al cambio de reglas de negocio
- Portabilidad a través de plataformas
- Seguridad
  - Contraseña única (Single Sign-On)
  - Administración centralizada de usuarios
  - Encriptación (técnicas de criptografía)
- Alta Disponibilidad
  - 7 por 24 -> transaccional

Aplicaciones Enterprise
Requerimientos no funcionales

- Mantenibilidad
  - Separación de capas de Presentación y Negocio
- Hot deploy
  - Contemplado en la arquitectura
- Autonomía del usuario
  - Definición de circuitos propios
  - Datawarehouse
- Descentralización de operaciones
  - Parametrización de funciones (Habilitar y deshabilitar funciones)
- Modos de operación
  - Normal
  - Mantenimiento
- Uso de Red
- Facilidad de Instalación
  - Para distintos ambientes de desarrollo
  - Producción
- Integración con sistemas legados

Aplicaciones Enterprise y no Enterprise

Metodología

- Definición de un equipo de arquitectura y sus actividades
  - Definir el marco teórico de arquitectura
  - Definir las tecnologías que se utilizarán en cada uno de los elementos del marco de arquitectura definidos
  - Actualización de las tecnologías y frameworks utilizados
  - Evaluación de frameworks existentes en la comunidad que puedan ser de utilidad en el proyecto
  - Evaluación de herramientas
  - Diseño y desarrollo de componentes y frameworks que resuelvan los problemas comunes de las aplicaciones Enterprise (componentes estructurales)
  - Diseño y desarrollo de componentes y frameworks que abstraen conceptos comunes del negocio (componentes funcionales)
  - Soporte al equipo de desarrollo
- Metodología
  - Organizar las actividades del equipo de arquitectura describiendo el alcance de cada una
  - Dividida en etapas de insumo - producto

Metodología propuesta

- Etapa 1: Marco de arquitectura
- Etapa 2: Generalización funcional
- Etapa 3: Extensión, corrección y mantenimiento

Metodología propuesta – Etapa 1 Marco de arquitectura

- ¿A que da respuesta el marco de la arquitectura?
  - Como reusar componentes gráficas
  - Como transferir objetos del servidor al cliente y viceversa
  - Como generar reportes
  - Como particionar el sistema en varios subsistemas. Esta partición favorece el desacoplamiento entre los mismos, evitando los efectos colaterales en el mantenimiento del sistema. Un subsistema ve a otro como una caja negra a través de una interfaz bien definida sin preocuparse por su implementación interna
  - Como distribuir los elementos que componen un caso de uso sobre la arquitectura
  - Como manejar la seguridad
  - Como realizar la persistencia
  - Como desarrollar test de unidad, test de integración y test funcionales
  - Como realizar la automatización de test de regresión
  - Como interactuar con otros sistemas
  - Como usar la tecnología y los frameworks
  - Como manejar errores
  - Como generar y consultar el log operacional
  - Como generar y visualizar la historia
  - Como generar información de auditoría

Metodología propuesta – Etapa 1
Marco de arquitectura

- Marco teórico
  - Definición
  - Documento de visión
    - Propósito del sistema
    - Descripción de los interesados y usuarios
    - Descripción global del sistema
    - Características funcionales del producto
    - Restricciones
  - A considerar
    - No existe la arquitectura Silver Bullet
    - Competencia entre atributos de calidad
    - Impacto de modificaciones

Metodología propuesta – Etapa 1 Marco de arquitectura

- Marco teórico (cont.)
  - Definir la estructura global y conceptual del sistema apoyándose en Estilos arquitectónicos y patrones
- Primera implementación de la arquitectura
  - Evaluación de herramientas
  - Diseño y desarrollo de las primeras versiones

Metodología propuesta – Etapa 1
Marco de arquitectura

- Documentación
  - Arquitectura de aplicación
  - Guía de uso
  - Guía de solución
- POC: Implementación de referencia
  - Valida la correctitud de la solución de manera completa
- Prueba de carga
  - Pruebas de stress y performance automatizadas

Artefacto generado: Arquitectura estructural

Metodología propuesta – Etapas 2 y 3

- Etapa 2: Generalización funcional
  - Existencia de un equipo de Arquitectura funcional
  - Artefacto generado: Arquitectura funcional básica
- Etapa 3: Extensión, Corrección y mantenimiento de la arquitectura
  - Artefacto generado: Arquitectura terminada

Metodología propuesta – Resumen

- Etapa 1: Marco de arquitectura
  - Insumo: LBR
  - Artefacto generado: Arquitectura Estructural
- Etapa 2: Generalización Funcional
  - Insumo: AE, CU más significativos modelados
  - Artefacto generado: Arquitectura funcional básica
- Etapa 3: Extensión, corrección y mantenimiento
  - Insumo: Necesidades de los demás equipos
  - Artefacto generado: Arquitectura terminada

Caso de estudio

- E-Sidif: Sistema de formulación del presupuesto Nacional y registro de la ejecución presupuestaria

- Características
  - Gran tamaño del sistema (14000 UCP)
  - Gran cantidad de usuarios concurrentes
  - Lógica de negocio muy compleja y evolutiva
  - Integración con otros sistemas
  - Seguridad con reglas particulares
  - Clientes: Organismos distribuidos a lo largo del país
  - Necesidad de facilidad de instalación
  - Mantenibilidad
  - Alta Performance
  - Reportes
  - Workflow
  - OLAP

Arquitectura e-sidif: Una visión de alto nivel

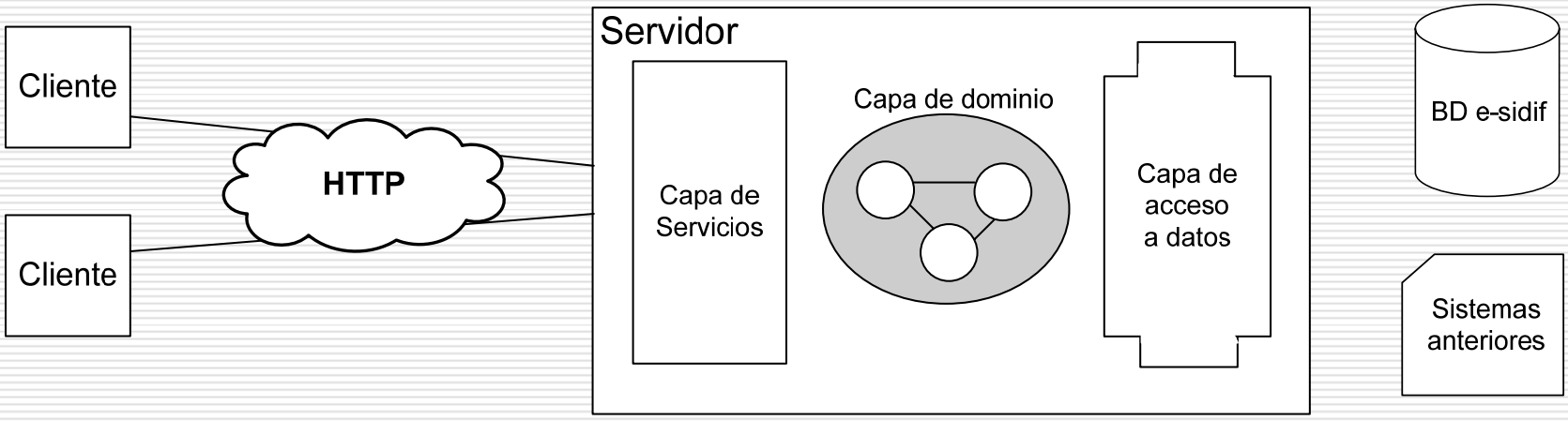
- Arquitectura con estilo N-layered
- Cliente – servidor

Servidor

- Capa bien marcada de Servicios
- Modelo de dominio
- Capa de acceso a datos
- Fuente de información: BD relacional, sistemas anteriores

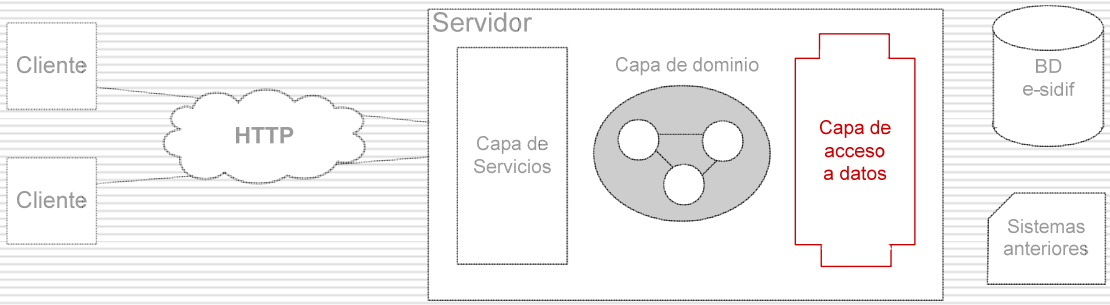
Cliente standalone

- Comunicación con el servidor via HTTP
- Y por qué no web?
- Desarrollo orientado a componentes reusables!
- Distribución via JNLP



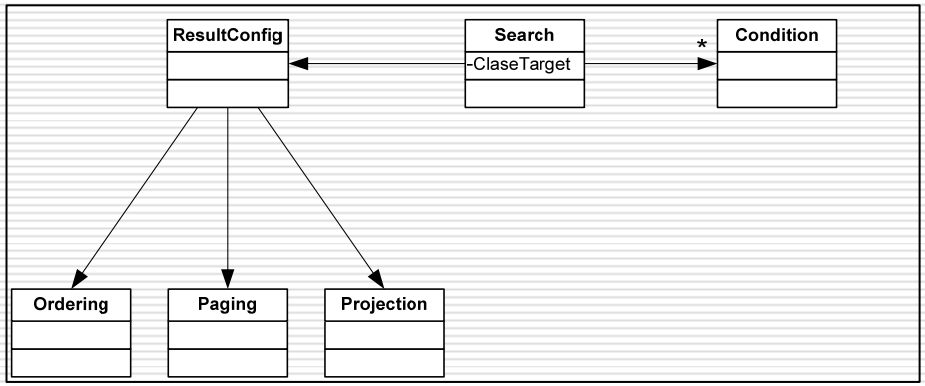
Arquitectura e-sidif: Capa de acceso a datos

- Capa de acceso a datos
  - Brinda acceso a los objetos persistidos de la aplicación
  - Es consumida por la capa de servicio y, en ciertos casos, por el modelo de dominio
  - Wrapea al mapeador objeto relacional Hibernate
  - Repositorio
    - Save(objeto)
    - Modify(objeto)
    - Delete(objeto)
    - Retrieve (id)



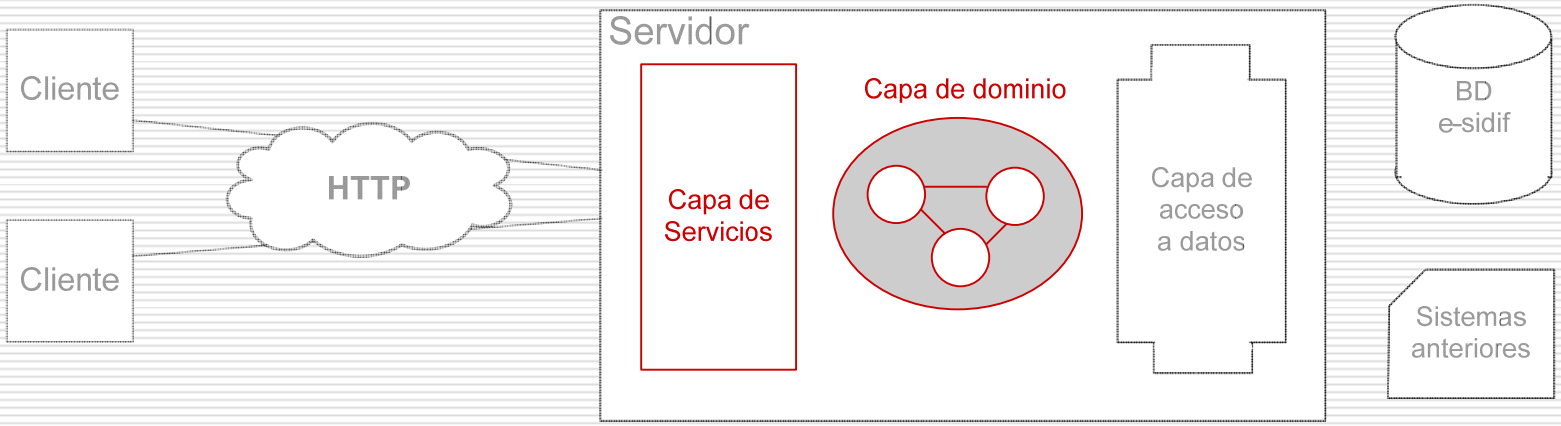
Arquitectura e-sidif: Capa de acceso a datos

- Capa de acceso a datos - Búsquedas
  - Un método adicional del repositorio es el >>Select(Search)



  - Las consultas propias de los casos de uso, están ordenadas en SearchFactories
  - Quien quiera ejecutar la consulta, pedirá al SearchFactory la construcción del search y pedirá al repositorio que lo ejecute a través del >>select
  - Retorna la lista de objetos resultantes. Soporta otros formatos de retorno: XML, ResultSet
  - Soporta escritura de código SQL

Arquitectura e-sidif: Capa de servicios



Arquitectura e-sidif: Capa de servicios

- Capa de servicios: Determina el conjunto de operaciones que brinda la capa de negocios de la aplicación

- Ejemplos de servicios
  - Alta "Escenario de Simulación de presupuesto"
  - Baja "Escenario de Simulación de presupuesto"
  - Modificación "Escenario de Simulación de presupuesto"

  → Servicios de ABM

  - Distribuir el crédito

  → Servicios que implementan algoritmos

  - Poner a la firma una modificación presupuestaria
  - Firmar una modificación presupuestaria
  - Autorizar una modificación presupuestaria

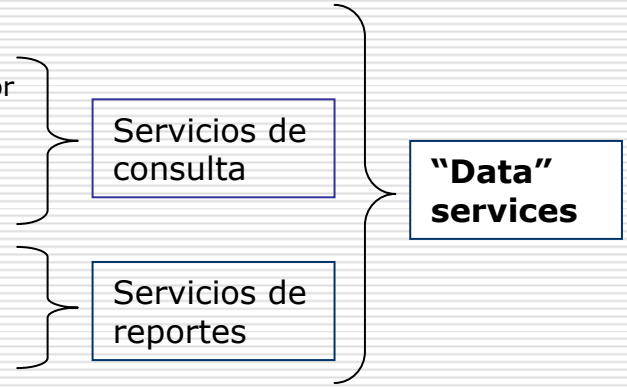
  → Servicios de transición de estado en un workflow

→ **Servicios "Transactionales"**

Arquitectura e-sidif: Capa de servicios

- Capa de servicios: Determina el conjunto de operaciones que brinda la capa de negocios de la aplicación

- Ejemplos de servicios
  - Recuperar el escenario de presupuesto, dado su identificador
  - Recuperar los escenarios de presupuesto que cumplan ciertas condiciones ingresadas por el usuario.
  - Generar un reporte de Listado de escenarios por etapa para exportar a PDF, Excel o imprimir



Capa de servicios

Servicios de ABM
- Alta ESP
- Baja ESP
- Modificación ESP

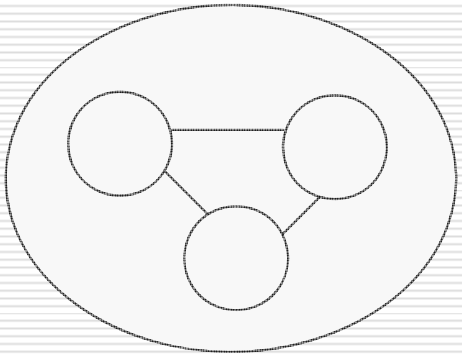
Servicios que implementan algoritmos
- Distribuir el crédito

Servicios de transición de estados en un workflow
- Poner a la firma una MP
- Firmar una MP
- Autorizar una MP

Servicios de consultas
- Recuperar el ESP a partir de su identificador
- Recupera los ESP que cumplan con ciertas condiciones ingresadas

Servicios de reportes
- Generar un reporte de listado de ESP por etapa para exportar a PDF, Excel o imprimir

Capa de dominio



Capa de acceso a datos

BD e-sidif

Sistemas anteriores

ESP = Escenario de simulación de presupuesto
MP = Modificación presupuestaria

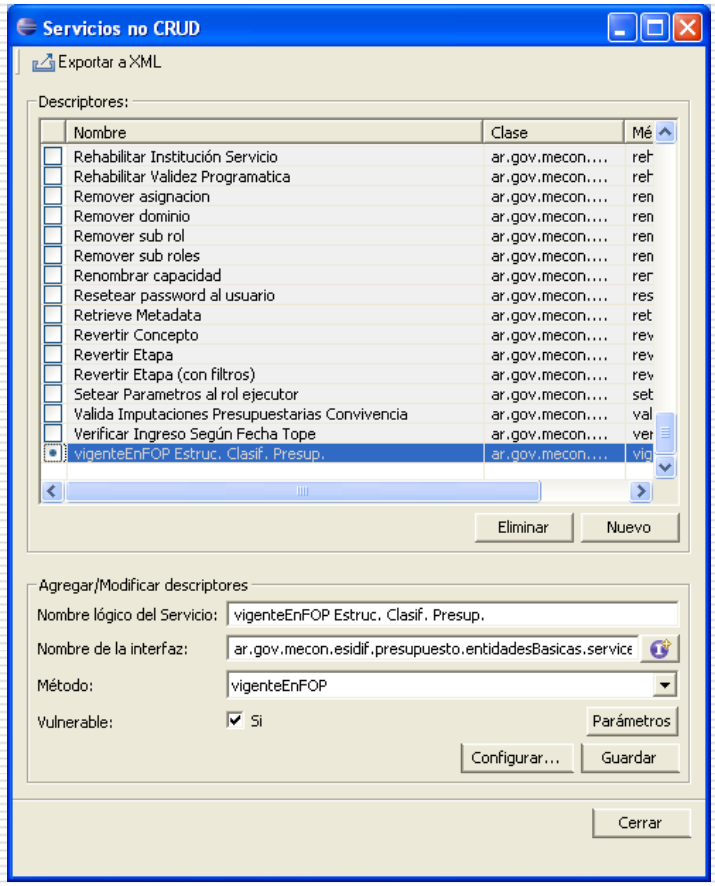
Arquitectura e-sidif: Capa de servicios

- El hecho de disponer de una capa bien marcada de servicios nos permite
  - Manejar la seguridad (autorización) a nivel de servicio
  - Mapeo bastante directo de los casos de usos al diseño y la implementación.
  - Manejar las precondiciones a nivel de servicio (como se especifica en el análisis)
  - Tener una API clara de las operaciones que brinda la capa de dominio
  - Bajo acoplamiento entre el servidor y el cliente. Se ocultan detalles de implementación, simplificando el mantenimiento y posibiltando refactors.
  - Manejar el resto de la lógica de aplicación a nivel servicio: log, concurrencia, transaccionalidad, remoting

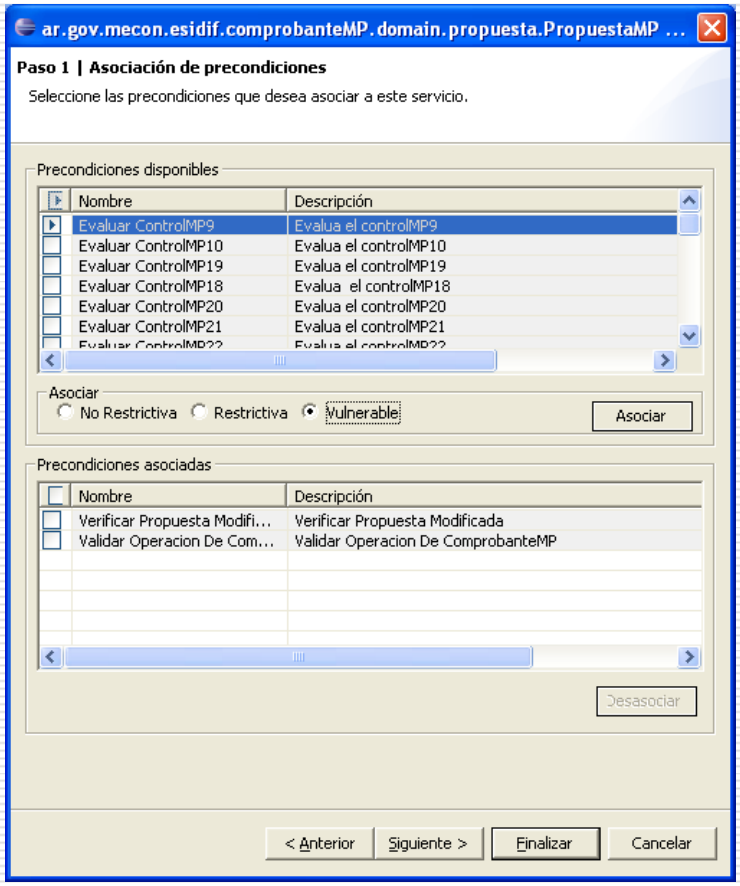
Arquitectura e-sidif - CoSE: Configuración de servicios

- Pensamos que sería útil desarrollar una herramienta que permita
  - Describir el servicio
  - Crear el permiso del servicio
  - Asociar precondiciones indicando su grado de restricción
  - Exportar e Importar la configuración entre ambientes

CoSE



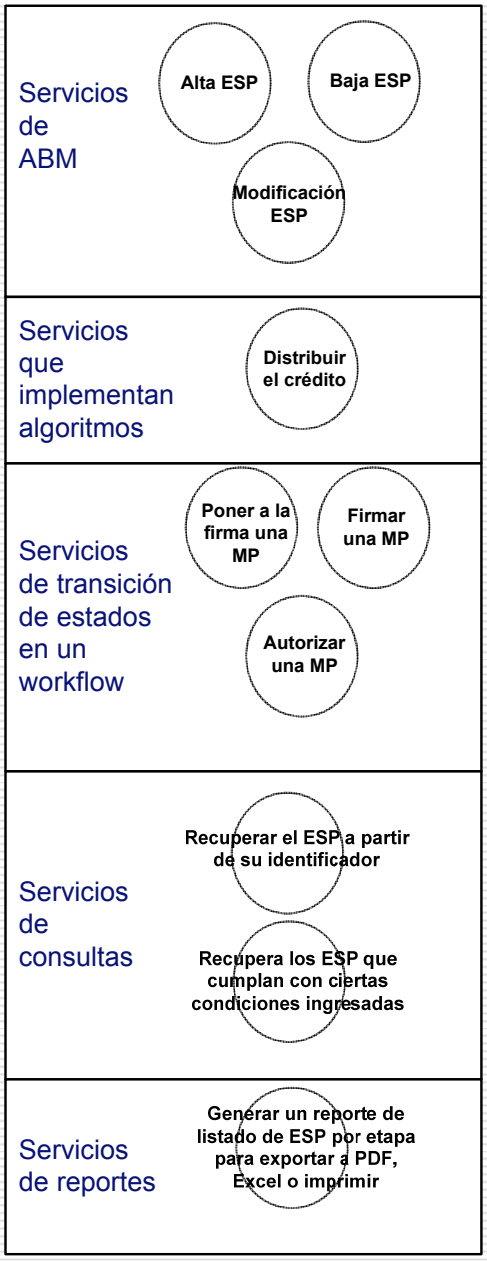
Arquitectura e-sidif - CoSE: Configuración de servicios

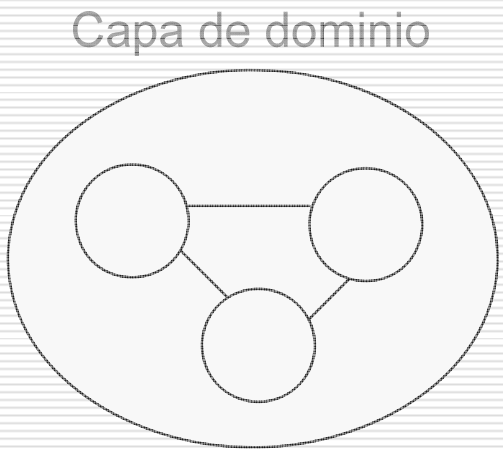


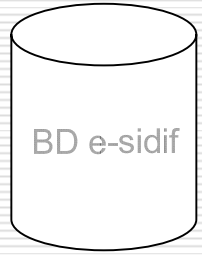
Ejemplo: Servidor – Creación de nuevos servicios

- Pasos para la creación de un servicio
  - Implementación del servicio (Varía dependiendo del tipo de servicio)
    - Elementos java
    - Lógica de aplicación (seguridad, log, concurrencia, transaccionalidad, remoting)
  - Clase de Precondiciones
  - Declaración de servicios con CoSE
    - Tipo servicio
    - Elementos java que lo identifican
    - Nombre lógico
  - Asociación de precondiciones con CoSE
- Uno de los trabajos futuros es que el CoSE genere los elementos de la implementación (Herramienta Case de servicios)

Capa de servicios







Arquitectura e-sidif: Implementación del servicio

- Implementación del servicio
  - Lógica de dominio
  - Lógica de aplicación

Arquitectura e-sidif: Implementación del servicio – Lógica de dominio

- Implementación de la lógica de dominio en un servicio (basado en la implementación de la arquitectura)
  - Algoritmos
    - Se implementa como un objeto que coordina la operación, interactuando con la capa de acceso de datos y delegando en los objetos de dominio
  - ABM:
    - Genérico. No es necesario implementar el servicio
    - Solo si es necesario, se define comportamiento para antes y después de las operaciones a través de un objeto que coordina ese comportamiento. En tal caso, el comportamiento se delega en la capa de dominio
  - Transiciones
    - Para cada comprobante deben definir su workflow: estado, transiciones, acciones adicionales en las transiciones
    - Cada servicio de transición será una transición del workflow de algún comprobante
  - Consulta: Genérica
    - El cliente tiene que mandar un search o un id
  - Reportes
    - Framework de reportes (fijos y variables)

Arquitectura e-sidif: Implementación del servicio – Lógica de aplicación

- Si ya escribí toda la funcionalidad, para qué agregar esto de… la lógica de aplicación?

- Múltiples usuarios concurrentes, acceso remoto, distintos clientes, reglas de negocio complicadas generan la necesidad de administrar la lógica de aplicación en el servicio
  - Seguridad
    - Autenticación (LDAP)
    - Autorización (lee la configuración de CoSE)
    - VPD
  - Validación (lee la configuración de CoSE)
    - Restrictiva, informativa, Vulnerable
  - Manejo de Concurrencia: 49 casos distintos
    - Edición cabecera, edición ítems, transiciones, se puede editar concurrentemente ítems?

Arquitectura e-sidif: Implementación del servicio – Lógica de aplicación

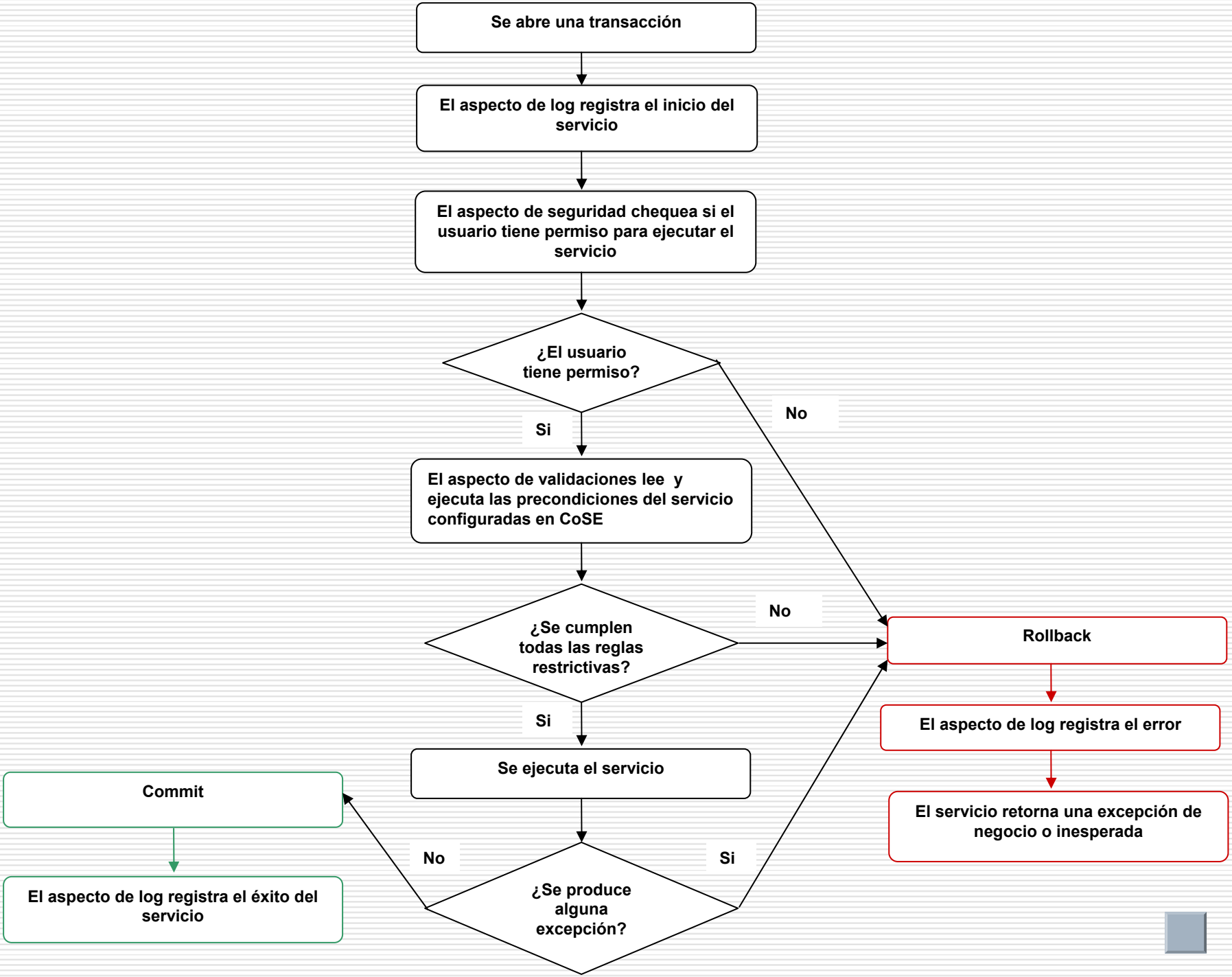
- Transaccionalidad (solo para los servicios transaccionales)
  - Causas de rollback
    - Validación restrictiva que falla
    - Excepción inesperada que llega al servicio
    - Excepción de negocio que llega al servicio
- Remoting
  - Cada servicio puede ser ejecutado via HTTP, usando serialización Java
  - HTTP exporter de Spring
- Manejo de errores
  - Se debe asegurar que desde el servicio no salga una excepción diferente a ServiceException o UnExpectedException
- Interoperabilidad
  - Web services

Arquitectura e-sidif: Implementación del servicio – Lógica de aplicación

- Cada una de estas necesidades las podemos considerar aspectos que aparecen en todos los servicios (cross-cutting concerns)

- AOP (Implementación de Spring)
  - Implementación de los aspectos
  - Agregado de aspectos a los servicios de manera declarativa

- El desarrollador del servicio debe indicar que "aspectos" quiere agregar a sus servicios (algunos están agregados por defecto)


Se abre una transacción
El aspecto de log registra el inicio del servicio
El aspecto de seguridad chequea si el usuario tiene permiso para ejecutar el servicio
¿El usuario tiene permiso?
No
Si
El aspecto de validaciones lee y ejecuta las precondiciones del servicio configuradas en CoSE
¿Se cumplen todas las reglas restrictivas?
No
Si
Se ejecuta el servicio
¿Se produce alguna excepción?
No
Si
Commit
El aspecto de log registra el éxito del servicio
Rollback
El aspecto de log registra el error
El servicio retorna una excepción de negocio o inesperada

Arquitectura e-sidif: Cliente

- Ambiente de la aplicación basado en RCP (editores, vistas, perspectivas)
- Objetos en el cliente
  - Dyto
    - Implementación de DTO
    - Lazy
    - Paginado de colecciones
    - Sincronización de datos con el objeto del lado del server
- Desarrollo de las GUIs
  - Orientado a componentes - Mejora de productividad
  - Cada componente se desarrolla asumiendo como modelo un objeto de una clase determinada. De esta manera se pueden hacer los bindings entre los widgets y los atributos del objeto
  - Cuando se usa esa componente, solo se debe indicar como conseguir el objeto que espera esa componente

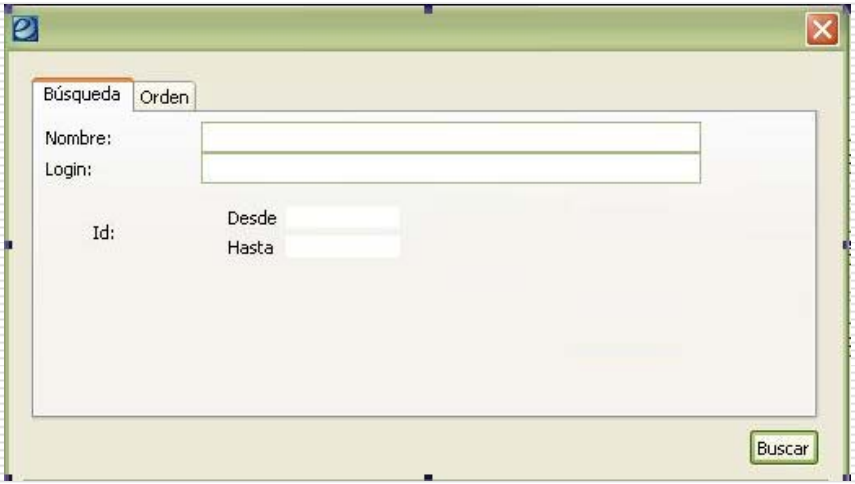
Arquitectura e-sidif: Cliente – Ejecución de servicios

- Invocación de servicios del lado del cliente
  - GenericServiceFactory.getService()

- Framework de acciones
  - El caso de uso instancia acciones
    - Las acciones indican lo que tienen que hacer cuando se ejecutan
    - Las acciones tienen condiciones (la de seguridad está agregada para todas las acciones)
    - Las acciones indican donde se deben pintar (menú, menú contextual, toolbar)
    - Las acciones se agregan en contenedores (vistas, editores, diálogos) que, cuando ganan foco, activan las acciones que contienen
    - Existen acciones con lógica predefinida
      - AccionDeEjecuciónDeABM(Class, operacion)
      - AcciónDeEjecuciónDeAlgoritmo (Interfaz, Metodo)
      - AcciónDeEjecuciònDeTransición(STE, transicion)
      - AcciónDeEjecuciónDeConsulta(STE, transicion)

Arquitectura e-sidif: Cliente – Formularios de búsqueda

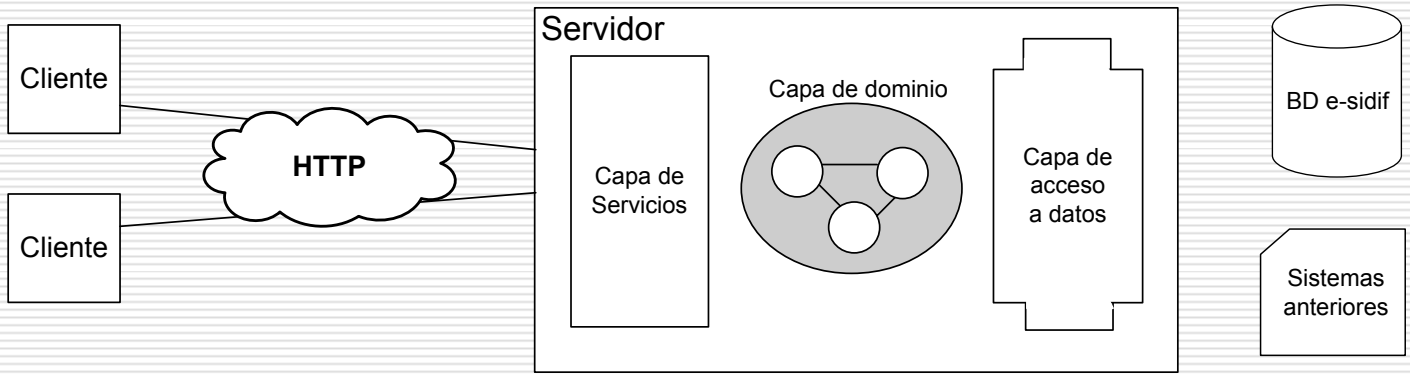
- Formularios de búsqueda
  - Framework de formularios de búsqueda
  - El desarrollador solo brinda el "panel" con los widgets para que el usuario ingrese los valores para las condiciones
  - El framework genera el objeto Search, ejecuta la acción de búsqueda, toma el resultado (conjunto de objetos o dytos), lo muestra en la vista de resultados



Arquitectura e-sidif: Frameworks de la implementación

- Estructurales
  - Repositorio
  - Servicios + Herramienta CoSE
  - Reportes
  - Seguridad
  - Asincronismo
  - Dyto
  - Formularios de búsqueda
  - Cliente rico (reuso de componentes gráficos)
  - Escritorio

- Funcionales
  - Workflow
  - Comprobantes
  - Cadena de firmas



Conclusiones

- Definición de arquitectura propuesta: “Soporte necesario para la construcción de los casos de uso, que ayude a garantizar el cumplimiento de los requerimientos no funcionales, asegure la calidad y maximice la productividad“

- Para conseguirlo es necesario:
  - Disponer en la etapa inicial del proyecto – antes de comenzar la implementación o construcción del sistema - de un equipo que defina la arquitectura del sistema para cubrir los requerimientos no funcionales y dar soporte a los funcionales.

  - Durante el desarrollo mantener ese equipo para que brinde una implementación de la arquitectura y haga evolucionar la misma, de manera tal que se logre:
    - Estandarizar el desarrollo
    - Simplificar el futuro mantenimiento
    - Mejorar la productividad del equipo de implementación
    - Mejorar la calidad del producto generado

- La misma por ser una **actividad** continua durante el proceso de desarrollo de un proyecto de software necesita, durante todo este proceso, de una metodología que le permita desarrollar estas actividades

Trabajo futuro

- Convertir al CoSE en CASE

- Tesis en las que se está empezando a trabajar
  - Asincronismo
  - Formularios de búsqueda
  - Framework de servicios y CoSE
  - DyTO
  - Problemas de performance en el uso de mapeadores objeto/relacional en aplicaciones con grandes volúmenes de datos

Gracias por su atención!

Preguntas?