UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

GUÍA ARQUITECTÓNICA PARA EL DESARROLLO DE APLICACIONES

REPORTE DE TRABAJO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

ARTURO VAZQUEZ CORONA

TUTORA:
DRA ELISA VISO GUROVICH
2010

Contenido

1 Introducción 3
2 Sobre el proyecto 4
2.1 Justificación 4
3 Consideraciones generales 6
3.1 Consideraciones de confidencialidad 6
3.2 Integridad del trabajo original 6
3.3 Contexto al trabajo presentado 7
3.3.1 Requerimientos 7
3.3.2 Productos relacionados 8
4 Experiencia adquirida 10
5 Estado actual 11
6 Conclusiones 12
Anexo A: Guía arquitectónica para el desarrollo de aplicaciones 13
Anexo A.I: Introducción 13
Anexo A.II: Visión Arquitectónica 14
Anexo A.II.I: Metas 14
Anexo A.II.II: Panorama objetivo 15
Anexo A.II.III: Alcance 17
Anexo A.III: Vista arquitectónica estructural 18
Anexo A.III.I: Módulos funcionales 19
Anexo A.III.II: Clasificación de componentes 20
Anexo A.III.III: Niveles 20

Anexo A.III.IV: Capas en el nivel de aplicación....21
Anexo A.III.V: Componentes aplicativos....23
Anexo A.IV: Lineamientos arquitectónicos....25
Anexo A.IV.I: Módulos simples....25
Anexo A.IV.I.I: Modelo de datos específico....26
Anexo A.IV.I.II: Modelo del problema....27
Anexo A.IV.I.III: Componente de persistencia....29
Anexo A.IV.I.IV: Capa de lógica....32
Anexo A.IV.I.V: Interfaces de usuario....33
Anexo A.IV.II: Integración de módulos....35
Anexo A.IV.III: Módulos como servicios....39
Anexo A.IV.IV: Módulos transitorios....43
Anexo A.IV.V: Un caso extremo....45
Anexo A.V: Desarrollo futuro....46
Anexo A.V.I: Escenarios previstos....46

1 Introducción

El presente documento forma parte del proyecto que con el objeto de obtener el grado de Licenciado en Ciencias de la Computación presenta el que suscribe, Arturo Vazquez Corona. El trabajo tiene como nombre “Guía arquitectónica para el desarrollo de aplicaciones” y fue desarrollado como parte de mis labores como Director de tecnologías en Root Technologíes S.C.

Este documento tiene como objetivo integrar los aspectos accesorios del trabajo que para presentarlo como proyecto de titulación deben llevarse a cabo. Estos aspectos incluyen pero no se limitan a:

- La definición del trabajo realizado, incluyendo el contexto en el cual se realizó, motivaciones y alcances.
- Notas sobre la confidencialidad y modificaciones que para ser presentado se realizaron al trabajo original.
- Conclusiones y resumen de la experiencia resultado del trabajo profesional.
- Notas sobre cómo trasladar la experiencia de este trabajo a otros contextos.

Se sugiere al lector avanzar sobre este documento de manera secuencial hasta el final de la sección 3, Consideraciones generales, suspender la lectura y reanudarla una vez que se haya leído la *Guía arquitectónica para el desarrollo de aplicaciones* que el lector podrá encontrar como Anexo A.

2 Sobre el proyecto

La *Guía arquitectónica para el desarrollo de aplicaciones* forma parte de un proyecto más amplio para la definición de los aspectos comunes que surgen en el desarrollo de aplicaciones y, como tal, me permito exponer a continuación el contexto y motivación que llevó a desarrollar este trabajo.

Durante los últimos años Root Technologies se ha desempeñado como proveedor de sistemas de importantes actores en el ámbito de seguridad nacional, participando el que suscribe en la mayoría de estos proyectos, en un papel de arquitecto de sistemas de información y asesor en la toma de decisiones. En una de estas entidades, cuyo nombre reservamos por motivos de confidencialidad, se encomendó a Root, y en particular a mi persona, el desarrollo de una *Arquitectura Tecnológica Institucional*[1] que sirviera de marco para el proceso de evolución que las aplicaciones sustantivas de dicha institución llevarían a cabo.

El proyecto completo tiene como fin proveer las herramientas necesarias para llevar una gestión y control del desarrollo de sus aplicaciones en el área de tecnología, alineándolas a procesos y guías que aseguren su calidad, empleando las mejores prácticas del mercado, siempre y cuando éstas se adecuen a las necesidades de la organización.

Dentro del marco de este proyecto se desarrollaron diversas líneas de acción, en cuatro rubros principales:

1. Implementar mejores prácticas en el desarrollo de sistemas.
2. Fortalecer la seguridad de las aplicaciones y servicios.
3. Garantizar la correcta gestión de la información.
4. Actualizar las aplicaciones institucionales.

En el rubro de mejores prácticas en el desarrollo de sistemas una parte fundamental es la “Guía arquitectónica para el desarrollo de aplicaciones”, la cual tiene como objetivo lograr una integración de los sistemas existentes y desarrollo futuros, así como el planteamiento de lineamientos, tanto conceptuales como específicos, orientados a una arquitectura de procesos y servicios.

2.1 Justificación

El documento que se presenta es no sólo parte fundamental del proyecto realizado por la empresa para uno de nuestros clientes, sino que tiene características que lo hacen

1 En la industria esto se conoce como una arquitectura empresarial (Enterprise Architecture), sin embargo dado que se desarrolló en un ambiente de sector público el nombre no es adecuado y se optó por definir una arquitectura institucional. La palabra tecnológica se añade para claridad del alcance de la misma.

particularmente adecuado para el fin de demostrar los elementos y tecnologías a tomar en cuenta para la práctica profesional. En esta sección presentaremos estas características junto con observaciones de su relevancia.

Considero que el trabajo es relevante en términos del perfil del egresado que la Licenciatura en Ciencias de la Computación tiene, debido a que éste presenta lineamientos que van más allá de la práctica de programación y que incluyen elementos de diseño orientado a objetos, ingeniería de software y arquitectura de sistemas distribuidos; todo lo anterior con consideraciones de ámbito general como seguridad, eficiencia, escalabilidad y protección de la inversión; así como particulares respecto a los objetivos y prioridades institucionales que la organización receptora de dicho trabajo tiene.

Finalmente, la guía arquitectónica para el desarrollo de aplicaciones, si bien fue desarrollada con objetivos y alcances particulares, es de espectro suficientemente general para presentarse por separado del resto de los productos del proyecto por su relevancia para los fines de ilustrar lo que constituyen buenas prácticas de análisis, diseño y programación de sistemas de software,, eliminando los aspectos que pudieran ser sujeto de debates de confidencialidad sin perder la esencia del trabajo.

3 Consideraciones generales

En esta sección presentaremos algunas notas con respecto al documento que se presenta como muestra de la experiencia profesional. Inicialmente, se presentan algunas notas referentes a los cambios realizados al documento principal, en su mayoría por motivos de confidencialidad. En la sección 3.2 se presentan las razones para integrar este documento y no modificar de manera más extensa el documento original. Finalmente, en la sección 3.3 se presentan brevemente algunos de los componentes del proyecto completo que mantienen relación con o influyen en el desarrollo de la guía arquitectónica.

3.1 Consideraciones de confidencialidad

Por motivos de confidencialidad no puede revelarse el nombre del cliente para el que se desarrolló este proyecto y, a fin de no dar indicios de su identidad, se han realizado cambios mínimos al texto del trabajo presentado. Los cambios realizados incluyen:

- Eliminación de nombres e instituciones en el texto y redacción para eliminar referencias directas. En el texto se sustituye de manera intercambiable el nombre y abreviaturas del cliente por la organización o la dependencia.
- Se sustituyen los ejemplos originales dados en el lenguaje y procesos del cliente por otros equivalentes en el contexto, pero relacionados con procesos de control escolar, probablemente más familiares para un público general. Tanto en los originales como en los modificados se consideran sobresimplificaciones de los problemas con fines ilustrativos y no deberán de extenderse fuera del contexto en que se presentan.
- El resto de los productos de la consultoría referidos en el texto del trabajo forman parte del proyecto general, pero no del trabajo presentado en este reporte. Estos documentos están fuertemente cargados de normatividad que, aunque pública, no puede ser revelada sin evidenciar la identidad del cliente. Considero que estos documentos no son relevantes para fines de evaluar la calidad del producto.

3.2 Integridad del trabajo original

Con el fin de mantener la integridad del trabajo original, las secciones relevantes que como complemento se requieran para redondear el contenido de este texto serán incluidas en este documento. Esto incluye la información introductoria previamente presentada y las aclaraciones y conclusiones que sobre la experiencia profesional deriven.

Considero que es importante para efectos de la evaluación mantener, en la medida de lo posible y salvo las consideraciones de confidencialidad expresadas en 3.1, la estructura y forma del trabajo original, ya que su propósito es distinto al de este trabajo y su alcance está

dentro de los límites impuestos por el proyecto en el que se desarrolló. El contenido del trabajo original se presenta en el Anexo A.

El trabajo que se presenta en las secciones principales de este documento puede exponer de manera más profunda el contexto en el que se desarrolló y, de la misma manera, exponer las conclusiones o sumarizar la experiencia en la realización y puesta en marcha de la guía.

3.3 Contexto del trabajo presentado

Como ya hemos comentado, la *Guía arquitectónica para el desarrollo de aplicaciones* forma parte de un proyecto más amplio que tuvo como objetivo generar un contexto tecnológico institucional para toda la dependencia. De manera más específica, los objetivos buscados en la implantación de la arquitectura fueron:

- Contar con una meta clara para la estructuración de la infraestructura de sistemas que soportarán la operación de la dependencia.
- Contar con herramientas para comunicar efectivamente a todos los involucrados en la procuración de soluciones de software los lineamientos tecnológicos y de procesos.

3.3.1 Requerimientos

Para dicho proyecto se definieron una serie de requerimientos a satisfacer basados en la situación actual de la dependencia. A continuación presentamos el requerimiento general que motivó la creación de la *Guía arquitectónica para el desarrollo de aplicaciones*:

Dar lineamientos para la construcción de nuevos sistemas. Establecer una guía arquitectónica para el desarrollo de sistemas que le permita, a aquellos interesados en implantar un sistema en la dependencia, conocer el contexto tecnológico, los lineamientos a los que se tiene que sujetar este sistema, y las interfaces requeridas para la integración de este en el ambiente actual y futuro de la dependencia.

Dentro de los mismos requerimientos se delinean las características específicas que influyeron en el desarrollo de la *Guía arquitectónica para el desarrollo de aplicaciones*.

- **Documentación de los componentes principales de la arquitectura objetivo de la dependencia.** Después de la definición de los elementos de la Arquitectura se requiere la documentación adecuada y clara de los mismos, propiciando un mejor entendimiento de su implementación.
- **Propiciar la interoperabilidad entre los sistemas de la dependencia.** Se deberá favorecer la funcionalidad de los nuevos sistemas o herramientas a integrar, de tal manera que puedan interactuar de forma eficaz con los sistemas ya existentes.
- **Adaptación a las plataformas de los sistemas existentes.** La dependencia ha desarrollado sistemas en diversas plataformas; es por ello que los nuevos sistemas

deberán ser adaptables a éstas.

- **Definición de una capa de intercambio de información.** Se requiere definir una capa de intercambio de información entre los sistemas, accediendo y replicando en la dirección que sea requerida, definiendo jerarquías entre las capas, para otorgar el impacto necesario de la interacción.
- **Escalabilidad de los sistemas a través del tiempo.** Debe de propiciar que los sistemas a desarrollar tengan la capacidad de crecer adaptándose a nuevos requisitos conforme cambien las necesidades del Instituto.
- **Portabilidad de los sistemas entre distintos ambientes.** Se debe establecer la capacidad de los sistemas para ser transferidos de un ambiente a otro.
- **Aumentar la confiabilidad de los sistemas desarrollados.** Es imperativo que los nuevos sistemas mantengan un nivel operativo adecuado bajo ciertas circunstancias, considerando la tolerancia a fallas, recuperación, apego a estándares..
- **Flexibilidad en los sistemas de forma eficaz.** Se busca que la capacidad de modificación de los sistemas sea eficaz conforme las necesidades de la organización cambien.
- **Contar con la documentación adecuada.** Se debe establecer la documentación requerida en cada desarrollo, con la finalidad de llevar a cabo el mantenimiento y modificaciones adecuadas de los sistemas.

3.3.2 Productos relacionados

En este contexto se generaron otros productos que, aun cuando no se presentan como parte del trabajo, tienen una relación con la *Guía arquitectónica para el desarrollo de aplicaciones* y se complementan mutuamente. En esta sección describiremos los componentes que mantienen una relación con el trabajo presentado y detallaremos dicha relación.

- **Proceso de implantación de soluciones de software.** Define las etapas que se deberán llevar a cabo al momento de elegir una solución informática para la dependencia. Establece puntos de control que propicien la evaluación de cada etapa, supervisando la construcción, aumentando la calidad y disminuyendo posibles errores. Este producto tiene una estrecha relación con esta guía; en la etapa de diseño de solución o evaluación de proveedores y herramientas da un marco objetivo contra el cual comparar las diversas soluciones ofrecidas. Aquellas que no se apeguen a los lineamientos establecidos en la guía arquitectónica deberán descartarse o considerar las modificaciones necesarias dentro de la propuesta. En este sentido la guía es un elemento de apoyo al proceso.
- **Guía para la documentación de software.** Determina la documentación necesaria que otorgue un mayor entendimiento de las soluciones de software implantadas, plasmando su arquitectura y funciones. Se define la documentación mínima requerida

así como los documentos arquitectónicos referentes a las vistas de los sistemas. Finalmente, se establecen los criterios de evaluación que serán aplicados a los documentos entregados, con el objetivo de verificar el cumplimiento de lo que la guía para la documentación de software establece.

En la guía de documentación se establece la manera en la que la arquitectura debe de plasmarse para futura referencia. La guía arquitectónica establece los lineamientos para el contenido de ésta. Es aceptable referirse a la guía arquitectónica para los aspectos generales del sistema, y en ese caso los documentos de arquitectura deberán de concentrarse en los componentes particulares de cada módulo. Por lo tanto, la relación entre estas dos guías es de mutua dependencia; la guía de documentación indica la forma y la guía arquitectónica el fondo de la arquitectura de los módulos.

- **Autorización, autenticación y auditoría.** Existe una línea de acción dirigida a proveer herramientas para el desempeño de estas funciones. En dicha línea de acción se planteó la puesta en marcha de OpenSSO como proveedor de autenticación y autorización. Esta herramienta se integra entonces a la arquitectura institucional como uno de los módulos de uso común planteados en el panorama arquitectónico. La relación entre estos dos elementos radica en que la solución de seguridad se apega a los lineamientos dados por esta arquitectura. La separación de los módulos, como la plantea la guía arquitectónica, es posible dadas las características de esta solución, en particular en la capa de presentación. Por lo tanto, podemos decir que existe una dependencia mutua entre estos dos productos.

- **Guía para el manejo de la información.** Clasifica la información que se maneja dentro de la dependencia, dando lineamientos para el uso y custodia, así como estableciendo una serie de políticas que permitan satisfacer las necesidades de seguridad e integridad. En esta guía se definen, entre otros, la necesidad de contar con un titular o área responsable de cada conjunto de información en la institución, y las clasificaciones de los intercambios permitidos de acuerdo con el tipo de información que se tiene. Estos dos elementos inciden en la definición de los módulos de acuerdo con lo planteado en la guía arquitectónica. Por ello es correcto afirmar que la guía para el manejo de la información soporta a la guía arquitectónica.

Existen algunos otros productos desarrollados bajo el mismo proyecto, pero se considera que no presentan relaciones relevantes con esta guía arquitectónica.

4 Experiencia adquirida

Como en todo proyecto existen resultados que van más allá de los entregados al cliente; los beneficios para el individuo que ejecuta el proyecto incluyen múltiples enseñanzas sobre puntos generales y específicos que contribuyen a lo que en general llamamos experiencia. En esta sección enumeramos los puntos más relevantes de la experiencia adquirida en el proyecto.

En primer lugar debemos mencionar el conocimiento profundo que se adquirió sobre las normas y reglamentos que regulan, desde el punto de vista tecnológico, a los diversos componentes del Gobierno Federal. Se destacan en este rubro la Ley Federal de Transparencia y Acceso a la Información, y el Marco de Referencia de Arquitectura Gubernamental publicado por la Secretaría de la Función Pública.

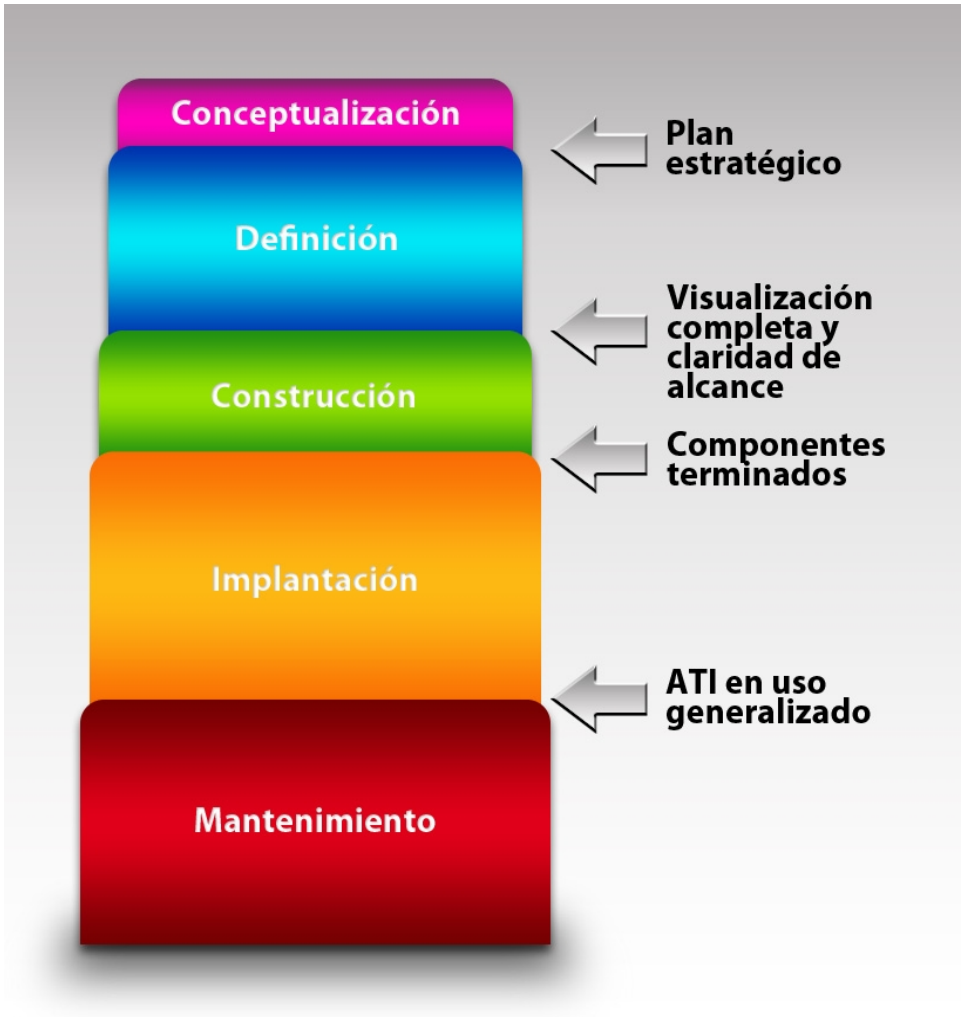
Otra importante fuente de experiencia fue el estudio profundo de las normas y mejores prácticas internacionales para sector público y privado que existen actualmente. Aun cuando éstas no apliquen de manera normativa al enfoque empleado en ellas es enriquecedor para efectos del proyecto desarrollado. En este rubro destacan el Federal Enterprise Architecture Framework y TOGAF[2].

Finalmente, y de manera general, se pudo apreciar y argumentar extensivamente la importancia de contar con lineamientos arquitectónicos dentro del proceso de desarrollo de software. Entre los principales beneficios de contar con lineamientos encontramos que se facilita tanto el mantenimiento como la integración de las soluciones encontradas en la empresa. Adicionalmente, las decisiones técnicas dentro de los proyectos son más ágiles ya que desde el inicio se cuenta con un conjunto marco de requerimientos no funcionales.

2 The Open Group Architecture Framework. Es un método detallado y herramientas de soporte para desarrollar arquitecturas empresariales. Puede ser usado de manera libre por cualquier organización que desee construir una arquitectura empresarial para uso interno.

5 Estado actual

El proyecto de la Arquitectura Tecnológica Institucional (ATI), se definió en cinco etapas principales, marcadas por hitos en el proceso de implantar la arquitectura institucional.



Actualmente se concluyó la etapa de construcción, lo cual implica que tanto la guía arquitectónica para el desarrollo de aplicaciones como la ATI, en términos generales se encuentran entregadas y aceptadas por la dependencia. Continuando con el plan, todos los productos se encuentran en etapas tempranas de su implantación y hasta ahora el resultado ha sido positivo. El objetivo de la etapa de implantación es establecer su uso generalizado dentro de la organización, de manera que se perciban no sólo los beneficios individuales de cada línea de acción, sino aquellos que se derivan de la colaboración o trabajo en conjunto de dos o más líneas de acción.

6 Conclusiones

Para cerrar este trabajo de presentación incluiremos en esta sección las principales conclusiones a las que se llegó como resultado de la definición de la Guía arquitectónica para el desarrollo de aplicaciones.

- Dar una arquitectura normativa no puede darse en aislamiento, sino que se deben considerar siempre las necesidades y prioridades de la organización que la albergará.
- Se deben prever mecanismos para la adaptación de la arquitectura a las nuevas tendencias y tecnologías sin descuidar los objetivos
- Ser demasiado específico en este tipo de documentos puede ser contraproducente ya que se fomenta un estancamiento tecnológico de la organización.

Anexo A: Guía arquitectónica para el desarrollo de aplicaciones

Anexo A.I: Introducción

Una importante agencia gubernamental a través de su Dirección de Sistemas inició el proyecto para la definición de la Arquitectura Tecnológica Institucional (ATI), con el objetivo de lograr una integración de los sistemas existentes y desarrollos futuros, así como el planteamiento de lineamientos conceptuales y específicos orientados a una arquitectura de procesos y servicios.

El presente documento contiene la guía arquitectónica para el desarrollo de sistemas, que permite a los interesados en desarrollar soluciones de software para la dependencia conocer el ambiente tecnológico, los lineamientos a los que se tiene que sujetar, y las interfaces requeridas para la integración en el ambiente actual y futuro de la organización.

Para lograr este objetivo, esta guía describe, antes que nada, la visión arquitectónica de la organización (sección A.II) y procede a indicar los elementos que cada una de las aplicaciones debe incorporar para poder contribuir a esta visión.

En la sección A.III se describe la estructura arquitectónica objetivo y se definen los conceptos necesarios para el establecimiento de los lineamientos que componen la ATI. Estos lineamientos se presentan en la sección A.IV y se exponen en un nivel creciente de complejidad.

Este documento es de carácter técnico y supone del lector cierta experiencia en el diseño y desarrollo de sistemas, así como conocimientos generales de la plataforma preferida de desarrollo[3]. Los ejemplos han sido cambiados de su forma original por motivos de confidencialidad y están dados en términos de funciones escolares, a fin de que pueda suponerse una cierta familiaridad con el entorno y así facilitar su entendimiento.

3 Al momento de escritura de este documento la plataforma preferida es *Java Enterprise Edition*

Anexo A.II: Visión arquitectónica

La organización, a raíz de lo experimentado en los últimos años e influida por la transformación que actualmente lleva a cabo, ha identificado como necesidades clave para su infraestructura de sistemas la integración, flexibilidad y mantenibilidad de los sistemas de información que soportan su operación y, por lo tanto, el cumplimiento de la misión y objetivos de la dependencia.

Debido a lo anterior, se identifica la necesidad de contar con una visión clara del panorama tecnológico que la organización desea, así como lineamientos que permitan hacerlo una realidad.

En esta sección estableceremos no sólo dicho panorama, el cual se presenta en la sección A.II.II, sino las metas que se buscan alcanzar (sección A.II.I).

Anexo A.II.I: Metas

La integración buscada de los sistemas debe entenderse como un concepto amplio que incluye no sólo la necesidad de los sistemas que al interior de la dependencia requieren intercambiar información para la realización de sus funciones, sino también la cooperación de las distintas áreas, representadas por los sistemas que las soportan, para completar los procesos a nivel organizacional. Más aún, las necesidades de integración deben de considerar la creciente cooperación que existe entre distintos agentes que proporcionan información o consumen los servicios que la organización brinda.

Derivado en parte de la transformación que experimenta la organización, y en parte de la experiencia con las necesidades cambiantes del mismo, la flexibilidad en las aplicaciones pretende facilitar nuevas maneras de realizar las tareas para las que fueron pensadas y combinar de manera ágil estas tareas para crear nuevos procesos u optimizar los ya existentes. Debido a lo anterior, elementos fundamentales de las aplicaciones serán la modularidad y una clara definición de los servicios que provee cada una de ellas.

El complemento, que permitirá a la organización reaccionar con la debida celeridad a los nuevos requerimientos que constantemente surgen en su labor, es una adecuada mantenibilidad de los componentes que conforman su plataforma tecnológica. La mantenibilidad deberá entenderse como la capacidad de un sistema o módulo del mismo para ser diagnosticado, en caso necesario reparado o bien transformado de manera ágil y correcta. Es decir, que dadas ciertas condiciones se posibilite la modificación ágil y segura de los sistemas.

Anexo A.II.II: Panorama objetivo

Para lograr los objetivos planteados, la dependencia contará con una plataforma tecnológica de sistemas compuesta de *módulos funcionales*[4] que proporcionan sustento a las tareas que cada una de las áreas debe realizar.

Cada uno de estos módulos funcionales cuenta con un objetivo definido y expone sus servicios de una manera uniforme y segura. Esta plataforma permite no sólo la integración de herramientas adicionales que facilitan o extienden la funcionalidad disponible, sino que habilita la creación de nuevas aplicaciones mediante la combinación de las funciones provistas por cada servicio. Dada esta relación, en ciertos contextos podemos hablar de manera intercambiable de los servicios y los módulos funcionales que los implementan.

Cada uno de estos servicios funciona de manera coordinada con ámbitos de acción únicos. En la siguiente gráfica se ejemplifican algunos de los servicios considerados, ubicados en la categoría a la que pertenecen.

4 Una definición más amplia de este concepto se presenta en la sección A.III.I.

Como se puede ver en la gráfica, la visión de la plataforma tecnológica es una colección de servicios que cooperan entre ellos para impulsar los procesos de negocio de la organización.

Los módulos de negocio se encuentran soportados de manera uniforme por servicios institucionales, que permiten separar las responsabilidades generales y las funciones institucionales específicas; de esta manera, se proporciona un servicio unificado a todos los que lo requieren.

Existen tres categorías principales de servicios:

- **Servicios institucionales**. Esta categoría se refiere a todas aquellos módulos que, aun cuando no realizan una función de negocio, son utilizados para soportar el funcionamiento de otros servicios; es decir, implementan una función común a varios servicios y permiten aislar dicha función y realizarla de manera uniforme en toda la dependencia. Un efecto colateral de esto es que los desarrollos individuales, al no tener que contemplar este punto, pueden ser más ágiles. Entre los ejemplos de este tipo de servicios, que son variados están incluidos:
  - Elementos de uso generalizado, como seguridad y auditoría.
  - Integración de tecnologías, como manejo de documentos.
  - Servicios particulares a las necesidades de la organización, como catálogos comunes o un sistema de notificaciones.

  Es importante mencionar que el hecho que estos servicios sean adaptados, o incluso particulares a las reglas de negocio de la dependencia, no los convierte en parte de ninguna de las dos siguientes categorías, debido a que no forman parte de la misión de la organización, sino un medio para lograrla.
- **Servicios esenciales.** Este conjunto de servicios forman una categoría, no sólo por ser los que sustentan las actividades principales de la dependencia, sino porque a través de ellos se genera la información consumida por otros servicios; o bien proporcionan los servicios que otros módulos extienden o consumen para realizar sus funciones. Los ejemplos de este tipo de servicios incluyen: inscripciones, control escolar y trámites.
- **Servicios adicionales.** Debe construirse múltiples módulos adicionales para lograr cubrir la totalidad de las actividades que realiza la dependencia y los servicios que ofrece. Estos servicios normalmente estarán relacionados a uno de los módulos esenciales, pero no necesariamente formarán parte de los mismos ni tendrán el mismo grado de importancia.

El panorama tecnológico de la organización se compone, entonces, de una colección de servicios que pertenecen a una de estas categorías. En las próximas secciones detallaremos la arquitectura de cada uno de estos servicios, indicando cómo contribuye a la construcción de la plataforma institucional.

Anexo A.II.III: Alcance

La arquitectura descrita en este documento es aplicable para todos los desarrollos que en un futuro deseen integrarse a la plataforma tecnológica institucional de la organización. Aplica indistintamente para los desarrollos realizados por el personal de la dependencia, así como a aquellos adquiridos a proveedores externos.

Se toma una decisión consciente de orientar la arquitectura institucional a un paradigma orientado a objetos y, más aún, se busca contar con una arquitectura orientada a servicios. Pueden admitirse otro tipo de paradigmas dentro de la plataforma siempre y cuando se ajusten a los principios delineados en esta arquitectura.

La plataforma recomendada para implementación será Java en su edición empresarial (Java EE); sin embargo, se preve el uso de la plataforma .Net por considerarla equivalente en las funciones requeridas para la correcta implementación de esta arquitectura.

Otros elementos, sobre todo fuera del nivel aplicativo, se dejan intencionalmente sin definir, considerando que la evolución de la tecnología permitirá incorporar nuevos elementos en este sentido, siempre y cuando presenten ventajas, de acuerdo con la naturaleza y requerimientos específicos del módulo a desarrollar. Dicho lo anterior, en la *Guía para el desarrollo de aplicaciones*[5] se presentan recomendaciones tecnológicas específicas para la implementación de los componentes descritos en esta arquitectura.

De manera inicial, el foco de esta arquitectura consiste en los sistemas relacionados con la operación sustantiva de la dependencia. Otros aspectos de su funcionamiento, notablemente las áreas administrativas, pueden en principio ser soportadas por sistemas con características distintas, entendiendo que las necesidades que motivan esta arquitectura pueden no aplicar para dichas áreas.

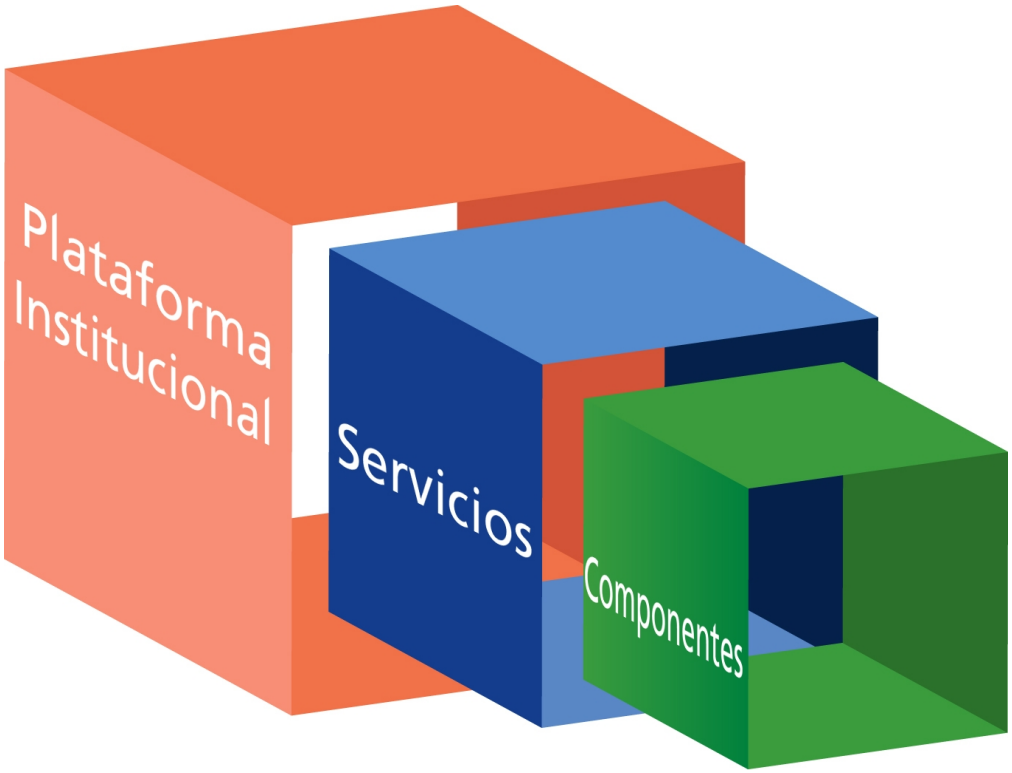
5 La guía para el desarrollo de aplicaciones no forma parte de este documento por no ser central a la guía arquitectónica para el desarrollo de aplicaciones.

Anexo A.III: Vista arquitectónica estructural

Desde el punto de vista estructural hemos hablado de servicios que integran la plataforma tecnológica, cada uno de los cuales está implementado por un módulo funcional, como lo definiremos en la sección A.III.I.

Estos módulos son el elemento marco en el que se encapsula un proceso de negocio. Aun cuando a menudo un servicio corresponde con un módulo, existen requerimientos de negocio complejos que para facilitar su concepción, construcción, operación y mantenimiento son soportados por un módulo, que a su vez coordina otros módulos que operan con cierto grado de independencia entre ellos. Esto es fácilmente entendible si pensamos que los servicios soportan procesos y que éstos pueden componerse de subprocesos.

Los módulos a su vez se encuentran formados por componentes, los cuales se describen en la sección A.III.V. Para facilitar la definición de componentes y describir los lineamientos de arquitectura de servicios contenidos en este documento, estableceremos un marco de referencia, dado en capas y niveles, como se describe en A.III.IV.



El diagrama anterior muestra la relación jerárquica entre los conceptos que se exponen en este documento.

Anexo A.III.I: Módulos funcionales

Definimos un módulo funcional como una agrupación de *funciones de negocio*. Las funciones del módulo están relacionadas en términos de negocio; a menudo tienen en común el modelo del problema sobre el que actúan.

A manera de ejemplo podemos concebir un módulo funcional cuyo objetivo sea el mantenimiento de listas de grupos. Este módulo funcional forma parte del servicio de inscripciones y agrupa a las funciones de alta, baja y cambios de la lista de grupos. Pueden ser necesarias funciones adicionales para estar en posibilidad de brindar el servicio de inscripción, pero el mantenimiento de la lista es un módulo funcional por sí mismo.

Los módulos funcionales cubren procesos completos de negocio, sin importar cuántos actores intervengan en el mismo. Es decir, un módulo sirve a uno o más perfiles de usuario para realizar un proceso de negocio. El conjunto de los casos de uso que permiten completar el proceso se transforman en las funciones del módulo. Debido a la definición anterior, un módulo es un concepto existente sólo en el nivel de aplicación[6].

Diferentes actores pueden estar involucrados en el escenario anteriormente descrito: por ejemplo, la carga de un elemento puede ser realizada por un perfil de usuario pero la baja requerirá de otro perfil. El proceso completo al que apoya este módulo funcional es el de planeación y ejecución del ciclo escolar.

Los módulos pueden, sin embargo, tener dependencias de otros módulos para realizar sus funciones, ya sea de otras funciones de negocio, por ejemplo para detonar otros procesos como en el caso de que se requiriera una notificación ante un movimiento de baja; o bien por dependencias del modelo de negocio en otro modelo de negocio. En nuestro ejemplo, el mantenimiento al tratar de grupos con una asignatura dependería del modelo común de asignaturas contenido en los catálogos institucionales.

Es importante que las dependencias no sean bidireccionales, ya que esto implica una relación más fuerte que sugiere que las funciones de los módulos deben ser redistribuidas.

Una aplicación o servicio sencillo puede estar implementado con un solo módulo mientras que aplicaciones más complejas pueden llegar a tener decenas de módulos soportando los diversos procesos que el servicio requiere.

Un módulo debe poder ser ejecutado y probado de manera separada de otros módulos de los cuales no depende. La integración de varios módulos en un ambiente de tal forma que parezca una sola aplicación, es un aspecto técnico localizado en la interfaz de usuario[7] y que está relacionado con uno de los servicios institucionales, el de seguridad. Una de las maneras de resolver esto es mediante *Single Sign-On*[8].

---

6 El significado preciso del nivel aplicativo se encuentra definido en la sección A.III.IV. Basta, por el momento, decir que es el nivel donde encontramos el software desarrollado específicamente para nuestro problema.

7 Uno de los componentes que normalmente forman parte de los módulos.

8 Single sign-on (SSO) es un procedimiento de autenticación que habilita al usuario para acceder

Anexo A.III.II: Clasificación de componentes

Como ya mencionamos, los módulos están formados a su vez por componentes que colaboran para realizar las funciones necesarias. Antes de ahondar en las características y categorías de los componentes, sentaremos las bases de un marco de trabajo para la clasificación de los diferentes componentes que integran un servicio.

Para representar un módulo funcional desde el punto de vista de los componentes que lo conforman, nos auxiliamos de los conceptos de capas y niveles. Éstos nos dan una visión en dos dimensiones del sistema que describimos. Las capas nos hablan de las funciones desempeñadas por el sistema dando una separación basada en el papel que ejecutan dentro de la arquitectura, mientras que los niveles separan dichas funciones de acuerdo con el nivel de abstracción en el que se encuentran.

En el nivel aplicativo, el concepto de capa es, al igual que un módulo, una agrupación de componentes; pero a diferencia de éstos, en las capas la relación está dada por las funciones que implementan. Es decir, la principal distinción radica en que los módulos representan una agrupación orientada a simplificar los requerimientos funcionales, mientras que las capas son una división de componentes orientada a simplificar la implementación de éstos. Las capas entonces agrupan componentes que tienen un papel similar y atraviesan todos los niveles definidos, incluyendo en cada nivel los elementos relevantes de éste para el funcionamiento de los componentes en el nivel aplicativo.

Anexo A.III.III: Niveles

Como ya mencionamos, los niveles están orientados a establecer una división de los elementos involucrados basada en el nivel de abstracción. Como tal, en cada nivel podemos identificar no sólo componentes de software, sino también protocolos e incluso elementos físicos.

Los niveles que se consideran en este documento, y a los que se deberán referir los documentos arquitectónicos de los servicios de la organización, son:

- **Aplicación**. Este nivel puede considerarse el principal nivel en la descripción del servicio, ya que es el que contiene los componentes desarrollados y es a menudo el que implementa la funcionalidad particular del servicio. Los lineamientos descritos en esta guía principalmente se enfocan en la arquitectura de este nivel.
- **Plataforma.** Este nivel agrupa las especificaciones, estándares o bibliotecas sobre las

---

a varios sistemas con un solo elemento de identificación. En su forma web los accesos son interceptados con la ayuda de un servidor proxy o de un componente instalado en el servidor web destino. Los usuarios no autenticados que tratan de acceder son redirigidos a un servidor de autenticación y regresan sólo después de haber logrado un acceso exitoso. Normalmente se utilizan cookies y sesiones de servidor para reconocer aquellos usuarios que acceden y su estado de autenticación.

que se sostienen los componentes del servicio. En este nivel incluimos las dependencias directas de la aplicación, ya sea a nivel de especificación o incluso de otros servicios que la soporten. La importancia de este nivel se debe a que las decisiones sobre los componentes que se incluyan serán la principal fuente de portabilidad del sistema, ya que establecen los requerimientos para la infraestructura de software que lo soporta. Ejemplos de componentes que se identifican en este nivel pueden ser JSF para presentación, bajo éste Servlets, o bien elementos concretos como Spring Framework.

- **Infraestructura de software**. Este nivel representa las implementaciones de la plataforma que se utilizarán en el sistema. El objetivo de este nivel es definir los elementos que proporcionarán los servicios especificados por la plataforma. En este nivel se pueden considerar las diversas opciones para la puesta en producción o desarrollo del sistema. Deben considerarse o incluirse en este nivel los sistemas operativos, servidores de aplicaciones, manejadores de bases de datos y otros elementos de software impulsando los elementos de la capa adecuada.
- **Hardware y telecomunicaciones**. El nivel del hardware debe considerar las diversas soluciones físicas sobre las que se ejecuta el sistema, incluyendo las funciones de procesamiento, almacenamiento y telecomunicaciones. Algunos elementos de software pueden incluirse en este nivel siempre y cuando aporten a las funciones que radican en él. Por ejemplo, si dos servidores se usan para proporcionar tolerancia a fallas, se puede incluir un elemento de software que se encargue de coordinarlos. Lo mismo aplica a protocolos de red utilizados en el despliegue del servicio.

Esta división por niveles facilita el mantenimiento, extensión e integración de los sistemas, ya que provee una referencia para ubicar jerárquicamente a los componentes utilizados en la aplicación.

Anexo A.III.IV: Capas en el nivel de aplicación

Como mencionamos anteriormente, el elemento central de este documento es el nivel aplicativo. En esta sección definiremos los componentes comúnmente requeridos para la entrega de un servicio y los ubicaremos dentro de las capas adecuadas de acuerdo con el modelo orientado a servicios de la ATI.

Como ya hemos referido, las capas tienen la función de dividir el trabajo de los servicios en componentes con cierto grado de independencia, de acuerdo con el papel que desempeñan. En consecuencia, definimos las siguientes capas, las cuales se presentan ordenadas de acuerdo a sus interacciones :

- **Datos.** La capa de datos agrupa la estructura y datos persistentes donde radica la información del servicio. En su forma más básica esta capa comprende las tablas y su contenido, aunque en el nivel aplicativo puede incluir elementos adicionales como índices para desempeño o procedimientos almacenados, necesarios para la operación del servicio; aun cuando no es recomendado por la ATI trasladar la lógica de negocio a esta capa, puede ser necesario implementar algunas funciones de esta manera,

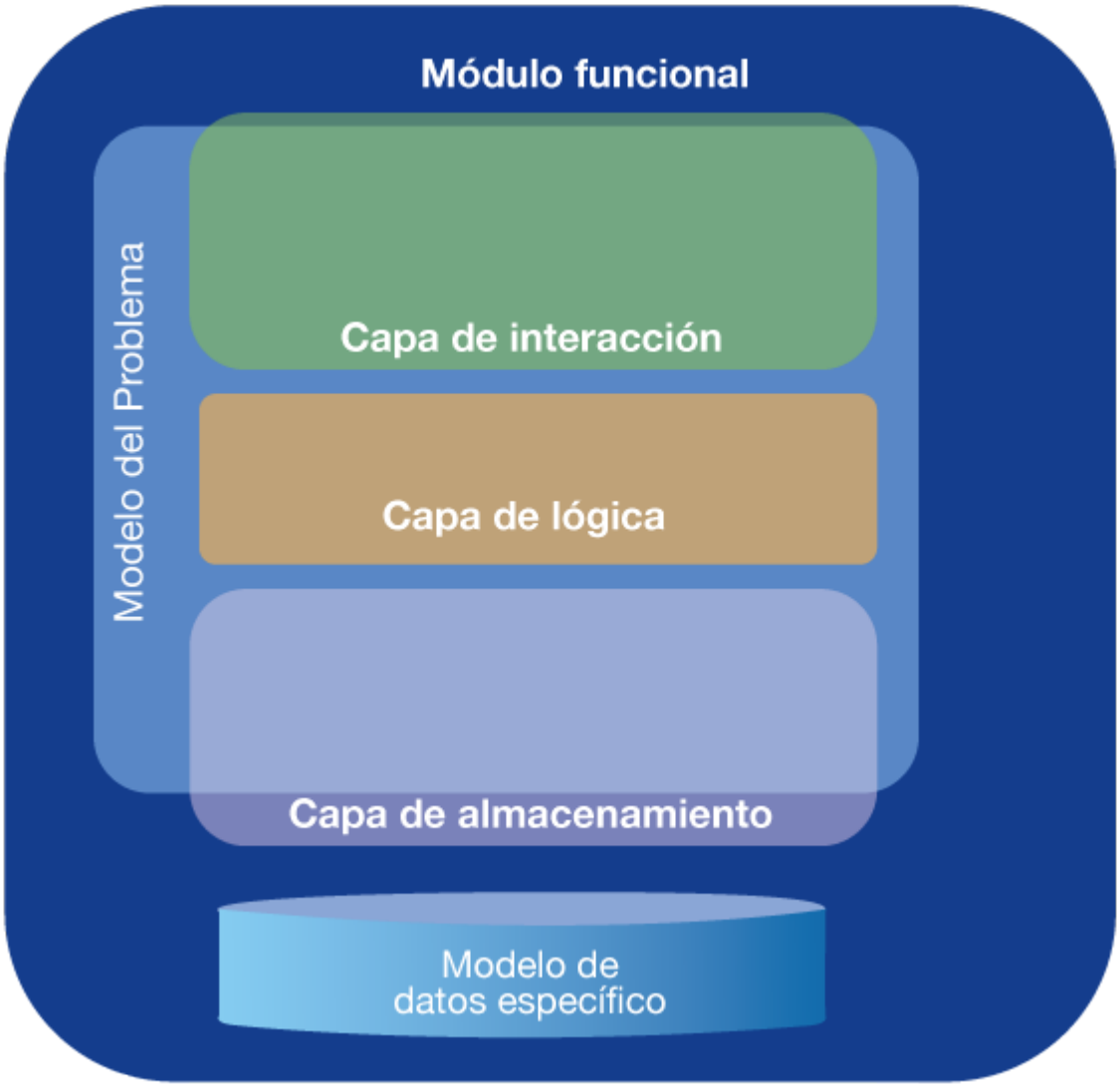
principalmente por criterios de eficiencia.

- **Almacenamiento.** La capa de almacenamiento tiene una función de traducción y acceso gestionado a la información contenida en la capa de datos. Esta capa proporciona servicios de persistencia y localización de objetos a la capa de lógica; debe estar orientada a proveer los servicios de manera flexible, robusta y eficiente, poniendo especial cuidado en preservar la independencia de la lógica de negocio.
- **Lógica**. Esta es la parte fundamental de cualquier servicio, ya que realiza las funciones particulares que apoyan los procesos de negocio y provee los servicios que la capa de interacción requiere. A su vez, es normalmente cliente de la capa de persistencia. Finalmente, la capa de lógica puede ser cliente de otros servicios para colaborar con ellos y así entregar sus funciones[9]. Es importante que aun cuando la lógica de negocio sirve a la capa de interacción, su diseño no esté subordinado a ésta y cuente con una identidad y propósito propios para asegurar su duración y supervivencia futuras.
- **Interacción**. La capa de interacción, como su nombre lo indica, es la que permite que la lógica y datos encapsulados en el servicio se relacionen con el ambiente. Los componentes que más frecuentemente encontramos en esta capa son las interfaces de usuario, que permite al sistema interactuar con humanos a los que sirve para realizar sus tareas y completar así el proceso de negocio del módulo. Sin embargo, existen otros tipos de componentes que deben considerarse en esta capa y son los que permiten al módulo comunicarse con otros sistemas. Estas interfaces pueden ser tanto para exponer los servicios que proporciona la capa de lógica como para consumir servicios que otros módulos expongan.

Estas son las capas que propiamente se distinguen en la arquitectura; sin embargo, existe un tipo de componente adicional que, aun cuando no constituye una capa, es ciertamente un elemento de clasificación de componentes. Nos referimos al **modelo del problema**.

El modelo tiene la finalidad de representar el dominio del problema objeto del proceso que el módulo automatiza; por esta razón, a menudo el o los componentes del modelo están presentes en las capas de almacenamiento, lógica e interacción; incluso en las interfaces que las comunican.

9 La manera en la que se realiza esta interacción se detalla en la sección A.IV.III



Anexo A.III.V: Componentes aplicativos

Entendido el marco en el que clasificaremos los componentes, ahondemos en la definición del mismo. Entendemos un componente como una pieza de software que cumple un propósito específico dentro del funcionamiento del módulo. Para ser considerados como componentes, las funciones que implementan deben ser relevantes desde el punto de vista del objetivo del módulo. En módulos sencillos a menudo tendremos uno o dos componentes por cada capa, mientras que en módulos complejos podemos incrementar este número; sin embargo, debemos mantener el número de componentes controlado, ya que una capa

demasiado compleja a menudo es indicativa de que el módulo agrupa demasiadas funciones; y es conveniente separarlo en otros módulos correspondientes a subprocesos del proceso objetivo.

Un componente, al agrupar diversas funciones relacionadas, puede alcanzar cierto grado de complejidad, por lo que a menudo el patrón de fachada[10] puede resultar útil para representar las funciones del componente y exponerlas a otros componentes.

Los componentes deben corresponder con paquetes en el caso de Java y con espacios de nombres en .NET. En otros lenguajes o plataformas, deben usarse elementos propios de los mismos para separar cada uno de los componentes. Es necesario considerar una estructura jerárquica que incluya la capa y el nombre del componente.

Para los desarrollos realizados particularmente para la dependencia, ya sean por personal interno o externo, se debe usar la siguiente convención de nombrado

```
mx.gob.dependencia.<modulo>.<capa>[.componente][.divisiones-adicionales]
```

Los elementos entre <> son obligatorios y entre [ ] son opcionales; en este caso el nombre del componente es opcional, ya que si es el único que se encuentra en esa capa puede omitirse, aunque lo anterior es una decisión que no debe tomarse a la ligera ya que limita la futura extensibilidad del módulo. Dentro de un componente puede haber razones para contar con subdivisiones, por ejemplo para separar las fachadas de la implementación o para reflejar el diseño interno del componente[11].

En el caso de los componentes descritos en el Anexo A.IV.I.II, Modelo del problema, los cuales normalmente no pertenecen a una capa, podemos utilizar en lugar del nombre de la capa `modelo`.

A menudo es conveniente proveer mecanismos de construcción independientes para cada componente, aunque esto no es requerido por la ATI.

---

10 Para información del patrón de fachada consultar *Design Patterns: Elements of Reusable Object-Oriented Software* (ISBN 0-201-63361-2) , Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides.

11 `mx.gob.dependencia` se presenta como la estructura típica; en todos los casos se deberá sustituir por el dominio principal de la organización en orden inverso, de acuerdo a las convenciones de codificación aceptadas por la comunidad; por ejemplo para la U.N.A.M, debería usarse `mx.unam` (http://java.sun.com/docs/codeconv/html/CodeConventions.doc8.html#367).

Anexo A.IV: Lineamientos arquitectónicos

En esta sección estableceremos los lineamientos arquitectónicos que deben respetar los servicios que se incorporen a la plataforma institucional. Los lineamientos se presentan ordenados desde los casos más sencillos hasta los más complejos. Se hace especial énfasis en las características que deberán presentar los componentes y la manera correcta en la que se pueden combinar para generar los servicios.

Anexo A.IV.I: Módulos simples

Como hemos mencionado, un módulo debe tener como objetivo la automatización de un proceso de negocio. Para lograr esto, el proceso debe estar correctamente identificado con un inicio, un fin y los resultados esperados. En la mayoría de los casos el proceso involucra cierta participación de uno o más usuarios, posiblemente con diferentes perfiles.

Observemos en la siguiente gráfica los componentes básicos y su ubicación en las capas definidas en la sección A.III.IV.

En la gráfica anterior observamos los cinco componentes del módulo:

- Modelo de datos específico, descrito en el Anexo A.IV.I.I.
- Modelo del problema, descrito en el Anexo A.IV.I.II.
- Componente de persistencia, descrito en el Anexo A.IV.I.III.
- Componente de lógica y sus interfaces de servicio, descrito en el Anexo A.IV.I.IV.
- Interfaces de usuario, descritas en el Anexo A.IV.I.IV.

Para efectos de la descripción de este tipo de módulo tomaremos una simplificación del proceso de inscripción. El proceso, de manera simplificada, inicia cuando una persona se presenta en una ventanilla y establece su intención de inscribirse en diversos cursos. El encargado procede a recabar su información y, de acuerdo con las reglas establecidas, solicita los requisitos para su inscripción. En caso de cubrir todos los requisitos, se procede al registro en los cursos y se autoriza su inscripción. Existen diversas razones por las cuales puede no completarse el flujo normal, y la mayoría de éstas implican que la persona en cuestión inicie un proceso de autorización. Pero estos flujos por el momento no serán objeto de este ejemplo. Recordemos durante la lectura que ésta es una simplificación del proceso con fines de ejemplificar la arquitectura. El funcionamiento actual o futuro de un módulo de inscripción puede diferir de lo aquí descrito.

Anexo A.IV.I.I: Modelo de datos específico

El modelo de datos específico representa aquellos datos que el módulo funcional va a administrar. Para la definición de este modelo se deberán considerar específicamente las recomendaciones de manejo de información contenidas en el documento: *Guía de manejo de información*[12].

El enfoque de servicios y la definición de los procesos que el módulo atenderá deberán tener como consecuencia que un conjunto de datos típicamente sea administrado por un solo módulo funcional. Es este el conjunto de datos que representamos con el modelo de datos específico.

El modelo de datos deberá de evitar al máximo la duplicidad de información con otros módulos y/o servicios. Para el caso de las entidades de uso común existe el proyecto de los catálogos institucionales. Si la lógica del módulo requiere de las funciones modeladas por otro servicio, será necesario que se integre mediante alguno de los mecanismos descritos más adelante en las secciones A.IV.II o A.IV.III.

Por ejemplo, el componente del modelo de datos de este módulo representará la información de los estudiantes, los documentos de identificación que presentan, sus inscripciones anteriores en cursos y, posiblemente, los medios por los que se inscribieron. Sin

12 Desarrollado como parte de la ATI, este documento establece los criterios para la definición de elementos de información y el tratamiento que debe dársele; notablemente define la necesidad de asignar un responsable de la información.

embargo, no representará otros elementos comunes, como asignaturas, profesores, tipos de documentos y similares, las cuales se obtendrán de los catálogos institucionales en una integración como la descrita en la sección A.IV.II. Tampoco incluirá la información de los requisitos por curso, ya que éstos se obtendrán mediante el acceso a otro de los servicios de la plataforma, integrado mediante los mecanismos descritos en la sección A.IV.III.

Visualicemos este componente en las tablas: Alumno, Curso_Inscrito y Requisito[13]. Adicionalmente, en este componente podemos identificar un índice sobre el número de cuenta del alumno que nos permita localizarlo rápidamente.

Esta es la forma típica del modelo de datos; sin embargo, es posible que haya mecanismos alternativos de almacenamiento de información: bases de datos de mallas de objetos, archivos planos, bases de datos de información geográfica o biométrica, entre otros. El componente de esta capa será el diseño que sobre ésta se haga para representar el sujeto de acción de este módulo.

Anexo A.IV.I.II: Modelo del problema

Como mencionamos, el modelo deberá de representar adecuadamente el dominio del problema que el proceso que implementa el módulo tiene como objetivo automatizar. Este modelado como componente es posible que se convierta en una dependencia del resto de las capas, ya que éste será el lenguaje en el que a menudo se expresarán los parámetros de las peticiones entre ellas. En ciertos casos sencillos, el modelo del problema puede verse como una representación en el lenguaje de programación de los datos contenidos en el modelo de datos y a menudo se hace un símil entre ambos. Sin embargo, existen diferencias entre ellos, ya que las relaciones que existen en uno y otro tienen naturalezas distintas. Aun cuando pueden mantener equivalencias entre estas relaciones, existe una semántica distinta en un caso y otro.

Lo anterior implica que este componente debe ser diseñado independientemente y, sobre todo, con consideraciones distintas a los elementos de la capa de datos. De manera destacada mencionamos que mientras que en la capa de datos las consideraciones de unicidad de información, eficiencia y normalización son relevantes, en el modelo del problema éstas deben ser sustituidas primordialmente por la facilidad de uso y la correcta semántica de las relaciones que se presenten entre los objetos.

Las clases de este componente a menudo tienen diseños basados en las convenciones de nombrado para propiedades descritas originalmente en la especificación de JavaBeans (get y set)[14] .

A menudo estas clases tienden a utilizar el patrón de diseño de Data Transfer Object

13 Estas tablas son ejemplos y no reflejan nombres reales presentes o futuros de la infraestructura de ninguna organización.

14 Consultar la sección 8.3 *design patterns for properties* de la especificación de Java beans. http://java.sun.com/javase/technologies/desktop/javabeans/docs/spec.html .

(DTO)[15] y se piensan como simples contenedores de información modificable; pero pueden incorporar funciones adicionales para agregar valores calculados o localización de objetos de uso común.

Un ejemplo de esto puede apreciarse si en la capa de datos tenemos una tabla con los requisitos proporcionados por el alumno: podemos, en el modelo del problema, agregar una función que localice específicamente la tira de materias del semestre anterior. Esto será una conveniencia para los que usen el modelo del problema sin afectar la estructura de los datos.

En los paradigmas orientados a objetos se definen dos principales tipos de relaciones entre clases: las relaciones “tiene un” y “es un”, que representan respectivamente la composición y la generalización. Estas relaciones pueden manifestarse de diversas maneras dependiendo del diseño específico de las clases. Consideremos por ejemplo una relación de herencia: un alumno es una persona. Esta relación puede estar representada en un modelo de objetos como:

```
public class Persona {
      ...
}
```

y

```
public class Alumno extends Persona{

      ...
}
```

Otra manera de hacerlo es por medio de una interfaz que represente a la persona y una implementación de la misma que defina a un alumno.

```
interface Persona {

      ...
}
public class Alumno implements Persona {

      ...
}
```

La decisión de cómo implementarla depende realmente de consideraciones sobre el uso que se le dará a dicho modelo. Por lo tanto, no existe un único diseño correcto para este tipo de componentes; la labor del diseño de la solución debe considerar todas la necesidades presentes y futuras del módulo.

---

15 Es importante notar que este patrón se originó debido dadas las limitaciones de ediciones previas de Java EE, sin embargo el tipo de clases y objetos que describe son aplicables a otras tecnologías y modelos. Para la especificación tradicional de este patrón consultar http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html .

Como nota adicional comentamos que estas relaciones pueden representarse de diversas maneras en la capa de datos. Por ejemplo, se puede definir una tabla por cada elemento en la herencia o se puede representar con una columna que indique si una persona es alumno o no; simplemente se puede indicar que aquellas personas que tengan un curso asociado son alumnos y, de hecho, el concepto de alumno no estaría presente en la capa de datos.

Estas discrepancias, como ya mencionamos, se deben a que las consideraciones en diseño de un componente y otro son distintas, lo cual, lejos de verse como un reto, debe tomar la forma de una oportunidad de dar a cada uno su identidad y diseño de manera que cumplan su propósito.

La independencia de estas capas permitirá, como veremos en la sección A.IV.IV, mantener la compatibilidad entre diversas versiones de una misma función de negocio.

Anexo A.IV.I.III: Componente de persistencia

Definidos el modelo del problema y la capa de datos, la función de un componente de persistencia puede ser definida simplemente como un traductor entre ambos modelos. El componente de persistencia establece las funciones para transformar un objeto, o un conjunto de ellos, en elementos de la capa de datos, garantizando su registro y almacenamiento. Adicionalmente, permite la localización de datos y los entrega a la capa de lógica como objetos del modelo del problema.

Es importante que las funciones de localización y almacenamiento de objetos no se extiendan a la capa de lógica, para mantener la independencia de ésta de los cambios en la manera en la que se realiza el almacenamiento en la capa de datos. Por ello es necesario que este componente establezca interfaces para realizar estas funciones de manera general y flexible, absteniéndose de definir operaciones que impliquen lógica en términos de negocio.

Lo anterior no implica que se definan funciones sobre elementos abstractos, de la manera en la que una herramienta ORM[16] lo haría; al contrario, se deben identificar los usos que se le dará a esta capa y presentarlos en términos que eviten propagar los detalles de la implementación o el almacenamiento de los datos a la capa de negocio.

Continuando con nuestro ejemplo, en esta capa debe existir una función que registre a un alumno en la capa de datos. Más aún, esta capa puede relacionar al alumno correctamente con elementos comunes preexistentes, como carrera o plan de estudios, ya que estas relaciones son particulares de la capa de datos. De hecho, la capa de persistencia puede tomar la decisión de cómo almacenar a la persona asociada a este alumno, y por lo tanto es probable que una función para almacenar personas no tenga razón de ser en las interfaces que este módulo expone.

Otro ejemplo del aislamiento que debe tener esta capa con respecto a la de negocio, se expresará en la manera en la que se realizan las consultas. Un mecanismo flexible para

16 Siglas en inglés de mapeo relacional de objetos (Object Relational Mapping).

realizarlas puede implicar una manera de combinar criterios de búsqueda para localizar cierto tipo de objetos, sin exponer los detalles de la implementación del componente de persistencia.

Si usamos Hibernate, o alguna otra tecnología similar, podemos inclinarnos a usar las interfaces y mecanismos propios de esta herramienta para exponer nuestras propias interfaces, digamos usando un objeto de tipo `Criteria` como argumento de un método de localización. Esto no debe tomarse como una alternativa, porque expone la capa de negocio a la tecnología subyacente en la implementación de la capa de persistencia, eliminando la futura libertad de cambiar la implementación a otra tecnología.

Ejemplifiquemos los puntos anteriores con el siguiente fragmento de código:

```
interface AccesoAlumnos {

      ...

      List<Alumno> buscaAlumnos(Criteria parametros);

      ...

}
```

Esta interfaz fuerza a aquellos que la usen a conocer el funcionamiento del tipo Criteria, conocimiento que no debería ser necesario para la lógica de negocio.

El verdadero valor de la capa de persistencia radica en aislar la lógica de negocio de los detalles de almacenamiento y reducir de esta forma la complejidad de la solución.

Observemos que esta forma vuelve la implementación de esta capa superflua, ya que no reduce de ninguna manera la complejidad, y en cambio fomenta que todos los elementos relevantes de esta capa puedan encontrarse en las capas de negocio superiores, con las consecuencias que esto trae para la longevidad de esa base de código.

Un segundo intento de esta funcionalidad podría ser:

```
interface AccesoAlumnos {

      ...

      Alumno buscaPorCuenta(String buscado);

      List<Alumno> buscaPorGrupo(Grupo seleccionado);

      List<Alumno> buscaPorClaveGrupo(String identificador);

      ...

}
```

El contrapeso a las formas muy generales surge al crear funciones para buscar el mismo tipo de elementos por criterios distintos, en funciones separadas, aun cuando éstas se localicen en la misma clase o interfaz. El problema que surge con el tiempo en este esquema es que un cambio en la manera de representar el elemento impacta diversos puntos, lo que puede llevar a un comportamiento heterogéneo de una misma operación.

Ciertamente, si las funciones se localizan en diversos puntos del componente de persistencia, este problema se incrementa, e incluso fomenta una duplicación de código, esfuerzo y riesgos.

Aun cuando todos los casos de uso del ejemplo pueden ser razonables y aplicables a la lógica de negocio, constituyen un patrón peligroso, ya que no sólo podemos imaginar que sus implementaciones serán muy similares, llevando a una repetición de código propensa a errores, sino que cada nuevo uso o cambio en la capa de lógica deberá estar respaldado por un cambio en esta capa, eliminando las ventajas de la separación de responsabilidades.

El enfoque recomendado es hacia operaciones de carácter particular al módulo con formas flexibles. Por ejemplo:

```
interface AccesoAlumnos {
     ...
     ResultadoConsulta<Alumno> buscaAlumnos(ConsultaAlumnos criterios);
     ...
}
```

Los parámetros pueden estar definidos de manera específica, si así se observa necesario, como en este ejemplo:

```
interface ConsultaAlumnos {
     Alumno getPlantilla();
     void setPlantilla(Alumno ejemplo);
     void setLimiteResultados(int numero);
     ...
     void setComparacionNombresExacta(boolean exacta);
     ...
}
```

O como en el caso de los resultados de este ejemplo, pueden ser tipos flexibles con funciones que permitan expresar combinaciones variadas, sin necesidad de entrar en detalles de implementación.

```
interface ResultadoConsulta<Tipo> {
     ...
     List<Tipo> getResultados();
     int resultadosRecuperados();
```

```
      int resultadosTotales();
      boolean hayAdicionales();
      boolean consultaExitosa();
      ...
}
```

La forma precisa de la interfaz evidentemente deberá ser determinada por las necesidades del módulo; sin embargo, los principios de encapsulamiento y flexibilidad deberán respetarse.

Anexo A.IV.I.IV: Capa de lógica

La capa de lógica de un módulo es el centro del funcionamiento de éste, y por ello es la que debe tener mayor libertad en su diseño e implementación.

La capa de lógica deberá representar el proceso que busca automatizar el módulo, recurriendo a las funciones de las otras capas o incluso de otros módulos, cuando esto sea necesario.

Toda la toma de decisiones del módulo debe ocurrir en esta capa, aun cuando para ello se requieran de elementos residentes en la capa de datos. Los principios de diseño y arquitectura deben tener su primera expresión en esta capa, manteniendo los distintos niveles del proceso ordenados y vinculados de acuerdo con su nivel de abstracción[17].

Aun cuando las transformaciones de los datos son parte vital de la lógica, dichas transformaciones se deben limitar al modelo del problema. Esta capa debe abstenerse de implementar búsquedas u operaciones sobre la capa de datos. Es labor de los componentes de persistencia mantener la sincronización entre el modelo del problema y la capa de datos.

Un punto requerido por la ATI es que los componentes que pertenezcan a la capa de lógica establezcan una *interfaz de servicio*. Dicha interfaz de servicio deberá representar las tareas del proceso de manera que puedan ser invocadas por las capas superiores.

Como mencionamos, una forma obvia de la capa de interacción es la interfaz de usuario. Sin embargo, es muy importante que las funciones ofrecidas por la capa de lógica sean orientadas al proceso y no a la interfaz de usuario.

Un precepto de esta capa es que es la que debe tomar las decisiones; en este sentido ésta debe ser lo más “inteligente” posible en términos de negocio y al mismo tiempo debe mantenerse aislada de los particulares de las interfaces de usuario a las que le proporciona todos los elementos necesarios para que éstas realicen su trabajo.

Es decir, en términos de negocio las capas superiores deben ser “ignorantes” de su

17 Para lineamientos sobre este tema, consultar *Clean Code: A Handbook of Agile Software Craftsmanship,* Prentice Hall PTR; 1 edition (August 11, 2008), Robert C. Martin (editor).

propósito, y limitarse a cumplir con las funciones de comunicación entre los usuarios y el proceso.

A manera de ejemplo, consideremos el proceso reducido de inscripción con sus pasos de negocio:

- Establecer identidad y cursos que desea inscribir.
- Capturar requisitos necesarios.
- Decidir sobre la inscripción.

Las interfaces de servicio deben proporcionar maneras para realizar estas funciones; por ejemplo, después de capturar los cursos que desea inscribir, una función debe capturar dicha información y proceder a tomar una decisión sobre los requisitos necesarios.

Al recibir los requisitos, deberá pronunciarse sobre si son adecuados o no y, finalmente, tomar una decisión sobre si la persona puede inscribirse sin mayor trámite o si debe solicitar una autorización.

Una segunda serie de funciones que pueden identificarse como necesarias son las de acceso a los elementos de catálogos y estructura, necesarios para el funcionamiento de la interfaz. Estos elementos en realidad pertenecen al proceso en un paso inicial, que es el de determinar las opciones o datos a capturar. En interfaces simples, esto es a menudo una respuesta fija, y por ello esta información se plasma de manera estática en formularios y similares. En sistemas con interfaces un poco más complejas, esto es parte de las decisiones que la lógica debe tomar.

La granularidad de las tareas y, por lo tanto, de las funciones que las interfaces de servicio exponen, deben depender del proceso y no de la manera en la que éste se ejecuta. Por ejemplo, si una interfaz de usuario ejecuta tres pasos del proceso como resultado de una sola interacción, esto no es justificación suficiente para colapsar estos tres pasos en una sola función. De manera simétrica, si una determinada interfaz de usuario requiere de tres interacciones para completar un paso en el proceso, es suficiente que los pasos intermedios permanezcan únicamente como funciones de presentación sin realizar llamadas a la capa de lógica.

Anexo A.IV.I.V: Interfaces de usuario

Las interfaces de usuario son, sin duda, uno de los principales focos de atención para todo módulo. Es en las interfaces de usuario donde se materializa el proceso de negocio desde el punto de vista del usuario.

Las interfaces de usuario deben, hasta cierto punto, también reflejar el proceso de negocio, pero las consideraciones de esta capa deben tomar otra tonalidad. El propósito de un componente de interfaz de usuario debe ser lograr la comunicación entre el usuario y el sistema, habilitando al usuario para completar las tareas que le corresponden.

En las interfaces de usuario el criterio que debe prevalecer es el de facilidad de uso, claro está sin descuidar el objetivo que cada interfaz tiene.

Por ningún motivo las interfaces de usuario deben contener lógica del proceso; en este sentido deben, en la medida de lo posible, ser independientes de éste. Sin embargo, lo anterior no significa que las interfaces no sean componentes sofisticados o con conocimiento implícito. Al contrario, en esta capa se encuentran implementadas importantes áreas del funcionamiento del módulo. Un ejemplo destacado es la lógica de navegación, la cual permite a los usuarios ubicarse en distintas interfaces, de acuerdo con la operación que busquen completar.

Otra de las funciones que se encuentran contenidas en las interfaces de usuario son la de validación de parámetros de entrada desde le punto de vista sintáctico; es decir, de la forma en la que se reciben dichos parámetros.

Ejemplos de lo anterior incluyen formatos de fechas, cantidades y otros elementos especificados como precondiciones para las funciones que las *interfaces de servicio*[18] ofrecen. Es importante que las interfaces de usuario, si bien no realicen las validaciones de la semántica de la información que proporcionan a la capa de lógica, sí manejen adecuadamente los errores que en dicha validación produzca la capa de lógica.

Con respecto a la correspondencia con la capa de lógica, mencionemos que una interfaz de usuario no necesariamente corresponde uno a uno con las interfaces de servicio o las funciones de ésta. Como ya mencionamos, puede darse el caso que una misma interfaz de usuario permita la ejecución de dos o más funciones de las interfaces de servicio. Más aún, lo inverso es también posible, ya que una interacción del usuario con la interfaz no necesariamente implica una interacción con la capa de lógica. Lo anterior se debe a que puede haber funciones de negocio para las que obtener del usuario todos los parámetros a utilizar implique varias pantallas o interacciones con las mismas.

En nuestro ejemplo, pensemos que para dar inscripción a un individuo requerimos establecer su identidad, vía la captura de su número de cuenta. Esto puede ser una sola interacción del usuario al teclear su número de cuenta, lo que desatará una llamada a la función que determina el resto de sus datos.

El ejemplo de interacciones múltiples del usuario que se reflejan en una sola interacción de negocio sucede a continuación, donde el operador selecciona o captura la información de cada uno de los cursos que desea inscribir antes de proceder con el siguiente paso, que consiste en enviar la información para determinar si se inscribe de inmediato o requiere autorizaciones.

Como lineamiento de diseño para esta capa debemos buscar la creación de componentes de uso general para la representación de los elementos del modelo del problema que este módulo ataca. Contar con estos componentes dará mayor flexibilidad a las interfaces de usuario, permitiendo la integración futura con otros módulos.

En los componentes de presentación es donde encontramos una mayor diversidad de bibliotecas o frameworks disponibles para la creación de interfaces.

A continuación presentamos una lista de las características preferidas por la ATI para la construcción de interfaces de usuario:

---

18 Ver Anexo A.IV.I.IV: Capa de lógica.

- **Orientado a componentes.** Tienen como objetivo permitir la reutilización de elementos de la interfaz de usuario dentro y fuera del módulo. Un framework orientado a componentes, en general, permite la creación de nuevos componentes que se comporten de igual manera en diferentes situaciones. Ejemplos de tecnologías de presentación orientadas a componentes son: JSF[19], Tapestry y WPF[20].
- **Basada en estándares.** Las tecnologías de presentación basadas en estándares permiten a la organización migrar, en un futuro y de manera fácil, a otras implementaciones de dichas tecnologías de presentación. Por ejemplo, al seleccionar la tecnología de presentación es importante identificar si implementa o extiende un estándar. Mojarra, que se incluye con muchos servidores de aplicaciones incluyendo Glassfish, es una implementación de JSF, mientras que IceFaces, por ejemplo, es una biblioteca de componentes de JSF que extiende al estándar.
- **Costo de propiedad considerado.** En la selección de las tecnologías a usar se debe considerar siempre el costo de propiedad, entendido no sólo como costos de licenciamiento de una sola vez o recurrentes, sino también de adopción y mantenimiento. Se prefieren tecnologías que ofrezcan perspectivas de desarrollo futuro. Las tecnologías propietarias que dependen para su mantenimiento de las estrategias de las compañías que las respaldan, presentan una desventaja con respecto a tecnologías Open Source de amplio uso, cuyo desarrollo está respaldado por las comunidades de usuarios. Otro factor que afecta este rubro es la vigencia tecnológica de las soluciones.

En general, la discusión de componentes de interfaces de usuario se ha centrado implícitamente en interfaces de usuario web. Aclaremos que esto no necesariamente es requerido por la ATI, aunque al extenderse a otro tipo de interfaces debemos tomar en cuenta los criterios anteriormente mencionados.

Anexo A.IV.II: Integración de módulos

Ya familiarizados con la forma general de un módulo, descrita en la sección A.IV.I, reconozcamos que no todos los módulos tienen como medio para ejecutar su proceso el uso de una interfaz de usuario, ya sea porque es un proceso automático o debido a que sus funciones son de soporte a otros módulos funcionales.

Para lograr la colaboración entre módulos, la ATI prevé dos mecanismos principales: el primero es la integración de módulos descrito en esta sección y el segundo es la utilización de servicios como se define en la sección A.IV.III.

Los mecanismos de integración de módulos descritos en esta sección deben considerarse para módulos con las siguientes características:

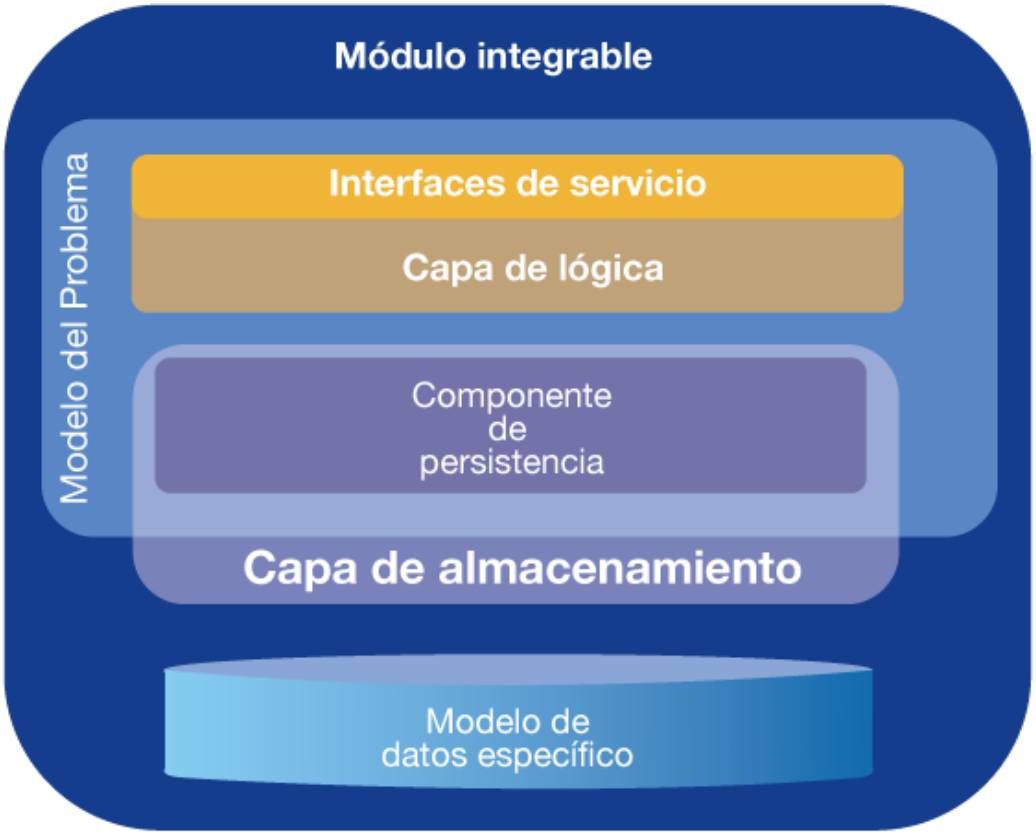
19 Java Server Faces.

20 Windows Presentation Foundation.

- La lógica del módulo a integrar no se ve afectada por replicación de información. Es decir, no hay una afectación del funcionamiento del módulo si se replica la información que utiliza en diversos puntos.
- La lógica del módulo integrador opera sobre las relaciones que los elementos de su propio modelo mantienen con elementos del modelo del módulo integrado.
- El volumen de datos que administra el módulo integrado es mínimo.

El ejemplo más claro de una situación de este tipo es el manejo de los catálogos institucionales donde el modelo y lógica que los rige es no sólo compartido, sino que sirve como base para la creación de otros módulos funcionales.

A continuación presentamos un diagrama de las capas presentes en un módulo integrable.

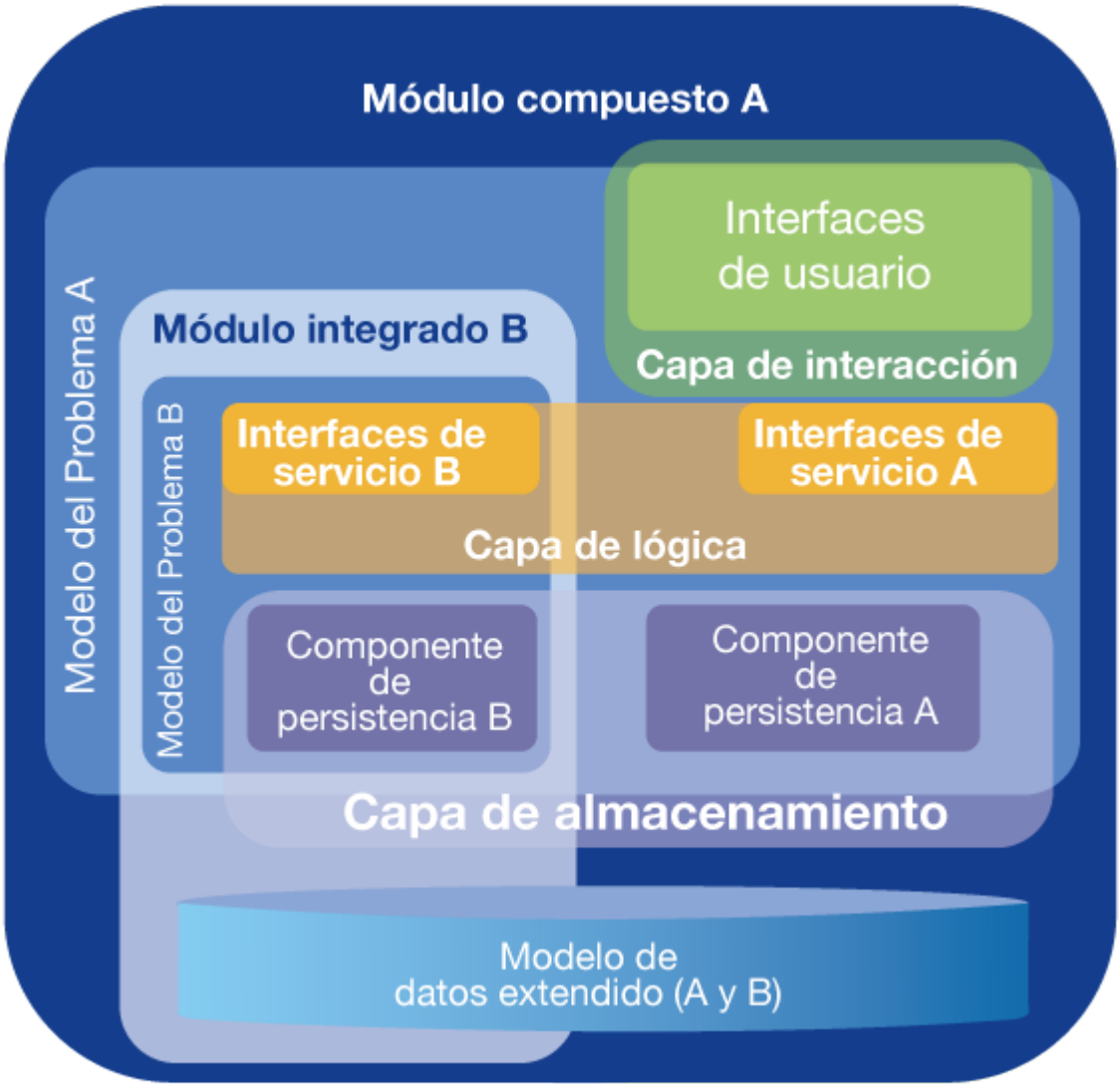


Nótese la falta de una interfaz de usuario propia del módulo, y cómo las interfaces de servicio definen la cara del módulo hacia aquellos que lo utilizan. Esto es debido a que, en caso de que los usuarios requieran interactuar para el funcionamiento de este módulo, se debe considerar una integración como servicio, tal y como se describe en la sección A.IV.III.

Sin embargo, puede darse el caso que el módulo incluya elementos de presentación que los módulos usuarios integren a sus propias interfaces. En este caso, dichos elementos formarán parte de un componente separado a integrarse en la capa de interacción.

En este ejemplo, aun cuando el módulo cuenta con elementos en la capa de presentación éstos no son interfaces de usuario sino meramente elementos de presentación que el componente de interfaz de usuario del módulo integrador puede utilizar.

Un módulo compuesto tiene la siguiente forma:



Como se puede observar, al realizar este tipo de acoplamientos el modelo de datos del módulo compuesto A integra ambos modelos. Esto, en la mayoría de los casos, tiene como consecuencia que se repliquen los datos del modelo de datos de B.

Aclaremos que en este tipo de integración no existe una relación directa entre los componentes de persistencia, aunque el componente de persistencia del módulo integrador A puede acceder a todos los elementos de los modelos extendidos de A y B.

En la capa de lógica las interfaces de servicio del integrado a menudo son utilizadas por la lógica del integrador. Aun cuando no está prohibido por la ATI, no es recomendable que los

componentes de la capa de interacción utilicen las interfaces de servicio de los módulos integrados. En general, las funciones que éstos deben utilizar estarán localizadas en las interfaces de servicio propias del módulo integrador.

Como nota final observemos que la complejidad del módulo crece cuando se utiliza este tipo de integración, y es por ello que su uso debe restringirse a los casos delineados al principio de esta sección; el resto de los casos de cooperación entre módulos se deberán solucionar utilizando los mecanismos de la siguiente sección.

Si ambos enfoques parecen correctos, la arquitectura tecnológica dicta que se tome el enfoque de servicios.

Anexo A.IV.III: Módulos como servicios

El enfoque preferido para la integración de funciones es el orientado a servicios. En esta modalidad encontramos que la capa de interacción contiene un componente de intercambio. Este componente de intercambio tiene la función de publicar las funciones del módulo de manera que otros módulos puedan invocarlas.

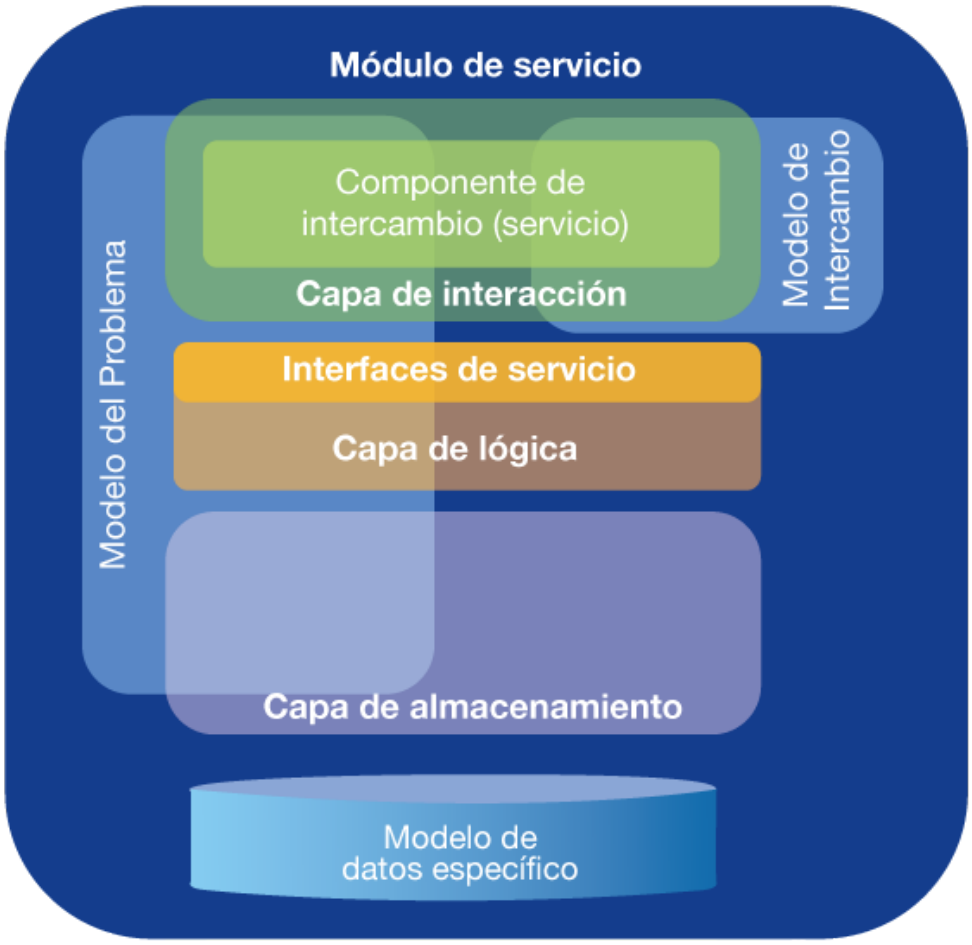
En la mayoría de los casos, las funciones que este componente de intercambio publica son un subconjunto de las que las interfaces de servicio proporcionan.

Existen dos lineamientos que se deben seguir en el diseño de componentes de este tipo:

- **Independencia**, la cual se debe poner de manifiesto tanto en las funciones que se exponen como en los datos que dichas funciones consumen o entregan. Es decir, a pesar de que técnicamente es posible utilizar el modelo del problema como lenguaje en el que se expresan las operaciones del componente de intercambio, esto no se recomiendan ya que introduce un elemento que limita la flexibilidad del componente y rompe con el encapsulamiento del mismo; ante cambios internos del modelo, las interfaces externas tendrían que modificarse. Por ello, es conveniente que el componente de intercambio diseñe su propio modelo adecuado a las necesidades particulares de las funciones que expone; dicho modelo, a menudo, es menos complejo que el modelo del problema.
- **Simplicidad**, que está orientado a facilitar a los usuarios del módulo el acceso a las funciones. Si bien las interfaces de servicio deben proveer todo lo necesario para una función general del módulo, es correcto que en estos componentes se adapten dichas funciones a los casos de uso específicos. El objetivo de este lineamiento es fomentar la integración de los servicios, sin aumentar la complejidad de los sistemas que utilizan las funciones.

En la siguiente gráfica observamos la forma general de un módulo de servicio.

El componente de intercambio, al igual que las interfaces de usuario, tiene las funciones de traducción entre el lenguaje externo, representado por su modelo de intercambio, y el modelo del problema, que permite hacer uso de las interfaces de servicio.



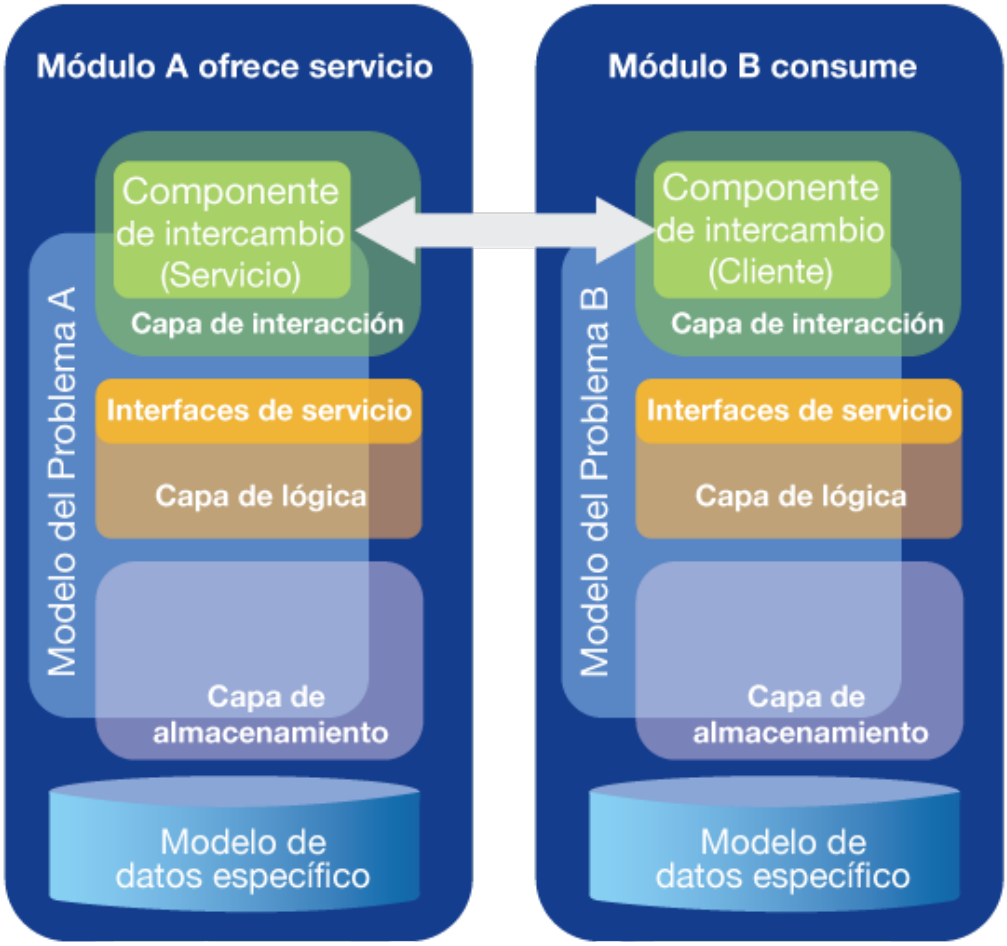
Las tecnologías utilizadas para implementar un componente de intercambio pueden ser variadas. Sin embargo, debe prestarse atención a las recomendaciones expresadas en la *Guía de manejo de información,* sobre todo en lo correspondiente a las medidas de seguridad requeridas de acuerdo con la información sobre la que operen las funciones de este componente.

Como regla general, deben utilizarse mecanismos independientes de la plataforma en la creación de estos componentes; en concreto deben usarse webservices, ya sea basados en SOAP o RESTful webservices, siempre cuidando la interoperabilidad entre plataformas.

Otras tecnologías para la ejecución remota, como CORBA, RMI o DCOM, deben evitarse; su uso será acotado a componentes donde el conjunto de clientes no tenga posibilidad futura de expansión, y ambos extremos estén implementados en la misma plataforma. En ningún caso utilizarán esta tecnología componentes de intercambio con entidades ajenas a la dependencia. Lo anterior, derivado del objetivo de maximizar el potencial de reutilización de las interfaces desarrolladas, no sólo para usos actuales sino futuros. Tanto RMI como DCOM

son especificaciones que en la práctica dependen de la plataforma de desarrollo. Por otro lado, CORBA, aunque cuenta con soporte en muchas plataformas, ha demostrado cierta rigidez bajo ambientes altamente distribuidos, en donde los tiempos de repuesta, latencia y restricciones de seguridad contribuyen a un desempeño insatisfactorio.

En la gráfica siguiente mostramos la interacción entre dos módulos mediante el esquema de servicios.



La relación entre los dos componentes se localiza en la capa de interacción. Nótese que el módulo B, quien consume el servicio proveído por el módulo A, cuenta también con un componente de intercambio, en este caso un cliente que accede al servicio definido por el módulo A.

Este componente cliente tiene un flujo contrario al que normalmente encontramos en la capa de interacción. Normalmente, los componentes que encontramos en esta capa realizan

las llamadas a las funciones definidas en las interfaces de servicio. Sin embargo, en el caso de los componentes de intercambio cliente, es la capa de lógica de negocio la que llama a este componente. Por esta razón los componentes de intercambio cliente, en general, deben estar en un componente separado del resto de los componentes de la capa de interacción, ya que de otra forma encontraríamos una dependencia circular. Lo anterior es de particular importancia cuando los componentes se construyan por separado.

Por supuesto, los componentes de intercambio no excluyen la posibilidad de que existan otros componentes en la capa de interacción, por ejemplo interfaces de usuario.

Este tipo de módulos los denominamos híbridos ya que acceden a los servicios tanto usuarios como otros sistemas.

Es en estos módulos donde empezamos a ver el verdadero valor del diseño de las interfaces de servicio, ya que las mismas funciones respaldan a ambos componentes.

Anexo A.IV.IV: Módulos transitorios

Considerando el estado actual de la plataforma tecnológica de la dependencia, la ATI debe prever un periodo de transición en el que los nuevos módulos interactúen con los sistemas existentes. Las razones para requerir esta configuración son múltiples; a manera de ejemplo, consideremos el proceso actual de inscripción conviviendo con un nuevo proceso de control escolar.

Para lograr esta convivencia se deberán generar componentes de compatibilidad; estos componentes residirán en la capa de almacenamiento, ya que dada la arquitectura de las aplicaciones actuales la integración más viable es a nivel de base de datos, estableciendo las operaciones necesarias en dos sentidos:

- **Acceso a datos en el modelo histórico**. Este tipo de operaciones permitirán a los nuevos módulos operar con la información contenida en aplicaciones existentes bajo una arquitectura no apegada a la ATI. En este tipo de accesos, el nuevo módulo es el que hace uso de la información existente para realizar sus operaciones. Este tipo de función es poco recomendado, ya que en general las funciones de las interfaces actuales requieren un soporte más robusto al existente.
- **Mantenimiento de datos.** La manera en la que se puede establecer la compatibilidad con aplicaciones existentes es permeando los cambios que el nuevo módulo realiza a un modelo de datos histórico, accedido por otras aplicaciones aún no migradas a la nueva arquitectura. Para realizar estas funciones se sugiere, como siempre, el establecimiento de una interfaz de servicio para el componente de persistencia, y una implementación temporal que delegue de manera atómica los cambios, tanto a una implementación de persistencia en el modelo nuevo como a la interfaz de compatibilidad. De esta manera, al terminar el periodo de transición, simplemente se afecta la configuración para evitar este paso.

Los módulos transitorios, como su nombre lo indica, presentarán este tipo de arquitectura durante un periodo limitado de operación; en general indicarán dependencias de servicio que existirán entre dos módulos una vez completada la migración.

En la gráfica siguiente observamos la distribución de componentes en un módulo transitorio.

Es importante notar que el modelo del problema no necesariamente es el usado por el módulo de compatibilidad; sin embargo, esto puede ser conveniente para simplificar la integración con la capa de lógica.

Anexo A.IV.V: Un caso extremo

Para cerrar la descripción estructural, observemos que los conceptos expresados en las subsecciones A.IV.IV y A.IV.III de ninguna manera se excluyen uno a otro, ya que dentro de las necesidades del proyecto los módulos pueden albergar diversas configuraciones de componentes.

Abajo observamos un componente que integra tanto compatibilidad como un componente de intercambio.

Anexo A.V: Desarrollo futuro

La presente guía arquitectónica responde el estado y necesidades actuales de la dependencia. Sin embargo, a pesar de que deliberadamente trata de ser independiente de la tecnología actual, puede requerir revisiones en un futuro, para acomodar nuevas necesidades de negocio o técnicas que tenga la organización. Dicho lo anterior, futuras versiones de esta guía tienen libertad para definir extensiones o redefinir ciertos conceptos; sin embargo, en este proceso se deberán respetar los lineamientos expuestos en la sección A.II de este documento.

Anexo A.V.I: Escenarios previstos

La presente arquitectura no excluye la posibilidad de utilizar software de soporte para la comunicación entre los servicios; en particular, el uso de un *Enterprise Service Bus* (ESB) puede resultar de utilidad una vez que una cierta masa crítica de servicios haya sido creada o migrada de los servicios actuales.

Existen razones considerables por las cuales, en el contexto de TI actual de la dependencia, un ESB aportaría poco valor a la infraestructura. La principal de éstas es que la mayoría de los sistemas traslapan responsabilidades de diversos procesos de negocio y, más aún, la lógica subyacente de estos procesos se encuentra replicada en diversos sistemas e incluso llega a presentar diferencias. Por si esto fuera poco, muchos de los sistemas están fuertemente integrados con las interfaces de usuario, lo que dificultaría su integración a un esquema de servicios como el que los ESB suponen o ayudan a implementar.

Otro escenario previsto por esta arquitectura lo constituyen sistemas distribuidos y poco acoplados, los cuales pueden requerirse dado el carácter nacional de la dependencia. Este escenario simplemente requiere de componentes de intercambio con un servicio centralizado de consolidación de información; más aún, este componente puede alternarse con la capa de almacenamiento, para funcionar en línea o fuera de ésta, en situaciones de contingencia.