Universidad Austral de Chile

Facultad de Ciencias de la Ingenieria
Escuela de Ingenieria Civil en Informatica

DEFINICIÓN DE UNA ARQUITECTURA PARA LA IMPLEMENTACIÓN DE APLICACIONES MÓVILES SOBRE PLATAFORMA .NET

Tesis para optar al Título de:
Ingeniero Civil en Informática

Profesor Patrocinante:
Sr. Juan Pablo Salazar Fernández
Ingeniero Civil en Informática - MBA

MARIA ALEJANDRA URIBE GALLARDO

VALDIVIA – CHILE
2006

Valdivia, 01 de agosto de 2006

De: Juan Pablo Salazar Fernández
Profesor Patrocinante

Ref: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación **"Definición de una arquitectura para la implementación de aplicaciones móviles sobre plataforma** .NET", presentado por la alumna Srta. María Alejandra Uribe Gallardo, mi evaluación del mismo es la siguiente:

Nota: 6,5 (seis, coma cinco).

Fundamento de la nota:

El presente trabajo de titulación se planteó como objetivo definir una arquitectura para la implementación de aplicaciones móviles sobre plataforma .NET.

Este objetivo se cumplió en términos generales, pero con un nivel de profundidad no homogéneo.

Los capítulos que exponen aspectos conceptuales y criterios de diseño están presentados de manera muy clara y precisa, así como también lo referido al contexto y requisitos de la aplicación PER-Móvil.

No obstante, considero restrictivo que se planteara como supuesto que toda la lógica de negocio estaría soportada por sistemas legacy, sin profundizar más en definiciones arquitectónicas a este nivel, cuando en la práctica lo que ocurre muchas veces es que es necesario incorporar lógica de negocios que no existe o rediseñar los procesos dadas las posibilidades que ofrece la tecnología móvil. En la misma línea, hubiese sido interesante profundizar en la utilización de patrones y exponer con mayor detalle la aplicación descrita.

Aspecto	**Evaluación**
Cumplimiento de objetivos	6.3
Satisfacción de alguna necesidad	7.0
Aplicación del método científico	6.5
Interpretación de los datos y obtención de conclusiones	6.0
Originalidad	6.2
Aplicación de criterios de análisis y diseño	6.7
Perspectivas del trabajo	6.5
Coherencia y rigurosidad lógica	6.5
Precisión del lenguaje técnico	6.5

Sin otro particular, saluda atentamente a usted,

Juan Pablo Salazar Fernández
Profesor Patrocinante

VALDIVIA, 11 de Mayo del 2006

DE: GLADYS MANSILLA GOMEZ.
A : JUAN PABLO SALAZAR. DIRECTOR ESCUELA ING. CIVIL EN INFORMATICA

MOTIVO INFORME TRABAJO DE TITULACION

Nombre Trabajo de Titulación: **DEFINICIÓN DE UNA ARQUITECTURA PARA LA IMPLEMENTACIÓN DE APLICACIONES MÓVILES SOBRE PLATAFORMA .NET**

Nombre del alumno: MARIA ALEJANDRA URIBE GALLARDO

Nota: 7.0 siete
(en números) (en palabras)

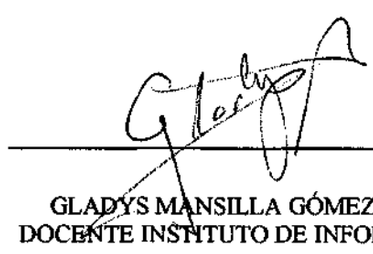
Fundamento de la nota:

- Este trabajo de tesis presenta un trabajo de gran valor toda vez que el alumno logró con éxito incorporar una gran cantidad de nuevos conocimientos a su formación y desarrollar una aplicación útil en el área de negocios en que está inmersa, desarrollo que es posible aplicar a proyectos similares.

- En este trabajo es posible destacar la claridad con que el alumno utiliza la nueva terminología relativa a servicios web y tecnología inalámbrica adquirida durante su trabajo.

- En la realización de este trabajo de titulación se alcanzan plenamente los objetivos planteados al inicio.

- La presentación y redacción del informe están bien elaboradas, abarcando tópicos que inciden directamente en esta tesis y expresado en un lenguaje formal apropiado.



GLADYS MANSILLA GÓMEZ
DOCENTE INSTITUTO DE INFORMATICA

Universidad Austral de Chile

Instituto de Informática

Valdivia, 10 de julio de 2006.

De : Luis Hernán Vidal Vidal.
A : Sr. Juan Pablo Salazar.
Director de Escuela de Ingeniería Civil en Informática.
Ref. : Informa Calificación Trabajo de Titulación.

MOTIVO: Informar revisión y calificación del Proyecto de Título "Definición de una arquitectura para la implementación de aplicaciones móviles sobre plataforma .NET", presentado por la alumna María Alejandra Uribe Gallardo, que refleja lo siguiente:

Se logró el objetivo planteado que permitió definir una arquitectura para soluciones móviles implementadas con tecnologías Microsoft .NET.

La revisión hecha sobre los estándares y tecnologías, junto al desarrollo propuesto, se presenta como una buena referencia para futuros trabajos en esta área.

Cumplimiento del objetivo propuesto.	7,0
Satisfacción de alguna necesidad.	7,0
Aplicación del método científico.	7,0
Interpretación de los datos y obtención de conclusiones.	6,5
Originalidad.	6,5
Aplicación de criterios de análisis y diseño.	7,0
Perspectivas del trabajo.	7,0
Coherencia y rigurosidad lógica.	7,0
Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración.	7,0
Evaluación Tesis.	**6,9**

Por todo lo anterior expuesto califico el trabajo de titulación de la alumna María Alejandra Uribe Gallardo con nota 6,9 (seis coma nueve).

Sin otro particular, se despide atentamente.

UNIVERSIDAD AUSTRAL DE CHILE
INSTITUTO DE INFORMATICA
FACULTAD DE CIENCIAS DE LA INGENIERIA

Ing. Luis Hernán Vidal Vidal.
Profesor Instituto de Informática.
Facultad de Ciencias de la Ingeniería.
Universidad Austral de Chile.

Resumen

Durante los últimos años, Chile ha experimentado gran crecimiento en su economía, en el cual, las tecnologías de la información aportan la mayor cantidad de calificadores para todas las organizaciones y en muchas ocasiones el intercambio de información en tiempo real constituye el éxito y la eficacia para aquellas compañías basadas en la movilidad de sus integrantes. Es más, las técnicas de negociación y ventas han evolucionado y el contar con información privilegiada en el momento adecuado y en el lugar correcto es posible gracias a los avances de las tecnologías de comunicación inalámbrica, las cuales permiten a las empresas automatizar ciertas actividades de venta utilizando dispositivos móviles.

Las soluciones móviles empresariales tienen como principal objetivo facilitar el intercambio de información valiosa entre los trabajadores que operan en terreno y los sistemas Legacy de su organización. Este trabajo se enfoca en el desarrollo especializado de soluciones móviles empresariales con tecnologías Microsoft .Net y propone la especificación de una arquitectura para la construcción de esta aplicaciones aprovechando las ventajas que brindan las arquitecturas orientadas a servicios y servicios Web.

Summary

During the past few years, Chile has experienced great growth in its economy, in which, the Information technologies contribute the greater amount of qualities for all the organizations and in many occasions the real-time Information exchange constitutes success and effectiveness for those companies based on their member's mobility. It is more, negotiation and sales techniques have evolved and to have privileged information at the suitable moment and the correct place it is possible thanks to wireless Communications technology's advances, which allow the companies to automate activities of sale using mobile devices.

Mobile enterprise solutions have like main target to help the exchange of valuable Information between the workers in motion and the organization's systems. This work focuses in the specialized development of mobile enterprise solutions using Microsoft technologies. Net and proposes a software architecture specification in order to construct mobile solutions and taking advantage of the benefits that offer Service-Oriented Architecture and Web Services.

A mis padres, infinitas gracias por sus sacrificios y paciencia durante mis años de estudio.

A mis queridos Jorge, Erna, Francisco y Marietta, por su incondicional apoyo y el enorme cariño que siempre me han regalado, siempre estaremos juntos.

y a Cristian ...le dedico mi vida entera por hacer de ella una historia de amor.

INDICE DE CONTENIDOS

CAPÍTULO I: INTRODUCCIÓN GENERAL 4

1. PRESENTACION 5

2. ANTECEDENTES 7

3. OBJETIVOS GENERALES Y ESPECIFICOS 10

3.1 OBJETIVO GENERAL 10
3.2 OBJETIVOS ESPECÍFICOS 10

CAPÍTULO II: PLATAFORMA DE APLICACIONES Y SERVICIOS .NET 11

1. INTRODUCCIÓN 12

2. PLATAFORMA DE DISEÑO DE APLICACIONES Y SERVICIOS NET 14

3. CAPA DE PRESENTACIÓN Y SUS COMPONENTES 15

3.1 COMPONENTES DE INTERFAZ DE USUARIO 15
3.2 COMPONENTES DE PROCESO DE USUARIO 17

4. CAPA DE COMPONENTES EMPRESARIALES 19

4.1 FLUJOS DE TRABAJOS EMPRESARIALES 20
4.2 COMPONENTES EMPRESARIALES 21
4.3 ENTIDADES EMPRESARIALES 23

5. CAPA DE DATOS Y SUS COMPONENTES 24

5.1 COMPONENTES LÓGICOS DE ACCESO A DATOS 25
5.2 AGENTES DE SERVICIOS 27

6. .NET COMPACT FRAMEWORK 29

6.1 ARQUITECTURA DE .NET COMPACT FRAMEWORK 30
6.2 DOMINIOS DE APLICACIÓN DE .NET COMPACT FRAMEWORK 32
6.3 ASPECTOS RELACIONADOS CON LA SEGURIDAD 32

CAPÍTULO III: PATRONES 34

1. INTRODUCCIÓN 35

2. DEFINICIÓN 35

3. MARCO GENERAL DE PATRONES MICROSOFT 37

3.1 PATRONES DE ARQUITECTURA 39
3.2 PATRONES DE DISEÑO 40
3.3 PATRONES DE IMPLEMENTACIÓN 42

4. CLUSTER DE PATRONES PARA SOLUCIONES EMPRESARIALES 43

5. ARQUITECTURA BASADA EN SERVICIOS 44

5.1 CONTRATO 45
5.2 SERVICIOS WEB 46
5.3 PATRONES DE SERVICIO 48
5.3.1 Service Interface 48
5.3.2 Service Gateway 52

SIGUIENTE CAPÍTULO IV: PER-MÓVIL, UN CASO REAL DE SOLUCIÓN MÓVIL 57

1. INTRODUCCIÓN 58

2. QUÉ CONTIENE ESTE CAPÍTULO 58

3. ORIGEN DE LA NECESIDAD 59

3.1 NECESIDADES GENERADAS POR EL ENTORNO DEL MERCADO 60
3.2 NECESIDADES INTERNAS DE LA ORGANIZACIÓN 61

4. DESCRIPCIÓN DE LA NECESIDAD 62

5. CARACTERÍSTICAS DE LA SOLUCIÓN 63

6. OBJETIVO DE LA SOLUCIÓN MÓVIL EN ESTUDIO 64

7. CONTEXTO DE LA SOLUCIÓN EN ESTUDIO 64

8. PER-MÓVIL IMPLEMENTADO SOBRE UNA ARQUITECTURA ORIENTADA A SERVICIOS 66

8.1 PER-MÓVIL ES UNA SOLUCIÓN BASADA EN COMPONENTES 68
8.1.1 Capa de Presentación y sus componentes en PER-Móvil 68
8.1.2 Capa Empresarial 70
8.1.3 Capa de datos y sus componentes 72

9. CONCLUSIÓN DEL CAPÍTULO 74

CAPÍTULO V: UNA SOLUCIÓN MÓVIL GENÉRICA 75

1. INTRODUCCIÓN 76

2. SUPUESTOS DEL MODELO 78

3. SEGURIDAD 80

3.1 AUTENTICACIÓN 80
3.2 AUTORIZACIÓN 82
3.3 COMUNICACIÓN SEGURA 85
3.4 AUDITORIA 86

4. ADMINISTRACIÓN OPERATIVA 87

4.1 ADMINISTRACIÓN DE EXCEPCIONES 87
4.2 CONFIGURACIÓN DE SERVICIOS 88

5. COMUNICACIONES 91

6. CAPA DE PRESENTACIÓN 92

7. CAPA DE COMPONENTES EMPRESARIALES 93

7.1 COMPONENTES EMPRESARIALES 93
7.2 ENTIDADES EMPRESARIALES 94

8. CAPA DE DATOS 97

8.1 DATOS CENTRALIZADOS EN SISTEMAS LEGACY 97
8.2 ALMACENAMIENTO DE DATOS EN DISPOSITIVO MÓVIL 98
8.3 AGENTES DE SERVICIOS 99
8.4 PROVEEDOR DE SERVICIOS WEB 101

CAPÍTULO VI: CONCLUSIONES 102

BIBLIOGRAFIA 107

Indice de Figuras y Tablas

Figura 1: Esquema simplificado de una aplicación y sus capas lógicas… 12
Figura 2: Arquitectura de aplicaciones .Net basada en componentes…… 14
Figura 3: Componente empresarial típico…….. 21
Figura 4: Componentes lógicos de acceso a datos…….. 26
Figura 5: Esquema general de la Arquitectura .Net Compact Framework..... 30
Figura 6: Marco de patrones de Microsoft…….. 37
Figura 7: Cluster de patrones de arquitectura……. 39
Figura 8: Diagrama de invocación de un servicio en Arquitecturas orientadas a servicios……. 45
Figura 9: Elementos de un servicio……. 50
Figura 10: Relaciones Service Gateway…….. 54
Figura 11: Contexto general solución PER-Móvil……. 64
Figura 12: PER-Móvil, una solución orientada a servicios……. 66
Figura 13: Vista de componentes de PER-Móvil……. 73
Figura 14: Esquema de arquitectura generica para aplicaciones móviles con reutilización de lógica de negocio……. 76
Figura 15: Directivas de seguridad……. 80
Figura 16: Diagrama de secuencia del proceso de autenticación…… 81
Figura 17: Diagrama de secuencia para autenticación y autorización…… 83
Figura 18: Diagrama red de comunicación…….. 85
Figura 19: Directivas de la Administración Operativa……. 87
Figura 20: Diagrama de secuencia de invocación de servicio Web con uso de un catálogo de servicios…….. 89
Figura 21: Directiva de comunicaciones……. 91
Figura 22: Colaboración de componentes empresariales…….. 94
Figura 23: Diagrama de secuencia de colaboración para componentes de la arquitectura……. 95
Figura 24: Acceso a datos a través de agentes de servicio…….. 97
Tabla 1: Clasificación de patrones de Diseño según GOF……. 40
Tabla 2: Clasificación de Cluster de Patrones…….. 43
Tabla 3: Comparativa de arquitectura de PER-Móvil y la arquitectura propuesta en trabajo de titulación…….. 105

CAPÍTULO I: INTRODUCCIÓN GENERAL

1. PRESENTACIÓN

Definición de una arquitectura para la implementación de aplicaciones móviles sobre plataforma .Net

Durante los últimos años nuestro país ha estado experimentando un ciclo de expansión y crecimiento en la economía, en el cual, las tecnologías de la información aportan la mayor cantidad de calificadores para todas las organizaciones y en muchas ocasiones el intercambio de información en tiempo real constituye el éxito y la eficacia para aquellas compañías basadas en la movilidad de sus integrantes. Más aún, las técnicas de negociación y ventas han evolucionado y el contar con información privilegiada en el momento adecuado y en el lugar correcto ya es posible, lo cual representa una herramienta fuertemente competitiva.

Los avances que ha tenido la tecnología inalámbrica durante los últimos años permiten a las empresas, por ejemplo, pensar en automatizar ciertas actividades de venta mediante la utilización de hardware móvil. Por otro lado, la popularidad de los dispositivos móviles está experimentando un gran crecimiento debido a la aceptación que tienen por parte de los usuarios. Así vemos que cada vez aparecen nuevos y mejorados modelos de dichos dispositivos, Pocket PCs y otras PDAs, teléfonos móviles y teléfonos inteligentes, híbridos entre teléfono y PDA[1], etc.

La plataforma .Net de Microsoft ha ido ampliando sus herramientas de desarrollo de aplicaciones y proporciona a los dispositivos móviles el mismo modelo completo de aplicaciones que ofrece para todas las aplicaciones .Net de

[1] Sigla en inglés de Asistente Digital Personal.

escritorio. Gracias a los WS[2] es posible que los sistemas les resulte muy fácil comunicarse entre sí, independientemente de la plataforma que utilicen.

"Los WS permiten una amplia interoperabilidad independientemente de las tecnologías subyacentes y proporcionan además un modelo de programación de acoplamiento flexible" [MIC2, 2002]. Todas estas ventajas tecnológicas se pueden utilizar en una solución móvil de carácter empresarial mediante la utilización de la lógica .Net y sus tecnologías.

En resumen, las soluciones móviles tienen como principal objetivo facilitar el intercambio de información valiosa en un instante preciso, entre los trabajadores que operan en terreno y los sistemas con los que cuenta su organización.

Conscientes de que las aplicaciones móviles llevan a las empresas a aumentar su productividad y dar valor agregado a los actuales sistemas de apoyo al negocio, se presenta este tema como trabajo de titulación que innova y que se enfoca al desarrollo especializado de soluciones móviles con tecnologías Microsoft .Net.

En el presente documento se propone la especificación de una arquitectura para la construcción de aplicaciones móviles basadas en la plataforma .Net de Microsoft.

[2] Sigla en inglés de Servicios Web

2. ANTECEDENTES

Es sabido que las organizaciones poseen, desde hace mucho tiempo, sistemas de información que automatizan muchas de sus actividades y procesos. En el siguiente comentario se ve claramente reflejada esta idea.

> *"A medida que las Tecnologías de la Información (TI) han aumentado su capacidad y consolidado su presencia en la productividad de las empresas, estas últimas las evalúan como un recurso cada vez más determinante para alcanzar el éxito. Así, cada vez más ha sido común utilizar las TI con el objetivo de obtener ventajas competitivas -lo que se ha denominado "valor estratégico de las TI". Sin embargo, hoy las funciones centrales de las TI (almacenamiento, procesamiento y transporte de datos), están al alcance de todos, por lo que éstas ya han comenzado a transformarse en factores de producción comoditizados, dejando de ser un recurso potencialmente estratégico. Vale decir, las TI dejaron de ser importantes para pasar a ser necesarias."* [CNC, 2005]

Sin embargo, aún cuando los sistemas tecnológicos pasaron a ser parte necesaria del funcionamiento de una organización todavía existen algunas necesidades no cubiertas desde el punto de vista de la disponibilidad en terreno de la información valiosa de una empresa. Más aún, aunque las empresas tienen en sus instalaciones grandes sistemas de almacenamiento y procesamiento de datos, los usuarios finales de los servicios y/o productos ofertados por ellas no cuentan con las ventajas de dichos sistemas en terreno. Pensemos en compañías de seguros, instituciones que entregan servicios bancarios e instituciones que ofrecen algún producto o servicio que requiere el

intercambio de información entre el representante organizacional y el cliente en el momento de la venta. Es aquí precisamente donde se puede dar un valor agregado a los sistemas tecnológicos existentes.

Los dispositivos móviles son, en esencia, computadoras de mano. Son dispositivos que tienen una capacidad de procesamiento y de memoria reducido en comparación a un PC, por lo cual, definir una arquitectura para una solución móvil escalable, de alta disponibilidad y de fácil mantención requiere de un análisis concienzudo respecto del modelo de capas lógicas y las tecnologías con las cuales se va a implementar. Cuando se diseña una aplicación se deben tomar decisiones acerca de cómo se comunican los componentes de dicha aplicación y también cómo se distribuyen, por ejemplo, si es recomendable o no utilizar Servicios Web y si la llamada será centralizada (patrón Proxy) o cada componente consumirá Servicios Web en forma independiente. Para las diferentes capas de una aplicación existen diferentes patrones arquitectónicos que pueden ayudar.

> *"Arquitectura es un conjunto de decisiones significativas acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema y las interfaces entre ellos, junto con su comportamiento, tal como se especifica en las colaboraciones entre esos elementos, la composición de estos elementos estructurales y de comportamiento en subsistemas progresivamente mayores, y el estilo arquitectónico que guía esta organización. Además del comportamiento y la estructura se deben definir las restricciones y compromisos de uso, funcionamiento, flexibilidad al cambio, reutilización y tecnología"* [BOO, 2000].

Lo que se propone es este trabajo de titulación es la definición de una arquitectura para soluciones móviles implementadas con tecnologías .Net con reutilización de lógica empresarial preexiste en sistemas corporativos. La estrategia adoptada para definir esta arquitectura será identificar capas lógicas, sus componentes y responsabilidades e interfaces, que permitan asegurar aspectos tales como robustez, escalabilidad y facilidad de mantención.

La arquitectura elaborada permitirá a los arquitectos de software o responsables del diseño de alto nivel de soluciones móviles aprovechar de manera adecuada las tecnologías .Net y además proporcionará un vocabulario común al equipo técnico involucrado, como consecuencia de la utilización de patrones descritos formalmente con Lenguaje de Modelamiento Unificado.

3. OBJETIVOS GENERALES Y ESPECIFICOS

3.1 Objetivo General

Definir una arquitectura para soluciones móviles implementadas con tecnologías Microsoft .Net.

3.2 Objetivos Específicos

1. Describir la plataforma de aplicaciones y servicios Microsoft .Net.
2. Diseñar una arquitectura lógica de componentes para diseñar soluciones móviles, desde el punto de vista de capas lógicas.
3. Desarrollar una arquitectura específica para la implementación de soluciones móviles con tecnologías .Net.
4. Evaluar y validar la arquitectura definida contra una arquitectura real de una solución móvil empresarial. Para realizar la evaluación de la arquitectura propuesta se tomarán en cuenta los siguientes parámetros:
    a. Escalabilidad de la solución
    b. Reutilización de código
    c. Disponibilidad y Mantenibilidad.

CAPÍTULO II: PLATAFORMA DE APLICACIONES Y SERVICIOS .NET

1. INTRODUCCIÓN

La plataforma .Net plantea el desarrollo en capas. Las capas son, básicamente, agrupaciones lógicas de componentes software que constituyen una aplicación o servicio.

Una capa lógica contiene tipos de componentes agrupados en subcapas, cada componente que pertenece a una misma subcapa realiza un mismo tipo de tarea.

Según los tipos de tarea o funciones que cumplen los componentes dentro de las subcapas se ha podido distinguir 3 niveles de capas:

- Capa de presentación, que se encarga de atender peticiones del usuario.
- Capa empresarial, en donde reside la lógica de negocio de la aplicación y/o procesamiento de información.
- Capa de datos, se comunica con orígenes de datos y servicios externos de información que actúan como orígenes de datos.

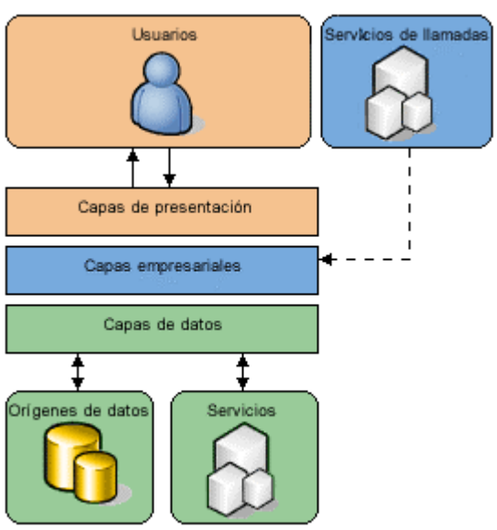


Figura 1: Esquema simplificado de una aplicación y sus capas lógicas.

En este capítulo, se presenta una breve descripción de los componentes agrupados por subcapas para cualquier solución basada en componentes.

2. PLATAFORMA DE DISEÑO DE APLICACIONES Y SERVICIOS NET

A través de los años, se ha trabajado para identificar componentes genéricos de una solución genérica, dado lo anterior, se hace posible contar con un mapa coherente de una aplicación o servicio cualquiera y utilizar este mapa como plano técnico para el diseño de software. Luego del análisis de muchas soluciones empresariales basadas en modelos de capas se puede inferir que existen varios tipos de componentes habituales, en la figura 2 se muestra una ilustración completa en la que se indican estos tipos de componentes.

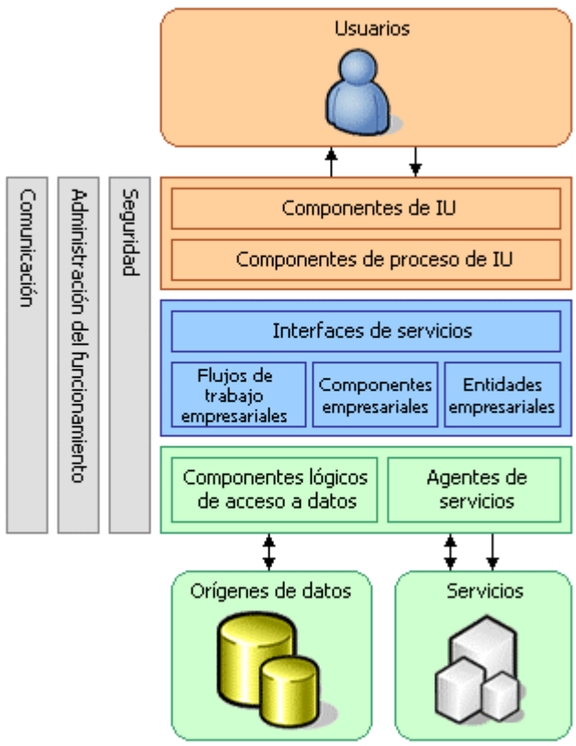


Figura 2: Arquitectura de aplicaciones .Net basada en componentes.

3. CAPA DE PRESENTACIÓN Y SUS COMPONENTES

La gran mayoría de las aplicaciones necesitan proveer al usuario un mecanismo de interacción para poder comunicarse con ella, existen dos tipos de componentes que permiten implementar esta comunicación:

1. Componentes de IU[3]
2. Componentes de proceso de IU

A continuación, se presenta una descripción para cada uno de estos componentes.

3.1 Componentes de Interfaz de Usuario

La capa de presentación se ocupa de mostrar datos al usuario, obtener datos del usuario e interpretar los eventos generados y cambiar el estado de la interfaz misma.

Una interfaz de usuario puede estar representada por medio de una página Web y/o de un formulario que contiene elementos a través de los cuales el usuario ingresa datos y la aplicación acepta las entradas.

Los componentes de la capa de presentación no inicializan, ni menos participan en transacciones, sólo presentan una referencia al componente de proceso de usuario actual en caso de que necesiten mostrar sus datos o intervenir en su estado. Pueden encapsular tanto la funcionalidad de visualización como la de un controlador.

[3] Interfaz de Usuario

Cuando se acepta la entrada de usuario, los componentes de interfaz capturan los datos de usuario y atienden su entrada a los procedimientos de validación, capturan los eventos del usuario y llaman a las funciones de control.

A continuación se despliegan las funciones de las cuales deben ocuparse los componentes de interfaz de usuario.

- Restringir los tipos de datos de entrada de los usuarios.
- Realizar validaciones propias de interfaz, por ejemplo, intervalos de valores permitidos.
- Garantizar la entrada de datos obligatorios.
- Transformar la entrada proporcionada por los controles de usuario a valores necesarios para que otros componentes realicen su trabajo.
- Interpretar las acciones de usuario.
- Utilizar componentes que permitan almacenar en caché.
- Formatear valores
- Procesar datos de una entidad empresarial e informar el estado al usuario.
- Presentar la facilidad de deshacer acciones de usuario.
- Personalizar el aspecto de la aplicación en función de las preferencias del usuario según el tipo de dispositivo utilizado.

3.2 Componentes de proceso de usuario

Durante la interacción del usuario con la aplicación a través de los componentes IU existen sin duda gran cantidad de *procesos predecibles.* Estos procedimientos predecibles pueden ser implementados para facilitar la sincronización y organización de las interacciones con el usuario [MIC, 2002].

Dentro de las responsabilidades de los Componentes de Proceso de Usuario se pueden mencionar:

- Facilitar la coordinación del proceso de usuario y controlar el mantenimiento del estado requerido al visualizar varias componentes IU.
- Coordinar la visualización de los elementos de la interfaz, abstrayéndose de la funcionalidad de procesamiento y adquisición de datos proporcionados por los componentes de interfaz de usuario.
- Encapsular el modo en que las excepciones pueden afectar al flujo de procesos de usuario.
- Realizar el seguimiento del estado actual de la interacción con el usuario.
- Podría eventualmente proporcionar una característica “guardar y continuar después”, mediante la cuál se puede interrumpir una sesión y continuar en otra posteriormente.

Los Componentes de Procesos de Usuario se utilizan en los siguientes casos:

- **Control de actividades de usuarios concurrentes,** permiten simplificar la administración del estado de varios procesos encapsulando lo necesario en un solo componente.
- **Utilización de varios paneles de actividad,** permiten sincronizar y centralizar el estado de varias interfaces o paneles que están presentes en una interfaz de usuario.
- **Aislar las actividades de los usuarios durante la ejecución de transacciones largas,** es posible diseñar procesos de usuario en serie, almacenados en un lugar distinto de los datos empresariales mientras se completa o retoma un flujo de proceso que está en espera.

4. CAPA DE COMPONENTES EMPRESARIALES

La capa de componentes empresariales corresponde a la capa medular de cualquier aplicación, dado que los componentes empresariales implementan la funcionalidad de dicha aplicación. Una aplicación puede realizar procesos que constan de una o varias tareas (caso simples) en los cuales cada tarea se puede encapsular en el método de un componente. En casos más complejos se requieren varios pasos y transacciones de ejecución larga, aquí la aplicación necesita disponer de un modo de organización de tareas y almacenar su estado hasta que el proceso se haya completado [MIC, 2002].

En la arquitectura de aplicaciones de .Net se pueden distinguir tres tipos de componentes empresariales:

1. Flujos de Trabajo empresariales.
2. Componentes empresariales.
3. Entidades empresariales.

A continuación se da una breve descripción de cada uno de estos componentes.

4.1 Flujos de Trabajos Empresariales

Los flujos empresariales organizan el proceso empresarial, son necesarios cuando se necesita administrar un proceso que conlleve varios pasos y transacciones de ejecución larga ó cuando se necesita contar con una interfaz que habilite la comunicación o contrato con servicios externos.

Los flujos de trabajo empresariales se utilizan cuando existe una secuencia concreta en el proceso empresarial y se diseñan componentes empresariales para encapsular cada uno de los pasos individuales en el proceso y organizar dichos componentes en un flujo empresarial [MIC, 2002].

4.2 Componentes Empresariales

Los componentes empresariales implementan las reglas empresariales, deben ser independientes del almacén de datos y de los servicios que se necesitan para realizar las tareas, además deben ser coherentes desde el punto de vista del significado y de la transacción. Los componentes empresariales pueden generar transacciones atómicas.

A continuación se muestra un caso típico de componente empresarial. Este componente interactúa con la capa de presentación, con agentes de servicios y componentes lógicos de acceso a datos [MIC, 2002].

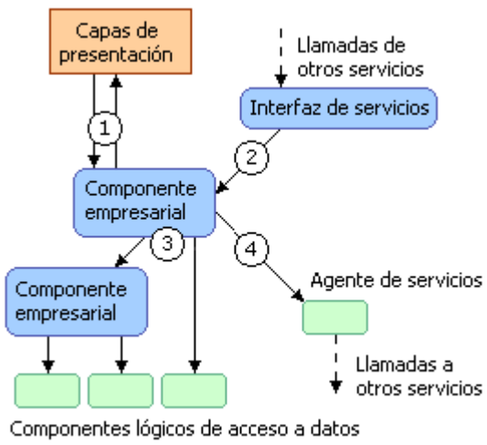


Figura 3: Componente empresarial típico.

La figura 3 ejemplifica que un componente empresarial típico puede ser invocado por la interfaz de usuario o por los procesos de usuario. Este a su vez puede invocar servicios Web XML a través de interfaces de servicio. También puede invocar a otros componentes e incluso invocar componentes lógicos de

acceso a datos para actualizar o recuperar información. Además, utilizar a agentes de servicios para comunicarse con otros servicios que le permitan la ejecución de reglas de negocio.

Los componentes empresariales aceptan y devuelven estructuras de datos simples o complejas (entidad empresarial).

4.3 Entidades Empresariales

Las entidades empresariales corresponden a conceptos propios de un proceso de negocio. Estos componentes permiten pasar estructuras de datos a través de las capas sin tener que escribir código en cada capa. Una estructura de datos que representa una entidad no tiene necesariamente correspondencia con los objetos de la base de datos, lo que permite abstraerse de estos formatos. Una estructura de datos que representa una entidad almacena valores de datos y los expone a través de sus atributos.

Es recomendable que la entidad provea un método UPDATE que propague los cambios a la base de datos y no que cada vez que cambie un estado se acceda al almacén de datos, las entidades no tienen acceso directo a la base, deben utilizar los componentes lógicos de acceso a datos para realizar sus tareas cada vez que se invocan sus métodos. Las entidades no deben inicializar ningún tipo de transacciones, ni utilizar APIs de acceso a datos. En resumen, las entidades empresariales son sólo una representación de datos con potencial comportamiento.

5. CAPA DE DATOS Y SUS COMPONENTES

La mayoría de las aplicaciones y servicios necesitan obtener y almacenar información, la capa de datos es la encargada de contener datos y de proveer el acceso adecuado a dichos datos.

La arquitectura de .Net provee, para su capa de datos, cuatro componentes [MIC, 2002]:

1. Componentes lógicos de acceso a datos.
2. Agentes de Servicios.
3. Orígenes de Datos.
4. Servicios.

Para implementar la capa de datos de cualquier aplicación se deben tomar decisiones acerca de tres temas esenciales:

- Almacén de datos que se va a utilizar (bases de datos relacionales, base de datos orientada a objetos, sistemas de archivos, etc.)
- El diseño de los componentes de acceso al almacén de datos.
- El formato de los datos que se utiliza entre los componentes y modelo de programación necesario.

5.1 Componentes Lógicos de Acceso a Datos

La lógica utilizada para obtener acceso a los datos desde un origen de datos proporciona los métodos necesarios para las actualizaciones y consultas de datos. Los componentes lógicos de acceso a datos son independientes del almacén de datos utilizado, estos componentes abstraen la semántica del almacén de datos y dan interfaz simple de programación para la recuperación y realización de operaciones con los datos. Estos componentes implementan un patrón de diseño que separa el procesamiento empresarial de la lógica de acceso a datos y provee métodos para realizar operaciones (*Create*, *Read*, *Update* y *Delete*) relacionadas con una entidad empresarial.

Cuando una aplicación tiene más de un componente lógico de acceso a datos, resulta útil contar con un componente genérico de acceso a datos que administre las conexiones a las bases de datos, ejecute comandos y almacene parámetros en caché. Este componente genérico de acceso centraliza el desarrollo de APIs de acceso a datos y la configuración de conexión a estos, lo que permite reducir código duplicado.

Las funcionalidades que presentan los componentes lógicos de acceso a datos son:

1. Asignar y transformar argumentos simples de entrada y salida.
2. Permiten acceder a los datos a través de un único origen.

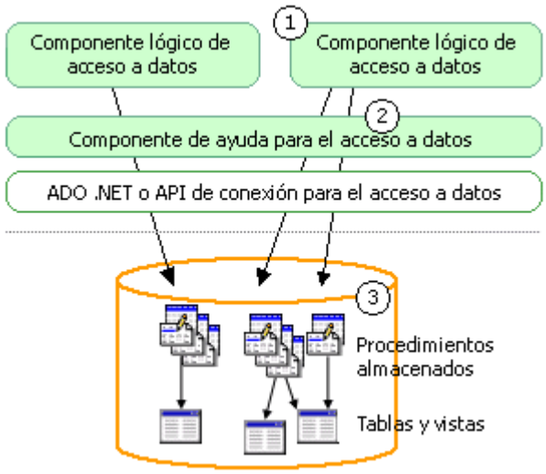


Figura 4: Componentes lógicos de acceso a datos.

La figura 4 ilustra lo siguiente:

- Los componentes lógicos de acceso a datos exponen métodos para insertar, eliminar, actualizar y recuperar datos, incluyendo la provisión de funcionalidad de paginación al recuperar grandes cantidades de datos.

- Se puede utilizar un componente de ayuda de acceso a datos[4] para centralizar la administración de la conexión y todo el código relacionado con el origen de datos específico.

- Se recomienda implementar las consultas y operaciones de datos como procedimientos almacenados (si es compatible con el origen de datos) para mejorar el rendimiento y la facilidad de mantenimiento.

[4] Microsoft proporciona Data Access Application Block para .Net, que se puede utilizar como un componente de ayuda de acceso a datos genéricos en la aplicación al utilizar bases de datos SQL Server.

5.2 Agentes de servicios

Los agentes de servicio se utilizan cuando en un proceso empresarial deben intervenir servicios externos y se hace necesario controlar la semántica de la comunicación con cada servicio que se utiliza. Se debe utilizar una API adecuada para llamar a cada servicio y realizar las transformaciones entre los formatos utilizados por el servicio externo y los utilizados por el proceso empresarial.

Los agentes de servicios son componentes lógicos de acceso a datos que proveen servicios externos o distintos a los almacenes de datos del proceso empresarial.

Las funcionalidades que presentan los agentes de servicios son:

- Encapsular el acceso a un servicio.
- Aislar la implementación de los procesos empresariales de los formatos de datos entre servicios externos y procesos empresariales.
- Dar formatos de datos de entrada y salida compatibles con los componentes empresariales que hacen uso de servicios externos.

También pueden ejecutar las siguientes tareas:

- Validar los datos intercambiados con el servicio externo.
- Almacenar en caché datos con los cuales se realizan consultas habituales.
- Autorizar acceso al servicio externo, autenticar y autorizar solicitudes.

- Definir el grado de seguridad adecuada (credenciales necesarias).
- Asegurar que las partes estén cifradas

Para el caso de comunicación asincrónica será necesario realizar el seguimiento de los mensajes intercambiados a través de datos empresariales.

6. NET COMPACT FRAMEWORK

.Net Compact Framework es un entorno de ejecución de aplicaciones para dispositivos con recursos de hardware reducidos, tales como asistentes de datos personales (PDA) como pocket PC, teléfonos móviles, decodificadores de televisión y otros que están integrados bajo el sistema operativo Windows CE .Net.

.NET Compact Framework ofrece las siguientes funciones principales:

- Ejecuta programas independientes del hardware y el sistema operativo.

- Admite protocolos de red comunes y se conecta perfectamente con servicios XML Web.

- Proporciona a los desarrolladores un modelo para orientar sus aplicaciones y componentes ya sea a una amplia gama de dispositivos o a una categoría específica de éstos.

- Facilita el diseño y la optimización de los recursos de sistema limitados.

- Obtiene un rendimiento óptimo en la generación de código nativo cuando se utiliza compilación Just-In-Time (JIT).

6.1 Arquitectura de .Net Compact Framework

.Net Compact Framework es un subconjunto de la biblioteca de clases .Net Framework y también contiene clases diseñadas expresamente para él. Hereda la arquitectura .Net Framework completa de Common Language Runtime y la ejecución de código administrado [NET, 2003].

La figura 5 ilustra la composición de la arquitectura de .Net Compact Framework.

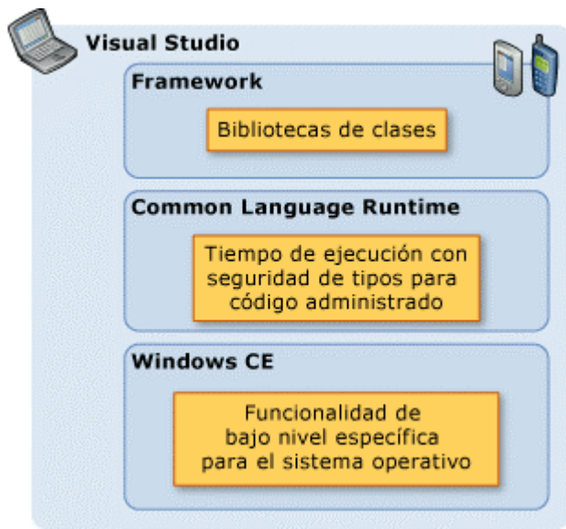


Figura 5: Esquema general de la Arquitectura .Net Compact Framework.

Windows CE. .Net Compact Framework utiliza el sistema operativo Windows CE para la funcionalidad central y para diversas características específicas de dispositivos. Se han adaptado varios tipos y ensamblados para que funcionen en forma eficaz bajo Windows CE, como los de los formularios Windows Forms, gráficos, dibujos y servicios Web.

.Net Compact Framework ofrece compatibilidad natural con Windows CE en los siguientes aspectos:

- Compatibilidad con seguridad nativa.
- Integración completa con programas de instalación nativos.
- Interoperabilidad con código nativo mediante la interoperabilidad COM y la invocación de plataformas.

Common Language Runtime. También se ha vuelto a generar para permitir que los recursos restringidos se ejecuten en memoria limitada y lograr un uso eficaz de la energía.

Framework. .NET Compact Framework es un subconjunto de .NET Framework.

Visual Studio. Ambiente de desarrollo de aplicaciones, incluye un conjunto de emuladores y tipos de proyecto que cubren el desarrollo para Pocket PC, Smartphone y Windows CE incrustado.

6.2 Dominios de aplicación de .Net Compact Framework

Las aplicaciones de .Net Compact Framework se ejecutan dentro de una construcción en tiempo de ejecución llamada dominio de aplicación, que es similar a un proceso del sistema operativo. .NET Compact Framework cerciora que todos los recursos administrados que utiliza una aplicación durante su ejecución sean liberados o se devueltos al sistema operativo cuando la aplicación finaliza.

Dentro de las ventajas que poseen los dominios de ampliaciones están: aislamiento de errores, mayor solidez y seguridad, sin necesidad de asistencia del sistema operativo.

6.3 Aspectos relacionados con la seguridad

A continuación se exponen algunos aspectos relacionados con la seguridad que deben ser considerados:

1. Actualmente las versiones 1.0 y 2.0 de .Net Compact Framework están concebidas bajo el estándar de plataforma abierta, por lo cual expone el código fuente completamente sin ninguna seguridad.

2. En la versión 1.0 de .NET Compact Framework, el motor en tiempo de ejecución no restringe el acceso de la invocación de plataforma a código nativo.

3. .NET Compact Framework no admite AllowPartiallyTrustedCallersAttribute, por lo cual, de forma predeterminada se concede un acceso sin restricciones a todas las bibliotecas.

4. Si el dispositivo acepta un código nativo que no es de confianza, no se puede garantizar una seguridad completa. La implementación de la seguridad sólo es válida para el código con seguridad de tipos comprobable. El límite entre el código administrado y no administrado es una amenaza grave para la seguridad del código administrado.

Microsoft tiene previsto incluir en versiones futuras directivas de seguridad para solucionar gran parte de los problemas mencionados.

CAPÍTULO III: PATRONES

1. INTRODUCCIÓN

En este capítulo se describe el concepto de patrón, para introducir al marco de patrones que propone Microsoft y finalmente se hace un recorrido por los patrones más representativos de arquitecturas orientadas a servicios (SOA).

2. DEFINICIÓN

Un patrón es una pareja problema/solución que se identifica por un nombre, contexto, y la descripción de la o las consecuencias de su utilización [STE,2003].

Solución al problema: en ciertas ocasiones un mismo problema real podría tener dos soluciones parecidas, correspondientes a dos patrones (Abstract Factory y Factory Method), pero la elección seguramente recaerá en el patrón que mejor se adapte al contexto específico del problema.

Las **consecuencias** son compromisos que se deben aceptar al adoptar un patrón. Expresa los beneficios, limitaciones, y problemas que conlleva el uso del patrón en la solución. Por ejemplo, un patrón como Flyweight soluciona un problema de espacio, a costa de una mayor complejidad en otras partes del desarrollo.

El **nombre e intención del patrón**: Se utiliza para describir un problema de diseño, su solución, y consecuencia en una o dos palabras. El uso de un nombre representativo ayuda al aumentar el uso de un vocabulario común entre profesionales del software.

Problema y contexto: Describe el problema de diseño específico, como algoritmos o comunicación entre objetos, el contexto se refiere a las circunstancias en las cuales es recomendable la utilización del patrón.

La **solución**: Provee una descripción abstracta de un problema de diseño y como deben disponerse en forma general los elementos (clases y objetos) para solucionar el problema.

3. MARCO GENERAL DE PATRONES MICROSOFT

Dada la importancia de la utilización de patrones, Microsoft plantea su marco de patrones, el marco de patrones es una matriz compuesta por niveles de abstracción versus puntos de vista de una aplicación, la figura 6 corresponde a lo mencionado:

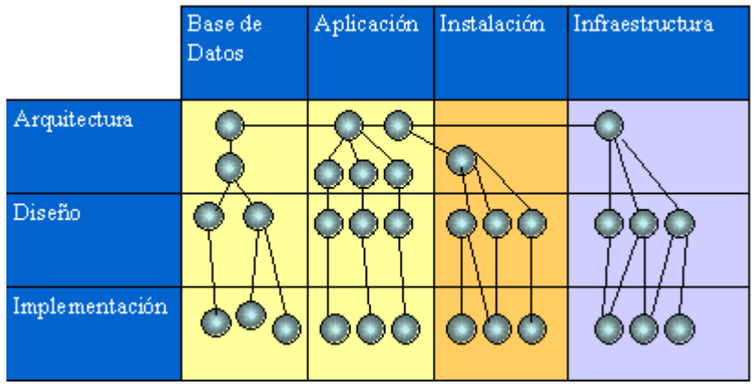


Figura 6: Marco de patrones de Microsoft

En la figura 6, las filas representan niveles progresivos de abstracción, estos niveles son:

- Arquitectura
- Diseño
- Implementación

Realizar clasificaciones por niveles de abstracción es útil, pues permite categorizar los patrones y ubicar fácilmente el área de interés que cubre cada grupo de patrones, además ayuda a los diversos integrantes de un equipo de desarrollo a comprender por dónde comenzar a trabajar cuando se diseña la arquitectura de la aplicación.

Cada columna de la figura 6 representa un punto de vista o perspectivas de una misma solución. Aunque los niveles de abstracción ayudan a profesionales de los diferentes grupos, no reflejan el efecto de la solución de software mucho más que componentes de código.

- La vista de base de datos describe la capa de persistencia de una aplicación, esta vista abarca cosas tales como esquemas físicos, tablas de bases de datos, relaciones y transacciones.
- La vista de aplicación se enfoca en los aspectos ejecutables de la solución, aquí se incluyen modelos de dominio, diagramas de clases, procesos y ensamblados.
- La vista de despliegue describe la ubicación de los componentes de la aplicación de acuerdo a la infraestructura sobre la cual deben ser instalados.
- La vista de infraestructura incorpora todo el equipamiento de hardware y de redes requerido para la ejecución de la solución.

El marco de patrones de Microsoft abarca patrones que van desde lo más general a lo más específico, por lo cual, está claro cuál podría ser el enfoque de un arquitecto ó el de un diseñador.

Siguiendo el marco de patrones Microsoft, existen entonces otros tipos de patrones, aparte de los de diseño descritos por GoF[5], de arquitectura y de implementación.

[5] Gang of Four es el nombre con el que se conoce a los autores del libro “Design Patterns”, Gamma, Helm, Johnson, Vlissides.

A continuación se introducen los tres tipos de patrones clasificados en marco de patrones de Microsoft, según niveles de abstracción de código, estos son los patrones de arquitectura, de diseño y de implementación.

3.1 Patrones de Arquitectura

Un patrón de arquitectura es: "Un patrón que expresa un esquema de organización estructural fundamental para los subsistemas o componentes de un sistema de software o las relaciones entre ellos. Provee un conjunto predefinido de subsistemas, especifica sus responsabilidades, e incluyen reglas y guías para organizar las relaciones entre ellos" [BUS, 1996].

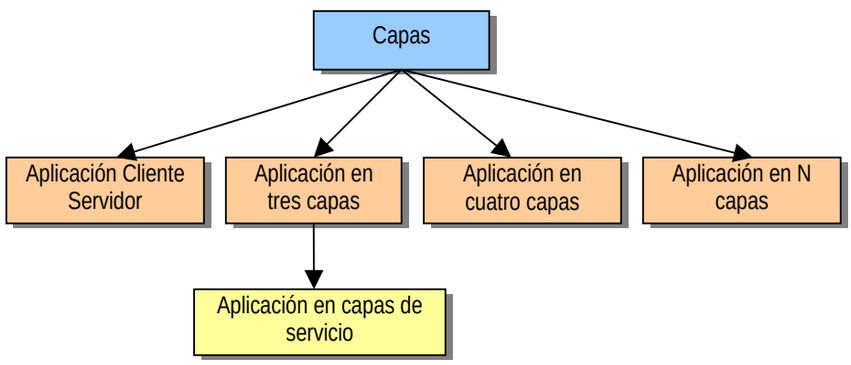


Figura 7: Cluster de patrones de arquitectura.

Las variaciones de patrones de arquitectura, básicamente, corresponden a las decisiones de divisiones por capas. Este cluster[6] de patrones, de arquitectura, es una agrupación lógica de un conjunto de patrones similares.

[6] Un cluster de patrones es una agrupación de patrones que se relacionan a con un tema específico.

3.2 Patrones de Diseño

GOF clasifica los patrones de diseño en tres grupos principales:

1. Patrones de creación
2. Patrones de comportamiento
3. Patrones de estructura

		De Creación	Estructurales	De Comportamiento
Ambito	**Clase**	Factory Method	Adapter (de clase)	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Adapter (de objetos) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Tabla 1: Clasificación de patrones de Diseño según GOF.

Los patrones de creación abarcan cómo gestionar la creación de instancias de las clases de un sistema, sea cual sea la complejidad de la solución en construcción, siempre se necesitará crear objetos. Los patrones de creación permiten crear objetos sin tener que identificar una clase específica en el código, algunos facilitan la creación de objetos y por los cual no se deben escribir grandes ó complejas extensiones de código, incluso otros fuerzan restricciones del tipo o número de objetos que pueden ser creados en el sistema. Algunos ejemplos clásicos de este tipo de patrones son:

- Fábrica abstracta
- Constructor

- Prototipo
- Singleton

Los patrones de comportamiento tienen que ver con el flujo de control en un sistema. Ciertas formas de organizar el flujo de control en un sistema pueden permitir mayor eficiencia y mayores niveles de mantenimiento del mismo. Algunos ejemplos clásicos de esta clasificación son:

- Observador
- Comando
- Estrategia

Los patrones estructurales indican como fragmentar y combinar los elementos de una aplicación. Dentro de este grupo se encuentran:

- Proxy
- Decorador
- Fachada

3.3 Patrones de Implementación

Un patrón de implementación esta en un nivel más detallado, son patrones específicos para un lenguaje de programación, se denominan también *idioms*. Es un patrón de bajo nivel especificado para una plataforma en particular y describe cómo implementar aspectos en particular de componentes o las relaciones entre ellos utilizando características de una plataforma dada.

Los patrones de implementación muestran cómo implementar conceptos de diseño utilizando el framework .Net. En algunos casos el framework ya incorpora algunas de estas implementaciones

4. CLUSTER DE PATRONES PARA SOLUCIONES EMPRESARIALES

Los cluster de patrones son agrupaciones de patrones que se relacionan a un tema específico.

Existen cinco cluster de patrones identificados y que se presentan en la tabla 2:

Cluster	Problema que resuelve
Presentación Web	Cómo crear aplicaciones Web dinámicas
Despliegue (Deployment)	Cómo dividir una aplicación en capas y luego distribuirla en una infraestructura de hardware multicapa.
Sistema distribuido	Cómo comunicarse con objetos que residen distintos procesos o diferentes computadores.
Rendimiento y confiabilidad (Performance and Reliability)	Cómo crear la infraestructura de un sistema que pueda resolver requerimientos operacionales críticos.
Servicios	Cómo acceder servicios que son proporcionados por otras aplicaciones. Cómo exponer funcionalidades como servicio para otras aplicaciones.

Tabla 2: Clasificación de Cluster de Patrones.

5. ARQUITECTURA BASADA EN SERVICIOS

Una arquitectura basada en servicios aplica el concepto de servicio para aplicaciones empresariales distribuidas, este tipo de aplicaciones exponen funciones de negocio de alto nivel para ser consumido por otras aplicaciones.

Una arquitectura orientada a servicios debe proveer funciones adicionales que permita invocar un servicio remoto. Las funciones más importantes son:

- **Hacer servicios localizables en tiempo de ejecución**: un servicio empresarial puede estar distribuido a través de muchos computadores, redes o instalaciones, la localización de los servicios puede variar en el tiempo.
- **Consumidores y proveedores deben comunicarse a través de un lenguaje común**: una vez localizado el servicio deseado, la aplicación consumidora del servicio debe ser capaz de determinar dinámicamente que protocolo utilizar para acceder al servicio, cómo debe ser el formato de la petición y conocer el tipo de respuesta que espera. Los servicios pueden ser implementados en una variedad de lenguajes y plataforma, ambos, proveedor y consumidor deben tener un acuerdo acerca del formato de intercambio de mensajes entre ellos.

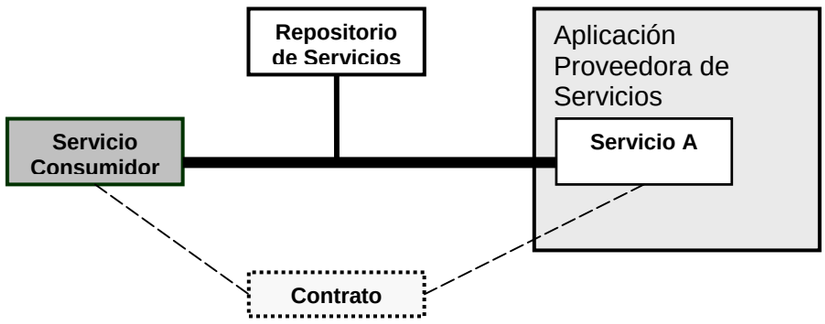


Figura 8: Diagrama de invocación de un servicio en Arquitecturas Orientadas a Servicios.

5.1 Contrato

Cuando un método llama a otro dentro de una aplicación, el método "signature" define un entendimiento entre el servicio proveedor y el que lo llama, por ejemplo el número y el tipo de parámetros pasados y retornados desde el método.

El contrato debe especificar la implementación del canal de comunicación que conecta el servicio consumidor con el servicio proveedor, tal como un protocolo de red. El contrato debe también especificar qué tipos de mensajes puede consumir o producir el servicio, describe el significado del esquema para cada mensaje involucrado en la interacción.

Un servicio puede necesitar dar apoyo a varios contratos, por ejemplo, un servicio consumidor que pertenece a la misma organización debe intercambiar información a través de mensajería de grano fino, un consumidor externo necesita intercambiar mensajes de grano grueso con el servicio por un tema de rendimiento.

Para invocar un servicio remoto se deben seguir los siguientes pasos:

1. **Descubrir**: el servicio consumidor consulta al repositorio de servicios, y el repositorio provee la localización del servicio deseado.
2. **Negociar**: el servicio consumidor y servicio proveedor están de acuerdo en un formato de comunicación especificado en el contrato.
3. **Invocar**: el servicio consumidor invoca al servicio proveedor.

5.2 Servicios Web

Los servicios Web proveen una implementación basada en estándares para una arquitectura orientada a servicios, definen una suite de tecnologías y protocolos que garantizan la simplicidad de soluciones basadas en un conjunto de aplicaciones colaborativas.

La definición de la W3C [7]dice lo siguiente respecto a los servicios Web:

> *"Es un sistema software identificado por medio de una URI[8], que publica interfaces que son definidas y descritas usando XML. Esta definición de interfaz puede ser descubierta por otros sistemas software. Estos sistemas pueden interactuar con el servicio Web de la manera prescrita en la definición, usando XML, utilizando mensajes cubiertos por los protocolos de Internet*."

[7] W3C, World Wide Web Consortium, organización que produce estándares para la World Wide Web
[8] URI, Uniform Resource Identifier, identificador uniforme de recursos

Los servicios Web poseen dos características claves:

1. Contrato de comunicación entre servicio proveedor y servicio consumidor.
2. Interoperabilidad, lo cual permite que sistemas de distinta naturaleza puedan comunicarse.

Los Servicios Web necesitan disponer de dos aspectos para la comunicación:

1. Canal de comunicación común.
2. Representación de datos y esquemas de mensajes.

5.3 Patrones de Servicio

Dentro del cluster de patrones de servicio se puede encontrar [ENT, 2003]:

- Service Interface
- Service Gateway

A continuación se detalla el contexto, problema, solución, beneficios y compromisos que expone la utilización de cada uno de ellos.

5.3.1 Service Interface

Contexto. Cuando se diseña una aplicación empresarial y se necesita que algunas de sus funcionalidades estén disponibles a través de una red, necesita ser funcionalmente accesible a varios tipos de sistemas, por lo cual la interoperabilidad es un aspecto clave en el diseño. Además de interoperabilidad, se necesita dar soporte a distintos tipos de protocolos de comunicación y que se acomode a diversos requerimientos operacionales.

Problema. Cómo hacer que las funcionalidades de la aplicación estén disponibles para otras aplicaciones mientras los mecanismos de interfaces están desacoplados de la lógica de aplicación.

Algunas ideas al respecto:

- Es deseable separar los elementos responsables de la lógica del negocio de la aplicación de los elementos responsables de protocolos de comunicación, de transformación de datos y cumplimiento de contratos.

- Los consumidores de la aplicación pueden necesitar respuestas optimizadas para escenarios de uso particular. Por ejemplo, algunos necesitan respuestas optimizadas para despliegue directo a usuarios mientras otros necesitan respuestas optimizadas de procesamiento de software.
- Los consumidores de la aplicación necesitan comunicarse con la aplicación utilizando tecnologías diferentes. Por ejemplo, los servicios consumidores externos a la compañía necesitan acceder a la aplicación a través de SOAP sobre Internet, mientras que los que están dentro de la compañía quieren acceder a través de Remoting.Net[9].
- La aplicación proveedora es la que debe imponer los requerimientos operacionales para los diferentes consumidores. Se puede tener un requerimiento de seguridad que autorice a los consumidores de una misma compañía para realizar operaciones de eliminación y de actualización, mientras que los consumidores externos a la compañía están autorizados para realizar operaciones de sólo lectura.
- La capacidad de la aplicación para responder a cambios en el ambiente de negocio de manera oportuna está influenciada por el hecho de que los cambios de la lógica de negocio están aislados de los mecanismos utilizados por consumidores que interactúan con la aplicación.

Solución. Diseñar la aplicación como un conjunto de servicios de software, cada uno con una interfaz a través de las cuales los consumidores de la aplicación deben interactuar con el servicio.

[9] Proceso por el cual aplicaciones o componentes interactúan a través de ciertos límites, estos límites podrían ser distintos procesos o máquinas. Remoting provee un modelo distribuido de objetos que permite la invocación remota de métodos entre diferentes entornos Common Language Runtime (CLR) por toda la red o entre diferentes dominios AppDomains del mismo CLR.

Un servicio de software es una unidad discreta de lógica de aplicación que expone una interfaz basada en mensajes que esta disponible para ser accedida por otras aplicaciones. Cada servicio software tiene asociada una interfaz que presenta a los consumidores. Esta interfaz define e implementa un contrato entre los consumidores de los servicios y el proveedor del servicio. El contrato y su implementación asociada están referidos a la interfaz de servicio.

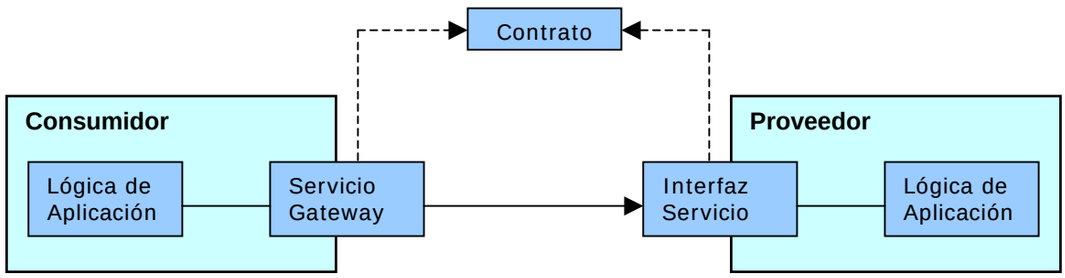


Figura 9: Elementos de un servicio.

La figura 9 muestra que Interfaz de servicio provee un punto de entrada que los consumidores utilizan para acceder a la funcionalidad expuesta por la aplicación. La interfaz de servicio generalmente tiene dirección de red, significa que es capaz de ser accedida por el consumidor sobre una cierta clase de red de comunicaciones. La dirección de red puede ser conocida o se puede obtener desde un directorio de servicio como UDDI[10].

[10] UDDI (Universal Description, Discovery and Integration) es una especificación para publicar y buscar información acerca de servicios Web disponibles. Los servicios UDDI son un servicio Web XML basado en estándares que permiten publicar, descubrir, compartir y reutilizar servicios Web directamente, a través de herramientas de desarrollo y aplicaciones empresariales dado que estos servicios UDDI comprenden una solución Basados en Microsoft .NET Framework.

Un aspecto clave del diseño de una interfaz de servicio es desacoplar la implementación necesaria para comunicarse con otros sistemas de la lógica de negocio de la aplicación. También provee una barrera que habilita a la lógica de aplicación cambiar sin afectar a los consumidores del servicio.

La interfaz de servicio implementa el contrato entre el consumidor y proveedor. El contrato permite intercambiar información aún cuando sean sistemas diferentes. La interfaz de servicio es responsable de todos los detalles de implementación necesarios para ejecutar la comunicación, entre los cuales está protocolo de red, formato de datos seguridad y acuerdos del nivel de servicio.

Beneficios. Los mecanismos de interfaz de servicio están desacoplados de la lógica de la aplicación. Esta separación permite agregar fácilmente nuevas interfaces y cambiar la implementación de la lógica de aplicación con mínimo de impacto en los consumidores.

Desacoplar el código de la interfaz del servicio del código de implementación del servicio permite el despliegue de dos bases de código en capas separadas, aumentando potencialmente la flexibilidad de despliegue de la solución.

Compromisos. Muchas plataformas exponen funcionalidades simples de la aplicación, sin embargo, esto puede conducir a una solución pobre en términos de granularidad. Si la interfaz es de demasiado grano fino, se terminará haciendo muchas llamadas al servicio para una acción específica. Se necesita diseñar interfaces de servicio que sean apropiadas para la red o comunicación fuera de proceso.

Se debe tener presente que cada interfaz de servicio adicional que provee un servicio aumenta la cantidad de trabajo requerido para hacer un cambio en la funcionalidad expuesta por el servicio.

5.3.2 Service Gateway

Contexto. Cuando se diseña una aplicación empresarial que consume un servicio que es proveído por otra aplicación, el servicio proveedor define un contrato que todos los consumidores deben respetar para acceder a dicho servicio. Este contrato define cosas tales como la tecnología, protocolos de comunicación y definiciones de mensajes necesarias para comunicarse con el servicio. Por último, para comunicarse con el servicio, la aplicación debe cumplir con las responsabilidades que establece el contrato.

Problema. ¿Cómo desacoplar del resto de la aplicación los detalles implícitos en la responsabilidad del cumplimiento del contrato definidas por el servicio?.

Cuando se diseña una aplicación que consume servicios proveídos por otras aplicaciones se deben tener presente las siguientes ideas:

- Implementar responsabilidades de contrato para el consumidor, implementar seguridad y mecanismos de comunicación tales como autenticación, marshaling[11], encriptación y enrutamiento de mensajes.
- El contrato puede especificar formato de datos que son diferentes de la representación interna de la aplicación, es más, los datos deben ser

[11] Marshaling es la técnica que permite la transparencia, respecto de la ubicación física de un componente servidor. Permite que los objetos se puedan utilizar en hilos, procesos y redes con independencia de la ubicación.

traducidos. A veces estas traducciones son simples renombres de campos o conversiones de tipos de datos, pero en otras ocasiones estas conversiones involucran transformaciones estructurales complejas y transformaciones semánticas.

- No es posible controlar el contrato especificado por un proveedor de servicio. Si el contrato cambia, es deseable minimizar el impacto en el código de la aplicación que utiliza el servicio.
- Los canales de comunicación que proveen conectividad entre la aplicación y los servicios, por lo general, proveen una API[12] genérica de bajo nivel a la aplicación. Esta API puede incluir funciones genéricas tales como SendData. En muchas situaciones, es deseable que la aplicación tenga una interfaz semánticamente más rica a través de métodos tales como ValidarTarjetaDeCredito o ObtenerDireccionCliente.

- Algunos contratos pueden especificar mensajería asíncrona, es decir, no pueden retornar un resultado inmediatamente, en lugar de eso, el servicio consumidor debe estar preparado para recibir mensaje resultante asíncrono desde el servicio proveedor.

Solución. Encapsular el código que implementa la porción consumidora del contrato dentro de su propio componente Service Gateway. Los Service Gateway juegan un rol similar al de los componentes de acceso a datos cuando acceden a la base de datos de la aplicación, actúan como proxies para otros servicios, encapsulan los detalles de conectividad a la fuente y ejecutan algunas traducciones necesarias.

[12] Application Programming Interface, Interfaz de Programación de Aplicaciones.

Service Gateway es una adaptación del patrón *Gateway* de Martin Fowler para arquitecturas orientadas a servicios, y como tal, su principal responsabilidad es encapsular el acceso a sistemas externos de la aplicación consumidora. Service Gateway interactúa con *Remote Facade* en lugar de interactuar con un sistema externo directamente. *Remote Facade* encapsula funcionalidades complejas en la aplicación proveedora y expone funcionalidades mediante interfaces simples para las aplicaciones consumidoras. Por otro lado, Service Interface es un tipo específico de *Remote Facade* adaptado para arquitecturas orientadas a servicios. En arquitecturas orientadas a servicios es común para un Gateway Service de una aplicación consumidora colaborar con Service Interface expuestas por una aplicación proveedora de servicios. La figura 10 muestra estas relaciones.

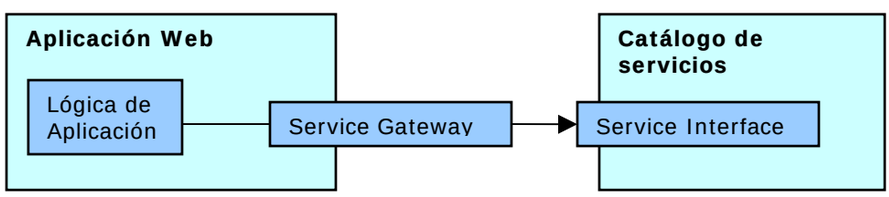


Figura 10: Relaciones Service Gateway.

El componente Service Gateway encapsula los detalles de bajo nivel de la comunicación con un servicio. Tales detalles incluyen pero no se limitan a canal de comunicaciones, formato de datos, semántica de las llamadas sincrónicas versus asíncronas, adaptadores de procesos.

Beneficios. Desacoplar la lógica de acceso al servicio del resto de la aplicación hace más fácil cambiar el servicio de los accesos a la aplicación. Por ejemplo, se puede querer cambiar a una nueva versión del mismo servicio o se puede querer utilizar un servicio con mejores garantías de otro fabricante. El cambio a

otro servicio es mucho mas fácil si automáticamente se genera código que realiza un mapeo de los datos.

Service Gateway oculta las complejidades del acceso al servicio desde la aplicación. Esto mejora reutilización de los componentes de aplicación y los componentes de accesos a servicios. La aplicación no hace referencia directa al servicio, es independiente de detalles de implementación y de localización del servicio. Encapsular la lógica de acceso al servicio en capas separadas también mejora la reutilización de lógica de acceso, pues puede ahora ser utilizado a través de múltiples servicios llamados como el mismo transporte y mecanismo de autenticación utilizado.

Service Gateway provee una localización ideal para proveer características comunes tales como invocación asíncrona, caching[13], y manejo de errores.

Compromisos. Service Gateway agrega un nivel de complejidad que podría ser innecesario para soluciones simples. En particular, el esfuerzo e infraestructura necesitada para soportar la generación automática de mapeo de componentes puede no necesitarse si la aplicación únicamente utilizará pocos servicios estáticos.

Un Service Gateway en particular es responsable de interactuar con un solo servicio. La coordinación entre múltiples servicios debe ser manejada por un componente adicional.

[13] Técnica que permite mejorar el tiempo de respuesta. Consiste en almacenar información frecuentemente utilizada en una memoria de acceso rápido.

El Service Gateway está contenido en una única aplicación, sin embargo, la duplicación de código podría darse si múltiples aplicaciones acceden al mismo servicio y ambas duplican la funcionalidad del Gateway. Desarrollar un Service Gateway reutilizable puede ser una alternativa.

CAPÍTULO IV: PER-MÓVIL,
UN CASO REAL DE SOLUCIÓN MÓVIL

1. INTRODUCCIÓN

El siguiente capítulo expone un caso real de implementación de solución móvil para optimizar el proceso de comercialización de servicios de una empresa de la industria financiera chilena. A continuación se analizará y describirá la problemática que dio origen a esta inversión tecnológica, la experiencia obtenida y detalles de la solución, con el objetivo de extrapolar una arquitectura de software genérica y poder dar bases fundamentadas de diseño a problemáticas similares.

2. QUÉ CONTIENE ESTE CAPÍTULO

Este capítulo muestra el análisis de componentes y de capas de la solución en estudio, PER-Móvil. El capítulo se inicia describiendo los antecedentes generales del dominio del problema, esto ayuda a inferir cómo se generó la necesidad y a partir de ella se esbozan las principales características que la solución debe implementar. Finalmente, se verá que dichas características son fundamentales y completamente restrictivas en el esquema de la arquitectura de diseño de la solución de software.

Importante:

Para proteger la confidencialidad de información y para cumplir con las normas de la compañía propietaria de la solución móvil que se estudia en este trabajo de titulación, en adelante nos referiremos a ella como “Organización B”. Luego, para referirse a los empleados de la organización se utilizará “Representantes de B”.

3. ORIGEN DE LA NECESIDAD

Todas las soluciones de software se basan en necesidades de un modelo de negocio. Se entiende como modelo de negocio a todos los procesos, datos, actividades o tareas, roles y reglas o políticas que una empresa utiliza para conseguir sus objetivos. Cada proceso se caracteriza por una colección de datos que son producidos y manipulados mediante un conjunto de tareas, en las que ciertos agentes (por ejemplo, trabajadores o departamentos) participan de acuerdo a un flujo de trabajo determinado. Además, estos procesos se hallan sujetos a un conjunto de reglas de negocio, que determinan la estructura de la información y las políticas de la empresa.

Para el caso particular de la iniciativa del proyecto en estudio, el modelo se identifica como el negocio de préstamos de dinero que otorga la entidad financiera a sus clientes o potenciales clientes. Dentro de este modelo en particular existen procesos de comercialización para los distintos productos. Dado lo anterior, las necesidades de este negocio se pueden clasificar en dos grupos:

1. Necesidades generadas por el entorno del mercado.
2. Necesidades internas de la organización.

3.1 Necesidades generadas por el entorno del mercado

Este proyecto se lleva a cabo en una importante entidad bancaria en la ciudad de Santiago. La baja en las tasa de interés del mercado durante los años 2004 y 2005, produjo que la oferta de la industria crediticia crezca exponencialmente. Lo anterior produce un cambio en la forma de comercialización de productos de financiamiento, la forma de vender cambia desde modalidad receptiva en sucursales a una más agresiva en el sitio de los potenciales clientes.

La entidad bancaria inversionista de la solución en estudio, decide que sus representantes deben salir a visitar a sus clientes y dar respuesta ágil y oportuna acerca de los trámites de evaluación y solicitud de los productos. Un factor importante para llevar a cabo esta estrategia, es que los representantes cuenten con las herramientas idóneas para la gestión de venta diaria.

3.2 Necesidades Internas de la organización

La fuerza de venta de organización B debe estar en los lugares de trabajo de los clientes debido a que se debe certificar que los futuros clientes cuenten con un determinado patrimonio.

Los representantes de B deben portar todos los antecedentes de los productos que pueden ofrecer y con información de la oferta disponible, además de formularios preimpresos en los cuales deben ingresar datos del cliente.

Una vez que el representante de B logra pactar la venta de un producto con determinadas condiciones, debe tomar nota acerca de los datos recopilados en terreno y concurrir a las dependencias de organización B a registrar dichas ventas en los sistemas Legacy.

Un factor importante a considerar dentro del proceso de venta, es que los representantes de B deben concurrir a diario a los lugares donde se encuentran los potenciales clientes. En muchas ocasiones se requiere más de un viaje, llamadas telefónicas y envío/recepción de información vía fax para cerrar la venta.

4. DESCRIPCIÓN DE LA NECESIDAD

La organización B requiere que sus representantes cuenten con herramientas de software que provean las mismas funcionalidades que proveen los sistemas Legacy actuales, al momento de comercializar sus productos en terreno. Esta herramienta debe permitir que representantes de B puedan consultar e ingresar información de manera expedita, con tiempos de respuesta aceptables y con un nivel de seguridad acorde a la magnitud de las exigencias del mercado y de la competencia.

Las operaciones efectuadas por este medio deben estar alineadas con las políticas de comercialización vigentes de la compañía, para lo cual no deberá ser necesario un proceso adicional para el cierre de la venta. Se espera lograr que el cliente perciba una atención más personalizada y que sus requerimientos sean atendidos de forma eficiente, dado que la evaluación comercial será realizada en línea e informada al instante.

5. CARACTERÍSTICAS DE LA SOLUCIÓN

Dada la necesidad de la organización B, se puede inferir que lo que ésta precisa es una solución que permita a sus empleados o representantes contar con información en cualquier lugar y en cualquier momento. Esta necesidad sólo puede ser abordada por una solución de software de tipo móvil.

El sistema de evaluación que utiliza actualmente la organización B para evaluar la situación social y económica de sus clientes, es conocido como PER (Proceso de Evaluación de Riesgo). PER apoya la toma de decisiones por parte de los representantes de B. Este sistema opera actualmente en las estaciones de trabajo en la red LAN de las dependencias de la organización B como una aplicación de escritorio. Esta aplicación es utilizada por toda la fuerza de venta de la empresa y posee toda la lógica requerida para el trabajo diario. Es por esto que la solución debe incorporar al menos las funcionalidades con que cuenta el actual sistema PER.

Para proceder con la solución móvil, se evalúan los distintos dispositivos que podrían proveer el soporte de este tipo de aplicaciones, sin embargo, para comenzar, la empresa descarta un computador portátil como herramienta de trabajo para sus representantes, ya que los costos y los riesgos de daño o pérdida son muy elevados. Se optó por dispositivos de menor tamaño que les brinde comodidad y facilidad de uso en el trabajo diario. La solución debe comprender la implementación de una aplicación móvil para dispositivos PDA.

6. OBJETIVO DE LA SOLUCIÓN MÓVIL EN ESTUDIO

El objetivo de la solución PER-Móvil, que se utilizará como caso de estudio, es brindar a los representantes de B las mismas funcionalidades del sistema PER en el momento y lugar donde se comercializan los productos que ofrece organización B.

7. CONTEXTO DE LA SOLUCIÓN EN ESTUDIO

El siguiente diagrama muestra el contexto de la aplicación que se utilizará para contrastar la hipótesis presentada en este trabajo de título.

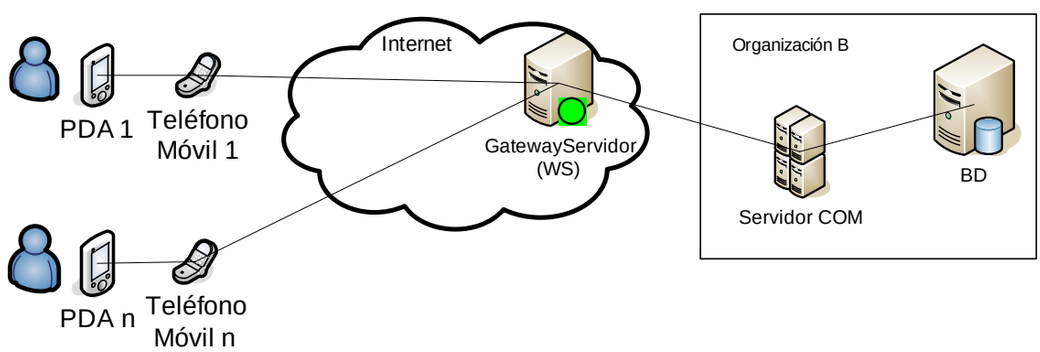


Figura 11: Contexto general solución PER-Móvil.

En la figura 11 se puede observar lo siguiente:

1. Cada representante de B porta un dispositivo inteligente Pocket PC, una PDA, que le permite conectarse a las oficinas centrales.
2. La conexión a Internet la provee un teléfono celular.
3. El servicio Web que expone la aplicación proveedora es llamado GatewayServidor, este es el encargado de atender las peticiones y entregar respuestas a los consumidores.

4. El servidor COM hospeda los servicios que encapsulan gran parte de la lógica de negocio de la aplicación PER. Estos servicios serán reutilizados por PER-Móvil.

8. PER-MÓVIL IMPLEMENTADO SOBRE UNA ARQUITECTURA ORIENTADA A SERVICIOS

Dada la necesidad de organización B, nace la característica fundamental del diseño de PER-Móvil, dar las mismas ventajas para los representantes de B que están ubicados en las dependencias que a los que trabajan en terreno, una característica de esta naturaleza constituye la principal restricción de diseño tecnológico y a la vez una gran ventaja competitiva desde el punto de vista comercial. Para cumplir con este requerimiento se plantea que PER-Móvil debe constituir una solución orientada a servicios.

Como se menciona en el capítulo 3 de este trabajo de titulación, las Arquitecturas Orientadas a Servicios (SOA) se basan en dos patrones fundamentales: "*Service Interface*" y "*Service Gateway*".

La figura 12 ilustra la arquitectura SOA para PER-Móvil, en ella se pueden identificar los dos patrones mencionados anteriormente.

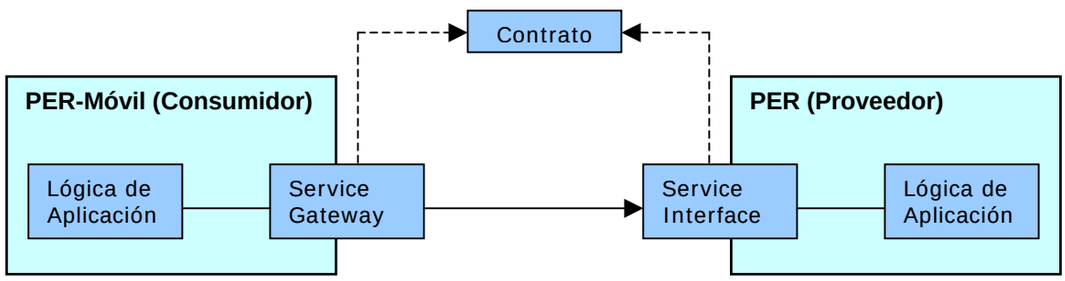


Figura 12: PER-Móvil, una solución orientada a servicios.

El **consumidor** está dado por la aplicación instalada en el cliente móvil, a través de este cliente se realizan todas las acciones de petición y de

recuperación de información proveniente de los sistemas Legacy de la organización B.

El **Service Gateway,** que expone la figura 12, corresponde a la componente llamada GatewayCliente en PER-Móvil, GatewayCliente es la responsable de proveer la información dentro de la aplicación, esto es, la lógica propia del cliente móvil no ve las complejidades de solicitar y recibir información. Esto hace que el recuperar y actualizar información sea transparente dentro de la aplicación móvil cliente.

El componente **Service Interface** está dado por GatewayServidor en PER-Móvil, esta interfaz de servicio posee el catálogo de operaciones disponibles, además establece y conoce el **contrato** entre consumidor y proveedor pues fija los parámetros de entrada/salida y el tipo de cada uno de estos, en general, posee la descripción del intercambio de mensajes de entrada y salida desde y hacia el proveedor.

El **proveedor** expone un servicio Web que accede a la lógica de negocio implementada a través de los componentes COM[14]. De esta forma se logra reutilizar gran parte de la lógica existente.

[14] Es una estructura de programación que incluye tanto datos como funcionalidad. Es un componente cuyo acceso público es implementado a través de interfaces.

8.1 PER-Móvil es una solución basada en componentes

Un sistema de software esta basado en componentes. Por lo tanto, un componente es una parte lógica de un sistema que puede ser reemplazado. Un componente[15] debe tener interfaces de comunicación bien definidas para poder comunicarse y prestar servicios a otros componentes.

A continuación se realiza un análisis de las capas lógicas de la aplicación PER-Móvil y los componentes residentes en cada una de ellas:

8.1.1 Capa de Presentación y sus componentes en PER-Móvil

La creación de una interfaz de usuario móvil presenta sus propios retos, en general, la interfaz de usuario de un dispositivo móvil necesita ser capaz de mostrar la información en una pantalla de un tamaño considerablemente menor al de las aplicaciones habituales y debe ofrecer un nivel aceptable de uso para los dispositivos de destino. Se debe considerar diseñar las interfaces de usuario minimizando los requisitos de entrada de datos.

La capa de presentación de PER-Móvil está compuesta por formularios Windows CE de una smart device application, esta componente es la encargada de obtener y entregar datos al usuario, de restringir las entradas y garantizar el ingreso de datos obligatorios y en todo momento debe además mostrar el estado de la aplicación.

[15] Microsoft define un componente como un grupo de funciones relacionadas que implementan una característica en particular de un sistema. Un componente es subconjunto de un módulo.

8.1.1.1 Componentes de Interfaz de Usuario

PER-Móvil utiliza una interfaz de dispositivo inteligente Pocket PC que posee el sistema operativo Windows CE, este dispositivo permite desarrollar interfaces de usuario conectadas a través de tecnologías inalámbricas. En esta interfaz móvil se mantiene el control de las excepciones e informa al usuario cuando se produce un error, además permite realizar nuevo intento o simplemente cancelar la operación. Los formularios smart device application implementan está lógica propia de validación, del lado cliente, para garantizar entradas válidas.

La interfaz de usuario está compuesta por formularios de una smart device application. Estos formularios corresponden a implementaciones de clases públicas que heredan de la clase System.Windows.Forms.Form proporcionada por el Compact Framework .Net. Para poder realizar despliegue y captura de información, estos componentes deben crear instancias de componentes y entidades empresariales.

8.1.2 Capa Empresarial

En esta capa se implementa la lógica de negocio. La lógica de la capa empresarial de PER-Móvil está diseñada para ser utilizada directamente por los componentes de presentación.

La lógica empresarial de PER-Móvil debe realizar llamadas a agentes de servicios con el fin de recuperar información y desarrollar operaciones lógicas.

A continuación se identifican los componentes empresariales y entidades empresariales que han sido diseñadas e implementadas para PER-Móvil.

8.1.2.1 Componentes empresariales

Estas clases poseen métodos que se responsabilizan de invocar servicios de GatewayCliente, con el fin de resolver las solicitudes emitidas por los componentes de la capa de presentación. Estos componentes procesan las respuestas entregadas por GatewayCliente y entregan como resultado objetos que representan a las entidades empresariales.

8.1.2.2 Entidades empresariales

Las entidades empresariales corresponden a conceptos propios de un proceso de negocio y es posible implementarlos a través de un caché de información que tiene coherencia dentro de una transacción efectiva. Estos componentes de entidad permiten pasar estructuras de datos a través de las capas sin tener que escribir código en cada capa. En resumen, las entidades empresariales son sólo una representación de datos con comportamiento potencial. En el caso específico de PER-Móvil, las entidades están

implementadas a través de clases simples que no contienen lógica sino únicamente atributos privados y métodos públicos para acceder a ellos.

Las entidades empresariales básicas en PER-Móvil son:

- **Ejecutivo**: esta entidad corresponde al representante de B, algunos atributos representativos de esta entidad son: rut ejecutivo, nombre, oficina, categoría.
- **Cliente**: esta entidad representa al cliente de organización B, algunos atributos representativos de esta entidad son: rut, nombre, dirección, teléfonos.
- **Evaluación**: esta entidad representa a la parte medular de la aplicación, identificador, rut, rut ejecutivo, activos, pasivos.
- **Producto**: esta entidad representa a los productos que ofrece la organización B, identificador de producto, nombre, entre otros atributos propios de cada producto.

8.1.3 Capa de datos y sus componentes

PER-Móvil utiliza una clase pública llamada Trancode para acceder al servicio remoto que se ocupa de coordinar la comunicación con los componentes que acceden a los almacenes de datos. El almacén de datos que utiliza la solución en estudio es una base de datos relacional. Las tablas y vistas de datos residen en la aplicación proveedora.

Trancode implementa un método llamado ServicerCall(). Este método se responsabiliza de invocar al GatewayServidor, entregándole como argumento el nombre del componente (COM) que debe atenderlo y los parámetros. El método retorna un elemento XML como resultado que es procesado por los componentes de negocio.

```
Option Explicit On
Option Strict On
Public Class Trancode
Public Function ServiceCall(ByVal serviceName As String, ByVal args() As [Object])
As System.xml.XmlElement
        Dim cls As New cl.bdd.des.Trancode
        Dim xml As xml.XmlElement
        xml = CType(cls.ServiceCallBDD(serviceName, args), xml.XmlElement)
        Return xml
    End Function
…
End Class
```

GatewayServidor corresponde a un servicio Web que atiende las peticiones provenientes de GatewayCliente (clase Trancode). A través de los parámetros que recibe, se encarga de buscar en su configuración o catálogo a qué servicio lógico debe invocar, crea una instancia de dicho servicio lógico e invoca su ejecución de acuerdo a los parámetros establecidos, para finalmente emitir la respuesta en formato XML al consumidor GatewayCliente.

.

En el catálogo de servicios se especifican los nombre de los servicios lógicos, sus parámetros y la componente COM asociada. Su definición esta dada por una estructura XML.

La figura 13 muestra un esquema de la disposición de los componentes de la solución PER-Móvil. La interfaz de usuario, compuesta por formularios, corresponde a clases cuyos métodos resuelven acciones realizadas sobre la interfaz. Cuando el usuario realiza una acción sobre la interfaz, un método asociado al evento crea una o más instancias de las clases correspondientes a las entidades empresariales y componentes empresariales. Las componentes empresariales se comunican directamente con el GatewayCliente para responder en base a la lógica de negocio establecida. Cómo respuesta de este proceso las instancias de entidades creadas por los formularios son inicializados con valores para posteriormente desplegar su información.

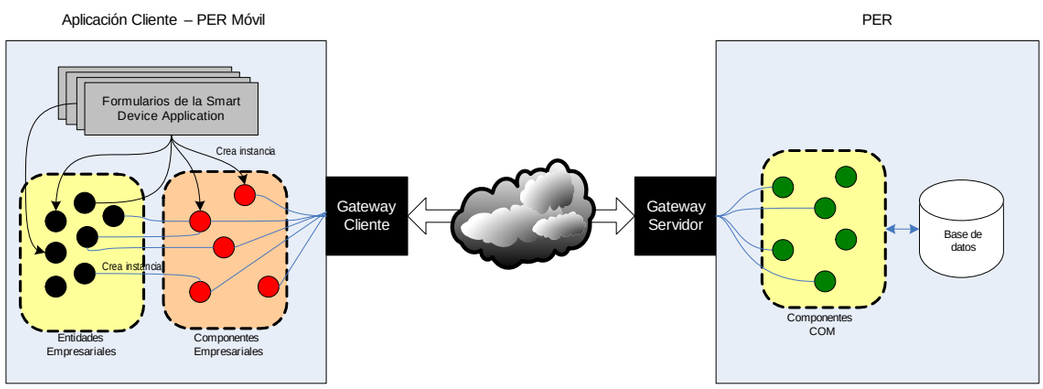


Figura 13: Vista de componentes de PER-Móvil.

9. CONCLUSIÓN DEL CAPÍTULO

La solución tecnológica PER- Móvil no fue concebida como una solución basada en componentes, ni tampoco bajo el paradigma arquitectónico SOA, más bien el equipo de desarrollo utilizó los conocimientos básicos relativos al lenguaje de programación con el fin de obtener una solución elemental funcional.

El análisis descrito anteriormente fue necesario para permitir identificar la arquitectura y composición de la solución. Queda de manifiesto, que aunque fue posible identificar algunos componentes lógicos y situarlos sobre capas que nos hacen pensar en una presunta arquitectura orientada a servicios enmarcada en el principio de arquitectura de aplicaciones .Net, esta solución carece de muchas de las de las ventajas y beneficios inherentes en estos dos paradigmas.

Sin embargo, se puede considerar el caso en estudio como una base mejorable y con gran potencial de convertirse en una aplicación móvil genérica.

CAPÍTULO V: UNA SOLUCIÓN MÓVIL GENÉRICA

1. INTRODUCCIÓN

En el presente capitulo se presenta la arquitectura propuesta en este trabajo de titulación. El enfoque utilizado para presentar la arquitectura será el mismo utilizado en el capitulo anterior para describir la arquitectura de PER-Móvil y que además es ampliamente utilizado por Microsoft para detallar la plataforma .Net.

En forma general la arquitectura se puede ilustrar mediante la figura 14, en donde se exponen las diferentes capas de la aplicación y sus principales componentes.

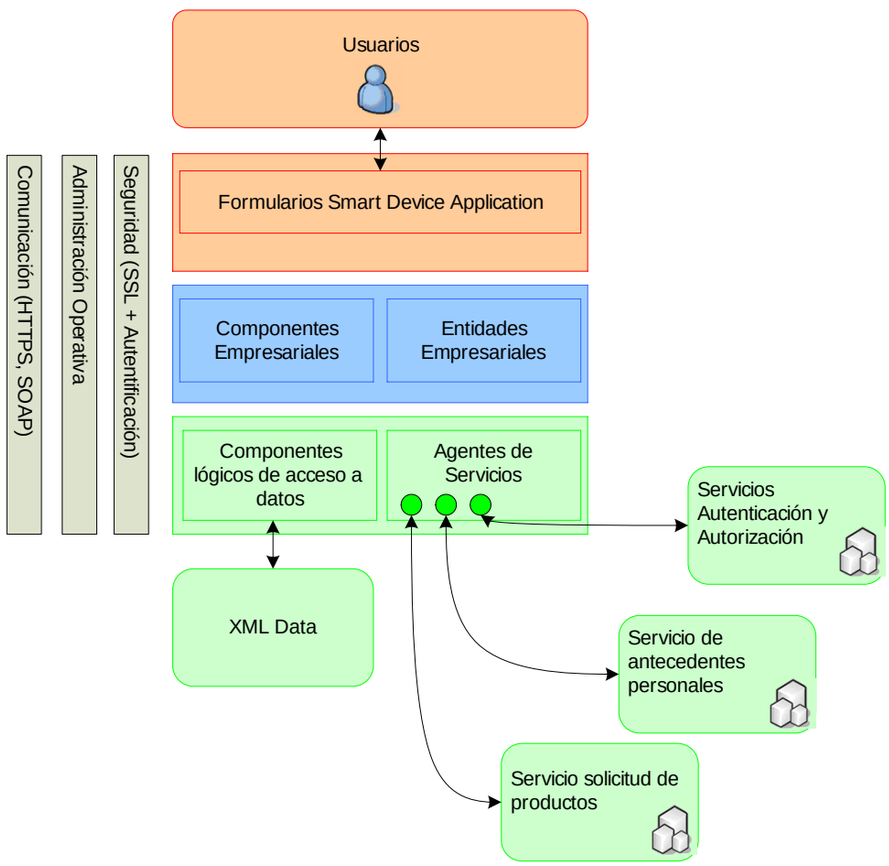


Figura 14: Esquema de arquitectura generica para aplicaciones móviles con reutilización de lógica de negocio.

La arquitectura que aquí se presenta se centra en mejorar aspectos tales como:

- Escalabilidad
- Implementación basada en componentes reutilizables.
- Alta disponibilidad
- Mantenibilidad
- Seguridad

Para verificar los aspectos anteriormente expuestos, se presentan los supuestos considerados para plantear el modelo, las condiciones sobre las cuales se basa el diseño de la arquitectura y se explica en detalle cada una de las capas y componentes que la componen. Además, para cada componente se revela cuál será el patrón más adecuada o la decisión de implementación que debe ser utilizada para trabajar en la arquitectura propuesta.

2. SUPUESTOS DEL MODELO

Es importante enfatizar que para generar la solución móvil propuesta en este trabajo de titulación se consideran algunos supuestos. Estos supuestos se asientan en la idea de que, en general, las organizaciones que pertenecen a la industria financiera chilena poseen su lógica de negocio implementada en sistemas de diferente naturaleza. En la actualidad existen 26 bancos establecidos y operando en el país, de los cuales 19 son bancos establecidos en Chile, además de 6 sucursales de bancos extranjeros, el banco estatal que corresponde al banco del Estado de Chile y finalmente Banco Central de Chile, lo cual nos hace pensar que la arquitectura propuesta podría potencialmente ser aplicable a cualquiera de estas instituciones.

Dentro de la lógica empresarial que poseen los sistemas legacy de este tipo de organizaciones, se pueden considerar elementos tales como: sistemas o componentes de registro de antecedentes personales y/o comerciales de sus clientes, procedimientos de autenticación, registros de sus productos o servicios disponibles, funciones de cálculos de precios para cada uno de los productos ofrecidos, funcionalidades de desarrollo de deuda de los distintos productos una vez vendidos, servicios de consulta de estado de deudas, componentes de actualización de antecedentes de los diversos ejercicios, interfaces de servicios que brindan sistemas externos a la organización, simuladores de créditos, etc.

Dado lo anterior, se han tomado los siguientes supuestos en la definición de la arquitectura:

1. La organización que desea invertir en una solución tecnológica móvil posee un almacén para la persistencia de la información y diferentes componentes tecnológicos que encapsulan la lógica del negocio. Estos componentes son por lo general:
    a. Componentes COM
    b. APIs
    c. Procedimientos almacenados
    d. Servicios Web

2. La aplicación móvil actuará como una interfaz remota para el usuario final, ocupándose de emular el flujo de trabajo o procedimientos con los que están familiarizados los usuarios, permitiendo el ingreso, procesamiento y presentación de resultados de forma óptima dado las limitaciones de interfaz provista para los dispositivos móviles. Lo anterior se logra reutilizando la lógica residente en los componentes de software existentes en los sistemas legacy.

Es importante enfatizar que el ámbito de la solución móvil propuesta se basa estrictamente en estos dos supuestos y no intentará ir más allá de ellos. Sin embargo se debe tener presente que la solución puede ser aplicada a organizaciones de otros rubros, cuyos sistemas cumplan con las condiciones anteriormente señaladas.

3. SEGURIDAD

La seguridad es un tema significativo en cualquier aplicación y toca en forma transversal a todas las capas lógicas y componentes de estas. Toda implementación que requiera aplicar ciertos niveles de seguridad debe considerar temas tales como: autenticación, autorización, comunicación segura, auditoria y administración de perfiles.

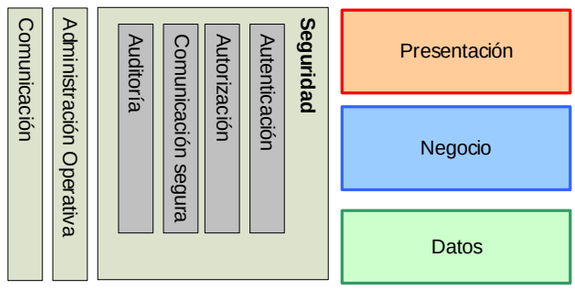


Figura 15: Directivas de seguridad.

3.1 Autenticación

La autenticación corresponde al mecanismo que sirve para identificar en forma segura y univoca a los usuarios, este mecanismo se debe adecuar a los requisitos de seguridad de la aplicación dependiendo del entorne en que opere. La autenticación se debe implementar en la capa de la interfaz de usuario para proporcionar funciones de autorización, auditoria y personalización.

Para implementar la autenticación en la solución móvil se propone que el usuario entregue sus credenciales para demostrar su identidad, como por ejemplo, nombre y contraseña. Estas pueden serán transmitidas de forma segura, en modo encriptado, mediante cifrado SSL (Security Socket Layer).

Con la anterior se hace posible también reutilizar el mecanismo de autenticación que posea la organización, aún mejor si esta posee un mecanismo del tipo SSO (Single Sign On), para así no tener que generar uno específico para la solución móvil.

La figura 16, corresponde a un diagrama de secuencia que ilustra la colaboración entre los diferentes componentes para implementar el mecanismo de autenticación de la aplicación móvil.

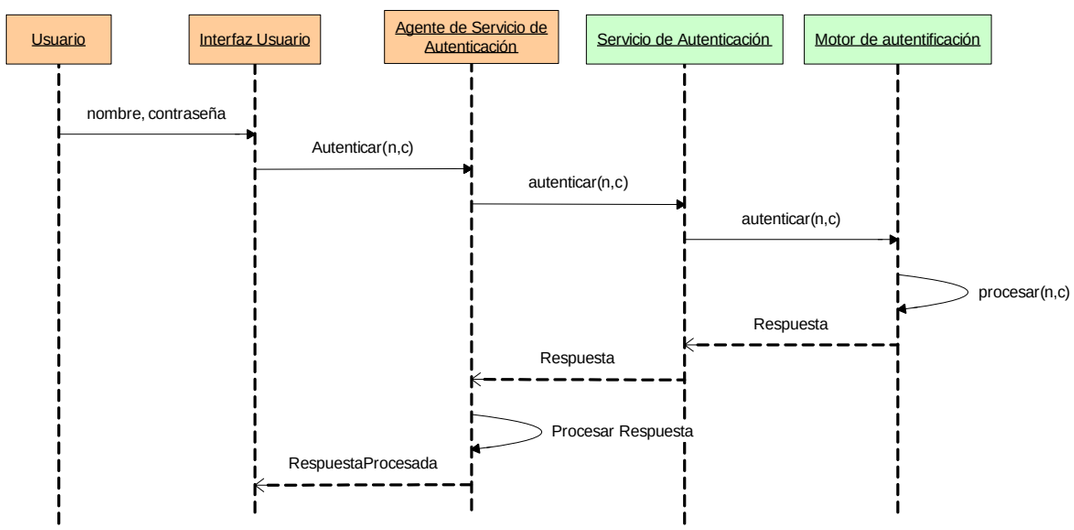


Figura 16: Diagrama de secuencia del proceso de autenticación.

El usuario ingresa su nombre y contraseña en el formulario que despliega en la PDA, la interfaz se encarga de crear una instancia de un agente de servicio que centraliza la comunicación con el servicio que provee la interfaz de autenticación. El servicio tiene la responsabilidad de comunicarse en forma segura con el mecanismo de autenticación que utilice la organización, por ejemplo, LDAP u otro motor de autenticación y autorización de usuarios.

Es importante señalar que dependiendo del contrato establecido por el servicio que provee el mecanismo de autenticación, el agente deberá cumplir

con lo estipulado en dicho contrato, por ejemplo, el servicio puede solicitar información adicional a la del nombre de usuario y contraseña, exigiendo el protocolo de comunicación, SSL, identificador de la aplicación y hasta credenciales de seguridad para consumidores autorizados.

Es importante considerar que la autenticación se puede implementar también a nivel de servicios Web para restringir la llamada a estos. Para ello Compact Framework provee algunas alternativas como por ejemplo la utilización del mecanismo de autenticación básica que forma parte de HTTP[16] que es proporcionado por cualquier servidor Web, por ejemplo: IIS o Apache. Por otro lado esta la alternativa de que si el servidor Web es IIS, se puede utilizar la Autenticación Integrada de Windows para identificar al dispositivo cliente.

3.2 Autorización

La autorización se realiza luego que el usuario ha sido autenticado satisfactoriamente y corresponde al mecanismo de activación de las posibles acciones que el usuario puede realizar en el sistema. Por ejemplo: formularios a los que puede acceder, consultas de información que puede realizar o acceso a evaluaciones de clientes.

Auque la autorización es considerada como parte de la seguridad, su definición corresponde a reglas empresariales y estas deben residir en los sistemas Legacy. Estas reglas deben ser informadas a través de un servicio a la aplicación móvil. De esta forma, la aplicación móvil deberá consultar al

[16] HTTP, HyperText Transfer Protocol.

servicio de reglas de autorización y habilitar/deshabilitar las acciones correspondientes a nivel de interfaz de usuario.

Para evitar realizar dos llamadas independientes a servicios de autenticación y autorización, se propone la creación de un único servicio que se encargue de autenticar al usuario y entregue el nombre de éste y las posibles acciones de podrá realizar en el sistema.

La figura 17 corresponde a un diagrama de secuencia que ilustra la sucesión de mensajes entre los distintos componentes que implementan el proceso de autenticación y autorización para la arquitectura propuesta.

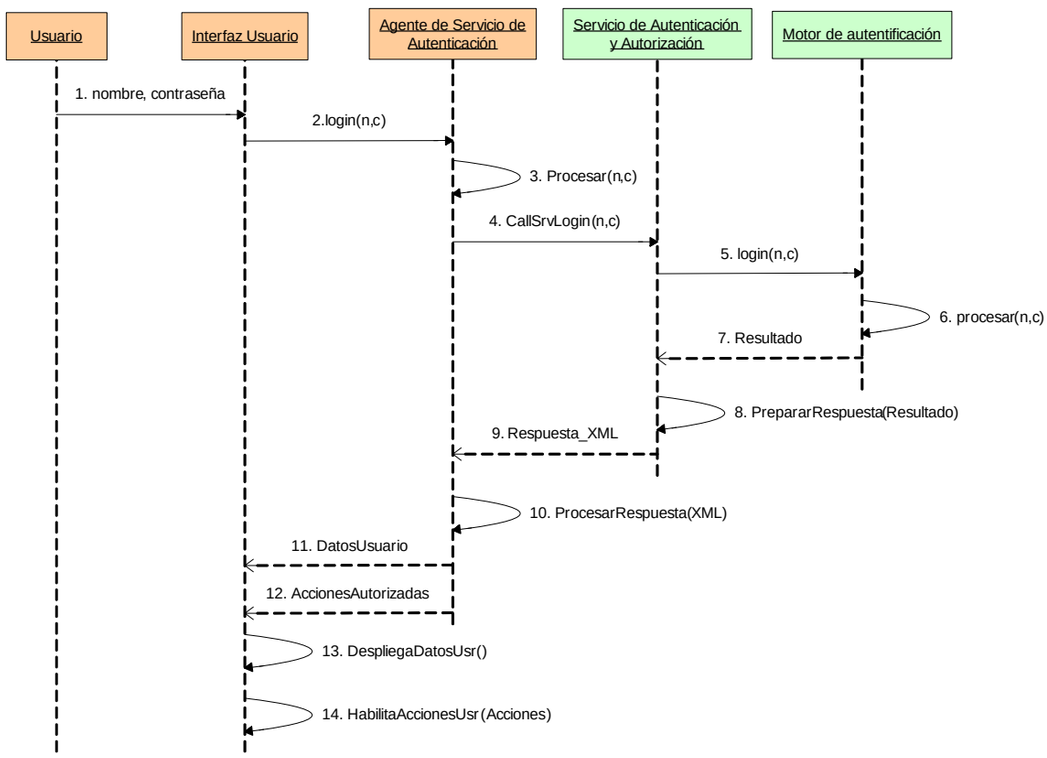


Figura 17: Diagrama de secuencia para autenticación y autorización.

A continuación se explica con mayor detalle cada uno de los mensajes:

1. El usuario introduce su nombre y contraseña en la interfaz del dispositivo móvil.
2. La interfaz de usuario crea una instancia del agente servicio encargado de la autenticación y autorización e invoca su método login con el nombre y contraseña del usuario como parámetros.
3. El agente de servicio procesa la llamada con el fin de preparar la petición según contrato acordado con el servicio proveedor.
4. Realiza invocación al servicio de autenticación y autorización. La comunicación se realiza en forma segura mediante SSL.
5. El servicio de autenticación y autorización se comunica con el motor de autenticación empresarial perteneciente a la organización.
6. El motor de autenticación procesa la solicitud de autenticación y autorización.
7. El motor de autenticación entrega el resulta al servicio. Es importante destacar que el formato de entrega del resultado y canal de comunicación puede ser cualquiera, dependiendo exclusivamente del tipo de servicio de validación de usuario.
8. El servicio de autenticación y autorización formatea el resultado en base a contrato establecido.
9. Entrega la respuesta resultante estructurada en un documento XML, la respuesta contiene datos del usuario y acciones posibles a realizar sobre el sistema. La entrega se realiza en forma segura mediante SSL.
10. El agente de servicio procesa la respuesta XML del servicio y genera objetos que encapsulan los datos del usuario y las acciones autorizadas.
11. El agente de servicio entrega a la interfaz de usuario los objetos que encapsulan los datos del usuario.

12. El agente de servicio entrega a la interfaz de usuario los objetos que encapsulan las acciones autorizadas.
13. La interfaz despliega los datos del usuario.
14. Finalmente la interfaz habilita/deshabilita acciones en base a acciones autorizadas para la sesión del usuario.

3.3 Comunicación segura

La comunicación segura implica la protección de la información que se transmite entre componentes (agentes de servicios) y servicios remotos.

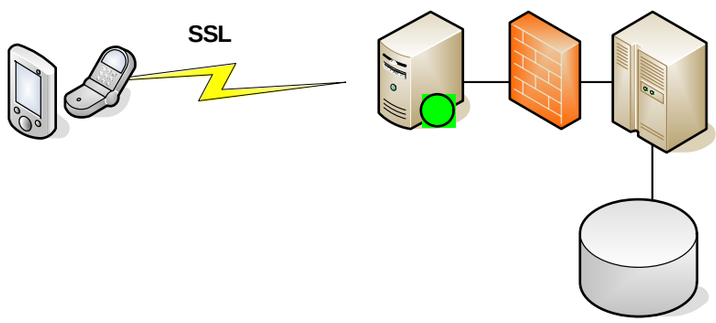


Figura 18: Diagrama red de comunicación.

Windows CE .Net, Pocket PC 2002 y versiones superiores permiten utilizar SSL, que dispone de claves de encriptación de 128 bit, con lo cual se logra un alto nivel de seguridad en la transmisión de datos.

3.4 Auditoria

La auditoria es el mecanismo permite realizar seguimiento de las acciones que realiza el usuario y de la actividad empresarial por motivos de seguridad.

Dado que la aplicación móvil no contiene datos, la auditoria es de responsabilidad del sistema Legacy, pero los contratos establecidos entre los servicios deben considerar la información necesaria para realizar el proceso de auditoria.

4. ADMINISTRACIÓN OPERATIVA

La directiva de administración operativa se ocupa de la ejecución constante de la aplicación, por lo cual abarca el manejo de excepciones, supervisión, supervisión empresarial, metadatos, configuración y ubicación del servicio.

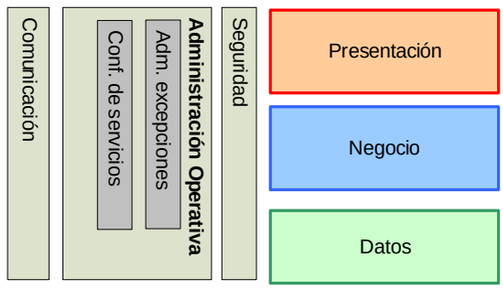


Figura 19: Directivas de la Administración Operativa.

4.1 Administración de excepciones

El manejo de excepciones se ocupa de la detección de errores, de la generación de excepciones, de su diseño, del su flujo de información y de la publicación de información de las excepciones a los distintos usuarios. Es muy importante que todas las aplicaciones implementen controles de excepción para detectar errores en tiempo de ejecución, estas excepciones se deben detectar y resolver si fuese posible, sino se debe mostrar un mensaje descriptivo del error al usuario y proporcionar el medio de registro o publicación de la excepción.

La administración de excepciones se debe aplicar en componentes de interfaz de usuarios, en componentes de procesos empresariales, y en componentes de acceso a datos.

Para el caso de la solución móvil se debe incorporar un componente que centralice el manejo de excepciones, esta componente debe tomar los errores generados en las distintas capas de la aplicación, procesarlos, registrarlos y presentarlos de manera comprensible para el usuario.

4.2 Configuración de servicios

La configuración de la aplicación móvil guarda relación con el conjunto de valores, que en muchos casos cambian con el tiempo, que requiere la aplicación para funcionar. Por ejemplo, URL de cada servicio Web.

Los agentes de servicio necesitan de una URL para invocar al servicio Web respectivo, dado que estas direcciones pueden cambiar, se propone utilizar un catálogo de servicios Web que almacene cada una de estas direcciones evitando así dejar esta definición como parte del código fuente compilado. El catálogo de servicios debe ser implementado a través de un archivo XML que almacene una lista del tipo [NombreServicio, URL].

La figura 20 corresponde a un diagrama de secuencia que ilustra la sucesión de mensajes que permite a un agente de servicios obtener la URL del servicio y realizar la invocación mediante una clase proxy.

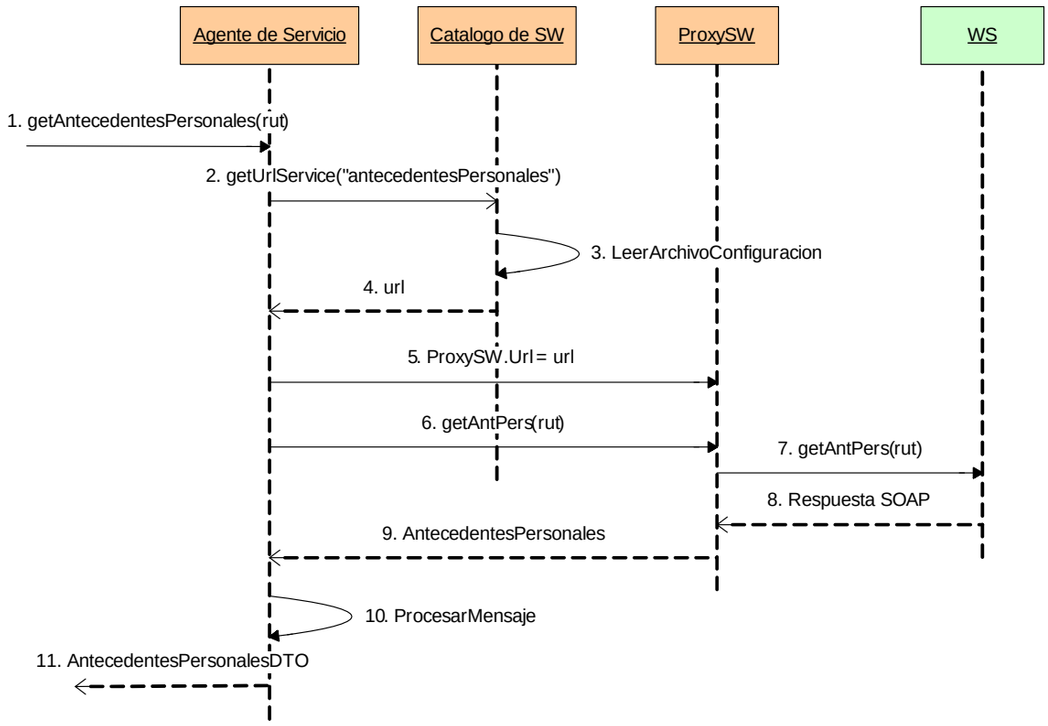


Figura 20: Diagrama de secuencia de invocación de servicio Web con uso de un catálogo de servicios.

1. Al agente de servicio llega una llamada que solicita los antecedentes personales de un cliente, para ello se le entrega el Rut.

2. El agente de servicio solicita al catálogo de Servicios Web la URL del servicio que permite recuperar los antecedentes personales del cliente.

3. El catálogo de servicios Web se encarga de leer el archivo de configuración y recuperar la URL del servicio que atenderá las peticiones de antecedentes personales de clientes.

4. El Catálogo de servicios Web devuelve la URL solicitada por el agente de servicio.

5. El agente de servicio fija la URL correspondiente en el Proxy.

6. El agente de servicio solicita al Proxy los antecedentes personales del cliente mediante la operación getAntPers.
7. El Proxy invoca el método getAntPers del servicio Web, a través de HTTPS[17].

8. El servicio Web devuelve los antecedentes personales solicitados por el Proxy.

9. El Proxy hace entrega de los antecedentes personales al agente de servicio.

10. El agente de servicio debe realizar el procesamiento de la respuesta obtenida

11. El agente hace entrega de antecedentes personales del cliente, esta entrega se hace a través del tipo o estructura de datos esperada por el invocador.

El objeto Catalogo de Servicios Web (catalogo de SW, en el diagrama) debe ser implementado bajo el patrón de diseño *Singleton* para restringir la existencia de una única instancia de la clase, de esta forma se evita que se realicen múltiples lecturas al archivo XML de configuración de servicios Web.

[17] HTTPS, corresponde a la versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado más apropiado para el tráfico de información sensible que el protocolo HTTP.

5. COMUNICACIONES

Esta directiva se ocupa de la forma en que los componentes se comunican entre sí, ya sea con otras aplicaciones o entre los componentes de la misma aplicación, este criterio se ocupa de la sincronización, formato y protocolo de las comunicaciones.

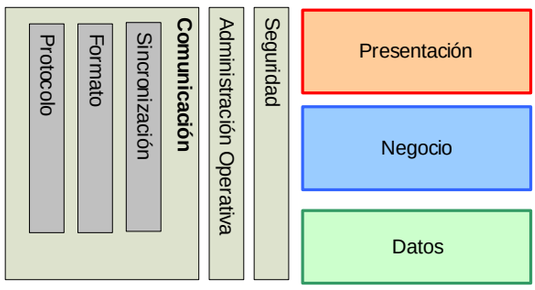


Figura 21: Directiva de comunicaciones.

Aunque la arquitectura presentada no lo restringe, se recomienda utilizar comunicación sincrónica basada en Servicios Web para la comunicación con servicios externos que encapsulan la lógica empresarial, en un formato SOAP[18] y un protocolo HTTPS.

[18] SOAP, de las siglas de Simple Object Access Protocol, es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

6. CAPA DE PRESENTACIÓN

Para la capa de presentación se optó por mantener el uso de formularios de .NET Compact Framework, el cual constituye un subconjunto de completo del espacio de nombres System.Windows.Forms de la plataforma .NET. La implementación en .NET Compact Framework se encuentra optimizada en desempeño y tamaño para dispositivos móviles, emulando el aspecto de las aplicaciones Windows escritorio.

Tal como lo resume el capitulo 2 de este proyecto de titulación, la capa de presentación implementada como formularios debe proporcionar un mecanismo de interacción para con el usuario, permitiendo el ingreso y despliegue de información en forma segura.

Las responsabilidades más relevantes de este componente son:

1. Restringir los tipos de datos de entrada de los usuarios.
2. Garantizar la entrada de datos obligatorios.
3. Transformar la entrada proporcionada por los controles de usuario a valores necesarios para que otros componentes realicen su trabajo.
4. Procesar datos de una entidad empresarial e informar el estado al usuario.
5. Personalizar el aspecto de la aplicación en función de las preferencias del usuario.

7. CAPA DE COMPONENTES EMPRESARIALES

En la capa de componentes empresariales se utilizarán 2 tipos de componentes: componentes empresariales y entidades empresariales.

7.1 Componentes Empresariales

En una aplicación normal, los componentes empresariales implementan la lógica empresarial de la aplicación, pero en este caso, la solución móvil no incluye reglas de negocio, sino que se reutilizará lógica preexistente que reside en uno o más componentes de sistemas legacy de la organización. Este último punto es muy importante ya que forma parte de uno de los supuestos en los que se basa esta arquitectura.

Dado que la lógica de negocio es reutilizada a partir de componentes preexistentes, los componentes empresariales tampoco implementarán transacciones de ningún tipo. En caso de ser necesarias, las transacciones deberán ser administradas a nivel de los componentes del sistema Legacy.

Finalmente, los componentes empresariales de la arquitectura propuesta son invocados desde la capa de presentación por los formularios (1), pueden interactuar con otros componentes empresariales (2) o pueden invocar agentes de servicios para realizar llamadas a servicios externos (3). La figura 22 representa los aspectos antes mencionados.

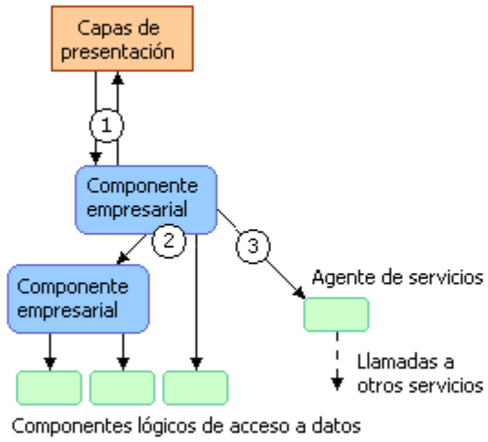


Figura 22: Colaboración de componentes empresariales.

7.2 Entidades Empresariales

Tal como se describe en el capitulo 2 de este proyecto de tesis, las entidades empresariales representan los conceptos propios de un proceso de negocio y no cuentan con lógica empresarial, únicamente poseen atributos y métodos para acceder a ellos y actualizarlos. Ejemplos de estos son: Persona, factura, orden de pago, solicitud.

Estos componentes son mayormente utilizados para transportar información a través de las diferentes capas de la aplicación.

El siguiente diagrama de secuencia, en la figura 23, ilustra la colaboración entre la interfaz de usuario, componente empresarial, entidad empresarial y agente de servicio para realizar una operación de actualización de información de un cliente.

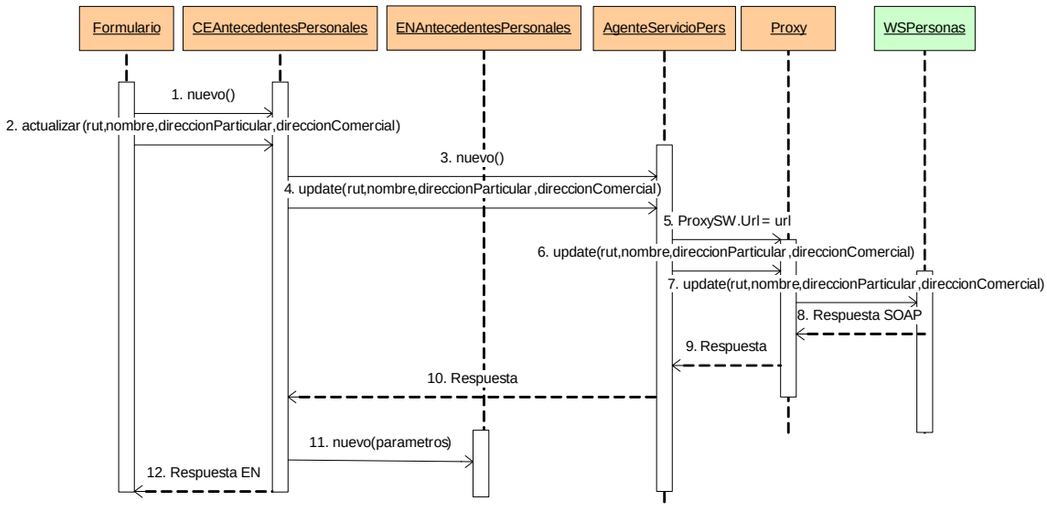


Figura 23: Diagrama de secuencia de colaboración para componentes de la arquitectura.

La secuencia descrita en el diagrama realiza lo siguiente:

1. El formulario crea una instancia del componente empresarial de antecedentes de personas.

2. El formulario invoca el método de actualización de antecedentes personales actualizar(…) del componente empresarial.

3. El componente empresarial crea una instancia del agente de servicio, AgenteServicioPers, encargado de proveer el servicio de actualización de antecedentes personales.

4. El componente empresarial invoca el método update(…) del agente de servicio.

5. El agente del servicio fija la URL del servicio Web que resuelve la operación de actualización en su proxy.

6. El agente de servicio recurre a su proxy para realizar la llamada al servicio Web.

7. El Proxy invoca al servicio Web mediante un canal seguro de comunicación (SSL) y SOAP.

8. El servicio Web WSPersonas, responde a la operación.

9. El Proxy procesa el mensaje SOAP enviado por el servicio Web y le responde al agente de servicio.

10. El agente de servicio valida la respuesta enviada y responde al componente empresarial.

11. El componente empresarial CEAntecedentesPersonales crea una instancia de la entidad de empresarial ENAntecedentesPersonales para responder a la interfaz de usuario.

12. CEAntecedentesPersonales responde a la actualización de antecedentes con ENAntecedentesPersonales completamente actualizado.

8. CAPA DE DATOS

8.1 Datos centralizados en sistemas legacy

El almacén de datos se mantiene centralizado en los sistemas Legacy. El mecanismo de acceso y actualización a los datos desde la aplicación móvil lo proveen los servicios Web que reutilizan la lógica de negocio encapsulada en diferentes tipos de componentes preexistentes en el sistema Legacy.

El siguiente diagrama ilustra la comunicación entre los diferentes componentes para acceder a información almacenada en repositorios del sistema Legacy e información almacenada localmente.

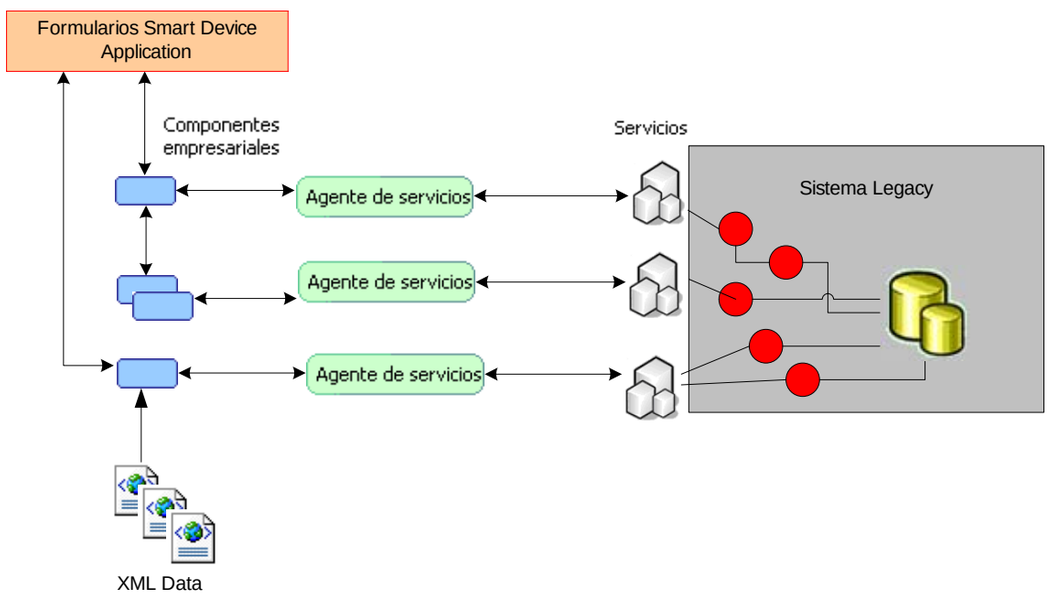


Figura 24: Acceso a datos a través de agentes de servicio.

8.2 Almacenamiento de datos en dispositivo móvil

Con la finalidad de disminuir la comunicación a través de la red y mejorar el desempeño de la aplicación, se recomienda una alternativa de almacenamiento persistente en los dispositivos móviles. Es posible disminuir el flujo de información entre proveedor y consumidores mediante el almacenamiento de datos poco cambiantes en un archivo de configuración XML. Para ello se debe implementar un servicio de recuperación y actualización de información. Este servicio entregará los documentos XML necesarios la primera vez que el dispositivo utilice la aplicación y notificará el proceso de actualización cada vez que los datos cambien. Esta alternativa se hace muy deseable sobre todo cuando el servicio de comunicación es contratado con cargo a volumen de información transmitida.

Dentro de esta categoría de datos se propone incluir:

- País de procedencia
- Estado Civil
- Régimen matrimonial
- Calidad de trabajador
- Nivel educacional
- Profesión u oficio
- Regiones, provincias y comunas
- Giro comercial

Adicionalmente se puede utilizar este servicio para distribuir cambios en URL de servicios Web, por lo cual la aplicación deberá actualizar la configuración de servicios Web (catálogo de servicios).

8.3 Agentes de servicios

Los agentes de servicio representan el componente esencial dentro de la arquitectura propuesta en este trabajo de título, ellos permiten a la aplicación móvil comunicarse e intercambiar información con el sistema Legacy. La aplicación proveedora contiene la lógica de negocio y el almacén de datos, por ello, el factor crítico a enfrentar lo constituye el modo de comunicación con ella.

La solución móvil propone una dinámica de intercambio de mensajes SOAP entre los diversos agentes de servicio y los servicios Web.

En el capitulo 2 se describe en detalle las responsabilidades de este tipo de componentes, entre las más importantes destacan:

1. Encapsular el acceso a servicios externos.
2. Validación y formateo de datos de entrada y salida para otros componentes y servicios.
3. Implementar la comunicación segura con los servicios.
4. y finalmente, proveer de la URL del servicio Web al componente proxy asociado.

Los agentes de servicios hacen uso de componentes proxy que facilitan la comunicación SOAP con los servicios externos. Los proxy deben contener la información necesaria para saber donde se ubican y como realizar las llamadas a los métodos expuestos en el servicio Web. La plataforma .Net provee herramientas para automatizar la creación de los componentes proxy, evitando así la codificación manual de estos componentes.

El componente proxy, es implementado a través del patrón *Service Gateway,* expuesto en el capitulo 3 de este documento. Los beneficios más relevantes obtenidos al utilizar este patrón son:

1. Desacoplar, en una clase independiente, la complejidad de acceso al servicio del resto de la aplicación.
2. Reutilización. El proxy puede ser reutilizado por otros agentes de servicio que necesiten comunicarse con el servicio Web.

8.4 Proveedor de servicios Web

El proveedor de servicios Web debe publicar los servicios necesarios para disponer de las funcionalidades requeridas en la aplicación móvil. Los servicios deben ser expuestos sobre un canal seguro por medio de SSL y mensajes SOAP sobre HTTP. Por lo general, este requerimiento deberá ser considerado como actividad de desarrollo en el proyecto.

Uno de los supuestos en los cuales se basa la arquitectura, es que la lógica empresarial reside en componentes de software preexistentes. Con el patrón *Service Interface*, presentado en el capitulo 3 de este documento, es posible crear interfaces de servicio que permiten exponer la lógica empresarial para ser utilizada por otras aplicaciones, en este caso por el proxy del agente de servicio. Esto permite desacoplar los detalles de la implementación de la interfaz (comunicación, seguridad, transformación de datos, etc.) de la lógica empresarial.

CAPÍTULO VI: CONCLUSIONES

Las características fundamentales y las restricciones principales de una solución tecnológica deben estar dirigidas por las necesidades de optimización de un proceso de negocio o flujo de trabajo organizacional. Es importante que previo a la concepción de una solución candidata, se plasmen en primer lugar las necesidades reales de usuarios y luego, a partir de ellas, deben evaluarse y comprometerse las características del producto de software. Existen muchos casos en los cuales soluciones aparecen antes de completar la evaluación de necesidades de los stakeholders, dando como resultado que los procesos de negocios son adaptados al software, perdiendo así la organización el dinamismo necesario para enfrentar un mercado en constante cambio. Muchas organizaciones del área financiera ya han invertido grandes sumas de dinero en plataformas y soluciones tecnológicas de punta, que actualmente se encuentran estabilizadas y alineadas con los procesos de negocio, por lo cual, las soluciones móviles deben estar enfocadas a integrarse a estos sistemas haciendo uso de estándares tecnológicos, como por ejemplo SOAP, que permiten utilizar información y lógica empresarial extendiendo así los sistemas más allá de la intranet corporativa.

PER-Móvil es una solución que posee restricciones de diseño muy propias de las empresas de hoy, en general, todas las la organizaciones como bancos e instituciones financieras, compañías de seguros e incluso la industria del retail, ya poseen dentro de sus sistemas Legacy, la lógica de su negocio, es así que sólo les falta llevar dicha lógica al lugares precisos para la venta.

El caso en estudio, PER-Móvil, permitió agilizar el proceso de comercialización de créditos a microempresarios. La solución se basó en creación de servicios Web que permitieron comunicarse con componentes de software preexistentes que encapsulaban reglas de negocio y acceso a

información. Estos servicios fueron utilizados desde la aplicación PER-Móvil sin incorporar nueva lógica empresarial. Gracias a esto, se logró contar con información y procesamiento de solicitudes de crédito en terreno que optimizó significativamente el proceso de venta. A continuación se muestra una tabla comparativa entre la arquitectura de PER-Móvil y la arquitectura propuesta en este trabajo de titulación.

Característica	Arquitectura PER-Móvil	Arquitectura propuesta
Seguridad	▪ Únicamente para autentificación de usuario.	▪ Autenticación de usuario. ▪ Autorización de usuarios, en base a perfiles. ▪ A nivel de servicios Web. ▪ Uso de SSL para canal de comunicación.
Administración Operativa	▪ No incorporado	▪ Administración de excepciones. ▪ Configuración de servicios.
Comunicación	▪ Comunicación sincrónica a través de servicios Web. ▪ Formato SOAP ▪ Protocolo HTTP.	▪ Comunicación sincrónica a través de servicios Web. ▪ Formato SOAP ▪ Protocolo HTTPS.
Presentación	▪ Formularios	▪ Formularios
Negocio	▪ Pseudo-componentes empresariales. ▪ Entidades empresariales.	▪ Componentes empresariales ▪ Entidades empresariales
Datos	▪ Centralizados en sistemas legacy y accesibles a través de servicios Web. ▪ Acceso a servicios a través de un único componente.	▪ Centralizados en sistemas legacy y accesibles a través de servicios Web. ▪ Caché de datos de uso frecuente con notificación y mecanismo de actualización en línea. ▪ Agentes de servicio.
Patrones de Diseño	▪ Utilización parcial de Service Gateway (proxy). ▪ Utilización parcial de Service Interface.	▪ Service Gateway (proxy) ▪ Service Interface ▪ Singleton
Escalabilidad	▪ No posee capacidad de de comunicarse con otros servicios. Para lograr esto se debe realiza un desarrollo adicional.	▪ La solución móvil puede acceder a nuevas funcionalidades, expuestas por sistemas externos a la organización, a través de servicios Web. ▪ La solución móvil puede incorporar nuevos tipos de usuarios en base a definición de perfiles. ▪ La solución móvil puede acceder a otros sistemas de la organización, a través de servicios Web.
Reutilización	▪ Se dificulta la reutilización de código al no haber una clara definición de responsabilidades a nivel de componentes.	▪ Los servicios expuestos pueden ser consumidos por otras aplicaciones. ▪ El bajo acoplamiento entre los componentes permite su reutilización y eventual reemplazo.
Mantención	▪ La carencia de documentación, hace difícil identificar la	▪ Al poseer una arquitectura definida y bien documentada

Característica	Arquitectura PER-Móvil	Arquitectura propuesta
	definición de su arquitectura. Por lo cual, se debe recurrir a una revisión rigurosa de código para establecer las responsabilidades de cada componente, lo que dificulta la mantención.	facilita las posibles mantenciones.
Disponibilidad	▪ No incorporado.	▪ La disponibilidad de la aplicación móvil depende de los servicios Web que son consumidos por ella. Se recomienda utilizar balanceadores de carga para distribuir el procesamiento de solicitudes en diversos servidores.

Tabla 3: Comparativa de arquitectura de PER-Móvil y la arquitectura propuesta en trabajo de titulación.

La propuesta de Microsoft .Net Compact Framework, posee muchas de las capacidades, herramientas y facilidades disponibles en el la arquitectura Microsoft .Net, facilitando así la construcción de aplicaciones a programadores sin experiencia en proyectos para dispositivos móviles. Entre estas ventajas se debe mencionar que la plataforma se encuentra potenciada para crear aplicaciones bajo la arquitectura orientada a servicios, lo cual permite optimizar los tiempos y costos implícitos en el desarrollo. Otra ventaja significativa es que el ambiente de desarrollo es el mismo para ambas plataformas. Se debe tener en cuenta que aun existen falencias en la seguridad que deben ser mejoradas.

La autenticación básica para mejorar la seguridad de servicios Web presenta algunas desventajas evidentes, necesita que se intercambien mensajes entre el emisor y el receptor de mensajes SOAP antes de enviar el mensaje requerido de forma segura. Lo anterior puede degradar el rendimiento de la solución móvil al realizar mayor intercambio de datos sobre el canal de comunicación. La especificación WS-Security está intentando agilizar estos procesos. WS-Security abandona las técnicas de protocolo de transferencia para adoptar el modelo de seguridad centrado en el mensaje, utilizando firmas

XML y encriptación de XML en el encabezado de SOAP. IBM, VeriSign y Microsoft han estado definiendo este estándar desde el año 2002, pero aun no se hace muy conocido para su uso.

En organizaciones que han asumido y madurado con buenas prácticas el desarrollo de software, se ha comenzado a definir un nuevo rol llamado arquitectos de software, conformado por ingenieros altamente especializados cuya responsabilidad es diseñar la arquitectura tecnológica que soportará las aplicaciones empresariales. Los arquitectos deben conocer y aplicar las ventajas de los patrones arquitectónicos y de diseño en su trabajo diario.

Contar con una arquitectura bien definida, permite agilizar el proceso de desarrollo y obtener como resultado soluciones robustas. Esto es una gran ventaja competitiva para empresas que desarrollan software, dado que la definición de arquitectura puede ser aplicada varios proyectos sacando ventaja del esfuerzo invertido en concebir la arquitectura en cuestión.

BIBLIOGRAFIA

- [CNC, 2005] Revista de la Cámara Nacional de Comercio, “Clasificación de las empresas en Chile” www.cnc.cl.

- [MIC, 2002] Microsoft, “Arquitectura de aplicaciones de .NET: Diseño de aplicaciones y servicios”. Última actualización 31 de mayo 2004. http://www.microsoft.com/spanish/msdn/arquitectura/das/distapp.asp

- [MIC2, 2002] Microsoft. Abdul Rasheed Usman.“Consulta de UDDI con Windows CE.NET”. Publicación Noviembre 2002. http://www.microsoft.com/spanish/msdn/articulos/archivo/210203/voices/winceuddi.asp

- [ARC,2006] Microsoft. “Arquitectura .Net”. http://msdn.microsoft.com/architecture. Última actualización 2006.

- [PRA, 2006], “Mejores prácticas sobre .Net”. Última actualización 2006. http://msdn.microsoft.com/practices

- [COM, 2006] Microsoft. “.net Compact Framework”. Última actualización 2006. http://samples.gotdotnet.com/quickstart/CompactFramework/

- [NET, 2003] Wigley, Wheelwright. “Microsoft .NET COMPACT FRAMEWORK”. Publicado año 2003.

- [ASP, 2003] Roxburgh Peter, Wigley Andy. “Building Microsoft ASP.NET Applications for Mobile Devices”. Publicado año 2003.

- [ENT, 2003] Cunningham Ward, David Trowbridge, Dave Mancini, Dave Quick, Gregor Hohpe, James Newkirk, David Lavigne “Enterpise Solution Patterns using Microsoft .NET”. Publicado año 2003.

- [BOO, 2000] Jacobson; Booch; Rumbaugh. “El proceso Unificado de desarrollo de software”. Editorial Addison Wesley. Publicado año 2000.

- [STE, 2003] Stelting, Maassen. “Patrones de diseño aplicados a java”. Editorial Pearson. Publicado año 2003.

- [BUS, 1996] Buschmann, “Pattern-Oriented Software Architecture”. Volume 1: A System of Patterns. John Wiley & Sons. Publicado año 1996.