UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA

Aplicación sobre Dispositivos móviles para la Monitorización de Contact Centers Multitenant

Ángel Manuel Pérez Villar

Septiembre 2012

APLICACIÓN SOBRE DISPOSITIVOS MÓVILES PARA LA MONITORIZACIÓN DE CONTACT CENTERS MULTITENANT

Autor: Ángel Manuel Pérez Villar

Tutor: José Javier Bonastre (Indra)

Ponente: Germán Montoro Manrique

Agradecimientos

Son muchos los que debo agradecer y pocas las palabras que abracen el significado tan importante que implican su mero nombramiento. Por lo que, no me esforzaré en engrosar este apartado de momentos o situaciones. Me serviré de la simplicidad para resaltar su importancia.

A mi tutor en Indra, por su paciencia y consejo constante.

Los amigos de siempre, quiénes me apoyaron desde el principio.

Los compañeros de la Universidad que hicieron más amena esta etapa.

Mi novia por su empuje esencial, cariño y fuerza.

Mi familia por básicamente todo.

A todos, gracias.

Índice de Contenidos

Índice de Figuras

Introducción

Los centros de atención al cliente se han convertido en un servicio indispensable en cualquier compañía, transformándose a su vez en una vía óptima para relacionarse con sus clientes. Se encarga de administrar, proveer soporte y asistencia a éste ,según los productos, servicios o información necesitada, generando en ellos, cierto nivel de satisfacción.

Este tipo de centros están completamente dominados por grandes empresas que requieren contacto permanente con sus clientes, de manera que, crecen en competitividad. El tipo de empresa abarca diferentes ámbitos, desde firma de pedidos por catálogo, de software o hardware, pasando incluso por el servicio público.

También denominados C*all Centers*, son controlados por compañías proveedoras de servicios, operando independientemente o estando interconectados con otros centros.

Y como toda actividad que se precie, siempre hay un supervisor, el cual debe controlar y registrar todo este flujo de datos al igual que monitorizar sus respectivos agentes.

Ahora bien, vivimos en una sociedad en pleno movimiento, gobernada por la necesidad de tener todo bajo control, haciéndose imprescindible en nuestra vida cotidiana el disponer del teléfono móvil más avanzado o con la última Tablet.

Y es, en este contexto en el cual se abarca el proyecto, en abrazar esa necesidad y ofrecer las tareas pertinentes que debe controlar el supervisor de los centros de atención al cliente y ofrecérselo de manera móvil usando para ello la plataforma iOS y el lenguaje de desarrollo (Objective-C) siendo todo sustentando por Apple.

Motivación

Este proyecto nace de la mano de mi Tutor, Javier Bonastre Vallés con la idea de ofrecer a los usuarios de un sistema de monitorización y supervisión de plataformas de Contact Centers, un prototipo para analizar cómo los dispositivos móviles pueden facilitar estas actividades tecnológicamente complejas a los responsables de los centros de atención de llamadas, que no siempre disponen de un expertise tecnológico para entender las herramientas que ofrecen los suministradores tradicionales.

Actualmente los responsables de los centros de atención de llamadas disponen de una herramienta web para la funciones de administración, monitorización y supervisión de agentes y servicios. No obstante, por su forma de trabajo, sería necesario que pudieran disponer de una aplicación en un dispositivo en movilidad que soporte las funciones más comunes de supervisión.

La aplicación mencionada deberá ser capaz de recoger datos de un *webservice* alojado en los sistemas de back-end de manera rápida y fluida sin afectar al rendimiento y mostrando después esos mismos datos en pantalla siendo su visualización intuitiva y fácil para el usuario. En este sentido, la plataforma iOS ofrece innumerables interfaces visuales cómodas y sencillas aprovechando los recursos del dispositivo .

Objetivo

El objetivo es desarrollar una aplicación de supervisión y monitorización de agentes de *call centers* usando el lenguaje de programación de Apple: Objective-C sobre el dispositivo iPad de una forma sencilla e intuitiva. Esta herramienta debe permitir a los responsables, acorde a su categoría y nivel de responsabilidad acceder a la información de la actividad que realizan los agentes que dependen jerárquicamente de él, así como disponer en tiempo real de información de los niveles de atención de los servicios de los que son responsables .

Para ello, se creará un interfaz visual que muestre toda la información y que permita la interacción con el usuario de forma sencilla. Las pruebas se realizarán usando interfaces de *webservice* sobre los sistemas de back-end de la plataforma de supervisión.

Organización de la Memoria

En un primer apartado se presenta el proyecto, motivación y objetivos.

En el segundo capítulo se exhibe el estado del arte: plataformas móviles existentes, productos similares.

En la tercera sección se sumerge en el diseño de la aplicación junto con su implementación así como las diferentes opciones que se barajaron explicando el porqué de la elección final.

En un cuarto punto, se muestran las pruebas realizadas junto con los resultados explicados.

Las conclusiones obtenidas junto con el trabajo futuro, mencionando aquí diferentes posibilidades de mejora, aparecen después.

Por último, los apéndices.

Introducción

Hoy en día no pasa desapercibido el potencial económico que presentan las plataformas móviles y la oportunidad que suponen sus ingresos y su visibilidad para muchos desarrolladores [1].

El mercado móvil ha ido evolucionando hasta ofrecer multitud de dispositivos: smartphones, netbooks, tablets..) todos ellos adaptados y pensados para facilitar las necesidades de los usuarios. Y dentro de cada dispositivo las posibilidades de plataformas móviles son varias, cada una con un diseño e implementación único.

La lucha por la supremacía de las plataformas móviles está candente. **Android y iPhone**, así como **BlackBerry o Nokia** son varios de los actores que más destacan, siendo el vencedor iPhone en el caso de España [2].

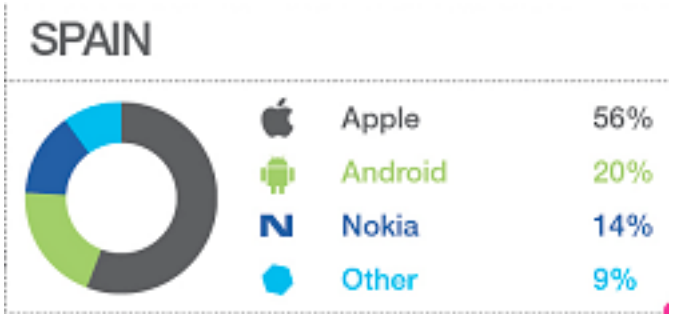


Ilustración 1-Situación sistemas operativos para smartphones

Cada una de estas plataformas ofrece unas características puntuales tanto en software como en hardware, debido a que en algunos casos, véase iPhone o iPad el software va ligado al dispositivo proporcionando interfaces múltiples. Bien es cierto que las plataformas móviles tienen limitaciones de recursos: capacidad del procesador, tamaño de memoria, etc. [5] eso no reduce su atractivo para la investigación y desarrollo.

A continuación, se hará un repaso sobre las plataformas móviles más importantes de la actualidad para ahondar en lo citado en este apartado, dando paso después a las posibilidades que ofrece el mercado similares a lo defendido en este proyecto así como las semejanzas con otras aplicaciones, en este caso únicamente sobre la plataforma escogida.

Plataformas móviles

Android

Es un sistema operativo móvil basado en Linux [3] que junto con aplicaciones middleware está enfocado para ser utilizado en dispositivos móviles como smartphones, tablets y otros dispositivos [4]. Fue creada por Google y Open Handset Alliance [6] en un intento de competir contra Apple y Microsoft.

Android se inició oficialmente el día 22 de octubre del 2008 en Estados Unidos con grandes aportes como una ventana de notificación desplegable, widgets en la pantalla de inicio, integración de Gmail y un Android Market [7]. Desde ese día no ha parado de evolucionar alcanzando la versión 3.0 enfocado a tablets hasta culminar con la versión 4.0 lanzada junto con el móvil Galaxy Nexus de Samsung [8].

El sistema permite programar aplicaciones en una variación de Java llamada Dalvik [9]. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java [10]. Esta sencillez, junto a la existencia de herramientas de programación gratuitas, hacen que una de las cosas más importantes de este sistema operativo sea la cantidad de aplicaciones disponibles, que extienden casi sin límites la experiencia del usuario.

A día de hoy son más de 600.000 las aplicaciones disponibles en su tienda oficial: Google Play (antes llamado Android Market) sin contar diversas tiendas no oficiales.

Esto hace a Android una apuesta de futuro.

Blackberry

Desarrollada por la compañía Research in Motion (RIM) [11] integra el servicio de correo electrónico móvil. Como toda alternativa posible, Blackberry incluye los servicios típicos de un smartphone: agenda, bloc de notas, calendario, etc. Sumando a estas ofertas un teclado *Qwerty* [12], el cual permitía escribir mails de una manera rápida y sencilla así como cualquier mensaje.

El primer dispositivo fue lanzado en 1999 en Canadá pero no le llegó la fama hasta el 2003. Los Blackberry usan un sistema operativo propio: el Blackberry OS multitarea diseñado para hacer uso de su trackpad o a través del teclado. Otra característica importante de estos móviles es su denominado Blackberry Messenger, software cuyo fin es enviar y recibir mensajes instantáneos, notas de voz, imágenes o vídeos mediante PIN. Esto permitió un despegue en sus ventas hacia un público más general expandiéndose más allá del ámbito de negocios al cual estaba enfocado [13].

Al igual que Android, también dispone de una tienda oficial: Blackberry App World disponible de forma gratuita pudiéndola descargar tanto desde su móvil como por el ordenador [14].

Windows Phone

Es la plataforma de Microsoft diseñada para su uso en Smartphones u otros dispositivos [15], siendo el sucesor de Windows Mobile, debido a la incompatiblidad del primero con dispositivos Windows Mobile y software [16].

Hace uso del sistema operativo Windows CE, apoyado en una serie funciones básicas de la API de Microsoft Windows. Para crear las respectivas aplicaciones se hace uso del entorno de desarrollo Visual Studio [17]. Está diseñado para ser similar a las versiones de escritorio de Windows estéticamente. Además, existe una gran oferta de software de terceros disponible para Windows Mobile, la cual se puede adquirir a través de Windows MarketPlace [18].

Originalmente apareció bajo el nombre de Pocket PC como una ramificación de desarrollo de Windows CE para equipos móviles con capacidades limitadas. En la actualidad, la mayoría de los teléfonos con Windows Mobile vienen con un estilete digital que se utiliza para introducir comandos pulsando en la pantalla.

Windows Mobile ha evolucionado y cambiado de nombre varias veces durante su desarrollo, siendo la última versión la llamada Windows Phone 7 anunciada el 15 de febrero del 2010 y sujeta a disponibilidad a finales de 2010.

Como resultado de la alianza estratégica de Microsoft y Nokia en octubre de 2011 [19] fue presentado el Nokia Lumia 800, primer terminal de la compañía finlandesa que aplica Windows Phone.

iOS

Antes llamado iPhone OS, es la plataforma de Apple usado tanto el iPhone como iPod Touch, iPad o Apple TV.

Su andadura comenzó en 2007 con la salida del iPhone incorporando aplicaciones tan conocidas actualmente como mail, fotos, calculadora, iPod, etc. Ese mismo año Apple presentó el primer kit de desarrollo, en parte a las capacidades existentes. Arropando esta idea, nació el *App Store* [20], la mayor tienda online de todas las plataformas mencionadas así como la mayor en términos de ventas [21].

Para hacernos una idea de su enorme éxito, menos de seis meses después, a comienzos de 2009, *el App Store* ya había alcanzado las 500 millones de descargas y cuatro meses más tarde se duplicó esa cifra, de modo que el crecimiento era exponencial y muy agudo: el 22 de enero de 2011 se descargó la aplicación número diez mil millones y ya se ha sobrepasado la cantidad de veinticinco mil millones de descargas [24].

Después llegaría la actualización iPhone OS 3.0, cuyas características más destacables de esa versión fueron las notificaciones *push* para advertir al usuario de los procesos que ocurrían en las aplicaciones mientras estas estaban cerradas, y la liberación de un SDK con más de 1000 APIs que los desarrolladores podían aprovechar en sus propios proyectos.

A partir de ahí, iOS 4.0 que cambió su nombre de iPhone OS coincidiendo con la llegada del tablet de Apple iPad, llegando en 2010. Le siguieron el iOS 5.0 y el que ya es una realidad: iOS 6.0 [23].

El lenguaje de desarrollo es Objective-C [22], pudiendo recurrir a todo lo necesario gracias a su SDK. Los diferentes dispositivos de Apple: iPad, iPod Touch o iPhone hacen uso de una fácil portabilidad entre sus aplicaciones lo que permite expandir enormemente su oferta.

Alternativas de mercado

En el mercado existen diversos productos enfocados hacia la monitorización o supervisión de agentes de centros de llamadas. En este apartado se resumirá un poco cada alternativa.

SimplyCT

Ilustración 2-SimplyCT

Como primer ejemplo, se encuentra **SimplyCT** [25]. Es un software de centros de llamadas. No requiere hardware de ningún tipo y trabaja tanto para llamadas entrantes o salientes de dichos centros. VoIP, web chat o web call-back junto con email están combinadas de manera sencilla sin afectar al coste.

SimplyCT unifica las comunicaciones equipándolas de interacción con el cliente mediante voz, live chat, email y fax de manera eficiente y efectiva.

La gran ventaja es que no requiere hardware o instalación previa. Su precio empieza en 80 dólares por mes.

Esta alternativa de producto sin embargo, no tiene soporte para aplicaciones móviles, es decir, se descarta su uso en cualquier móvil o tablet.

RingCentral

Ilustración 3-RingCentral para Blackberry y Android

Esta alternativa [26] aún no centrándose tanto como la anterior en centros de llamadas sí ofrece cierto control sobre las llamadas recibidas, controlando todas. También permite a los clientes acceder de manera sencilla a directorios o departamentos de la compañía pertinente, así como permitir que cada empleado tenga una asignación determinada de teléfono o voicemail para su control.

Una ventaja clara es poder acceder a los registros de llamadas, RingOut o voicemal visual desde cualquier dispositivo móvil ofreciendo la posibilidad tanto en Blackberry como Android como iPhone-iPad.

Este producto sin embargo, no es claro competidor, debido a que no está enfocado a centros de llamadas como tales.

Oracle CRM On Demand

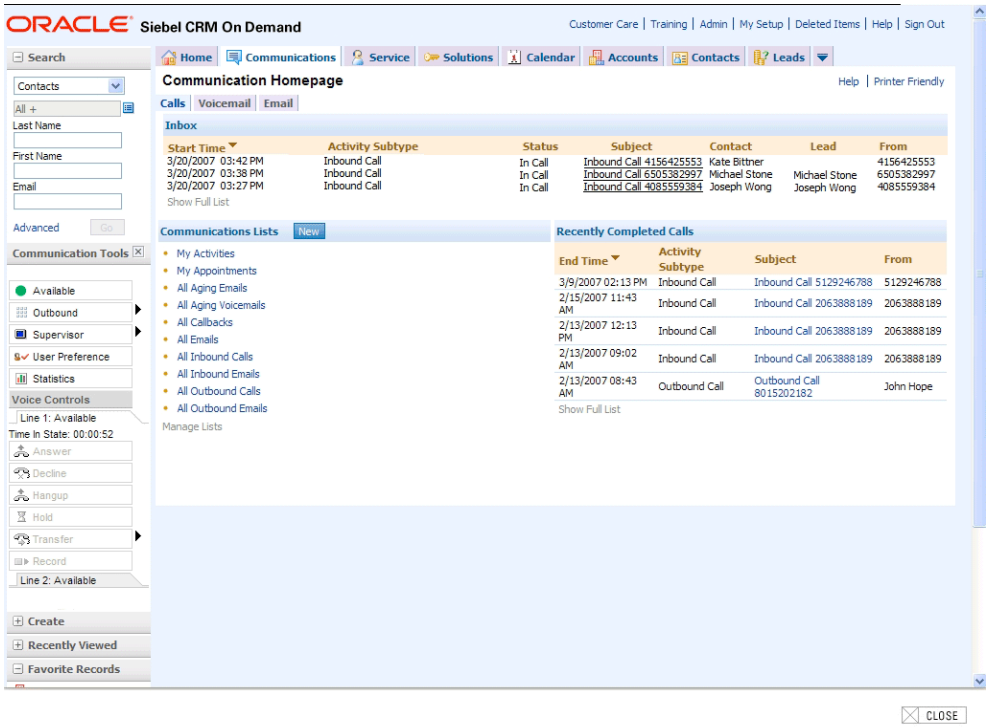


Ilustración 4 Oracle CRM On Demand

Esta alternativa [27] por otro lado, permite una mejora de productividad de los agentes de servicios, proveyendo una mejor experiencia de cliente. Los agentes pueden trabajar eficientemente desde cualquier lugar del mundo con todas las interacciones y necesidades gracias a su portabilidad a aplicaciones móviles como iPhone o iPad. Por otro lado, los clientes ganan en rapidez y ventas y servicio personalizados mientras que la empresa reduce costes y mejora las ventas.

Como queda claro en la descripción, esta aplicación está enfocada a *call centers*, pero el términos de venta al público. Por tanto, no es claro rival al igual que el anterior.

Conclusiones

Una vez mostrado las propuestas del mercado, queda claro que es un campo aún por explotar debido a la falta de competidores centrados en ello. A continuación, se explicará la plataforma elegida para hacer el proyecto y aún no siendo iguales en términos de funcionalidad, se mostrará ejemplos de aplicaciones con similitudes de diseño.

Plataforma seleccionada

Las elecciones eran claras: **Android**, **iOS** o **Blackberry**. Todas ellas, son propuestas factibles en términos de negocio. La última a pesar de su descenso en ventas y su supremacía por el control, sigue teniendo muchísimo interés en los ámbitos de negocio.

Sin embargo, viendo los interfaces de cada uno y su modo de visualización, ganó enteros el dúo iPhone-iPad. Primero, porque Apple siempre se ha caracterizado por su interés por la captación visual y segundo por la facilidad de portabilidad a cada uno de sus dispositivos. Tercero y último aunque no menos importante, porque en ocasiones el volumen de información en una sola pantalla y la posibilidad de visualizar los detalles es esencial. Por ello es necesario tener disponible la aplicación para tablets y es ahí, donde iPad controla el mercado de manera apabullante teniendo un 68% del mercado de las tablets [28].

Aplicaciones similares en diseño

Como se mostrará en el apartado de **Diseño**, la aplicación empleará una manera de visualización basada en **Tab Bar** [30]. Aquí, se indicará aplicaciones conocidas que hacen uso de dicho modo de visualización.

Ilustración 5- Aplicaciones que hacen uso de TabBar

En la imagen superior se puede echar un vistazo a aplicaciones que hacen uso de lo mencionado: Instagram [31], Foursquare [32], SoundCloud [33], Shazam [34] y Barclays [35].

Aunque todas ellas disponen de una barra inferior donde poder acceder a diferentes vistas (se explicará en el capítulo sucesivo) algunas muestran pequeñas diferencias, ya sea por un tamaño central mayor o por la forma geométrica de alguno de sus botones. De cualquier manera, su diseño como he dicho, varía de manera ínfima.

En definitiva, nuestra aplicación se unirá al nutrido grupo de aplicaciones que hacen uso de una barra inferior.

Introducción

Este proyecto toma como herramienta la arquitectura de iPhone OS para ofrecer una serie de facilidades a ciertos perfiles de agentes. Para ello, como se ha mencionado anteriormente, se ha escogido la aplicación iPad para dar soporte.

El diseño de este proyecto, al ser ofrecido a un cliente, debe tener ciertas pautas o criterios establecidos en reuniones y, expuestos a continuación:

- Atractivo visualmente: Es lo primero que ofreces. Debe provocar cierto interés.
- Facilidad de uso.: Toda aplicación debe regirse por este dogma.
- Adaptación a cada usuario: La aplicación debe adaptarse de forma automática al perfil del responsable.
- Rápida: La aplicación puede ser atractiva, fácil etc. Si es lenta, el interés se reduce y con ello puedes perder a un cliente potencial.

Una vez instaurados los criterios, se debe comenzar a trabajar. Para ello, empezaremos primero por la parte de diseño estrictamente hablando, se comentarán las diferentes posibilidades que se barajaron y después se explicará el por qué de la elección escogida.

Posteriormente, se ahondará en la arquitectura del sistema, se explicará el funcionamiento de éste y es aquí, cuando se introducirá el concepto de implementación en el que uniremos todo lo hablado anteriormente enfocado a esa idea.

Ilustración 6- iPad

Como se puede observar en la Ilustración anterior, este dispositivo, es el iPad, con unas medidas de 24,2824 cm(9.56 pulgadas) x 18,9738 cm(7.47 pulgadas) x 1,27 cm (0,5 pulgadas) . Después de presentar al protagonista de este proyecto, prosigamos.

Para trabajar y desarrollar sobre iPad, lo primero que debes hacer es bajarte e instalarte el SDK. En él, se incluyen las herramientas necesarias para conseguir tu fin: Xcode, Interface Builder, Instruments, iPhone&iPad Simulator.

-Xcode: es el entorno de desarrollo en sí. Incluye una colección de compiladores de proyecto GNU y puede compilar desde código C, C++ hasta Java, AppleScript y Objective-C que es, en nuestro caso, el utilizado.

-Interface Builder: herramienta gráfica para la creación de interfaces de usuario. Puedes agarrar diferentes vistas o ventanas y conectarlas mediante acciones de tal modo que, pueden interactuar con el código.

-Instruments: herramienta para analizar y visualizar todo lo ocurrido en la aplicación. Muestra para ello, una línea de tiempo con todos los eventos: actividad

de red, actividad de CPU y memoria. Con esto se consigue detectar fallos de memoria o problemas ocurridos en tu aplicación que de otro modo, sería casi imposible.

- iPhone&iPad Simulator: todas las pruebas se realizan sobre los simuladores de iPhone y iPad suministrados. Otra alternativa factible es sobre los dispositivos reales. Aunque, por rapidez siempre se prueba en el simulador.

Pero, ¿qué es el *SDK* de iPhone?. El *SDK* de iPhone consiste en una serie de *frameworks* que no son otra cosa que, librerías de software que proveen de ciertas y específicas funcionalidades. Para nuestro proyecto como es obvio, se han hecho uso de diferentes *frameworks*:

- Foundation.: es el principal *framework*, la cual provee de las clases necesarias de objetos tal como *NSObject* así como tipos básicos de datos, operaciones y un largo etc.

- Appkit: todo lo relacionado al interfaz de usuario está en este *framework*. Ventanas, botones, eventos o campos de texto.

- Core Data: este *framework* hace fácil el guardar objetos a un archivo y después cargarlos en memoria.

- QuartzCore: provee de la habilidad de configurar animaciones y efectos.

- CFNetwork: permite el acceso a servicios de red y sus configuraciones, tal como http o FTP entre otros.

Una vez bajado, instalado y parcialmente comprendido, llegó la pregunta temida: ¿y ahora, qué?

Pues bien, como he nombrado anteriormente en diferentes apartados, queremos ofrecer al cliente la capacidad para supervisar agentes. Para ello, desarrollaremos una aplicación que una vez conectada a un *webservice* (esto quedará extensamente explicado en sucesivas páginas), se descargará un contingente de datos, entre ellos unos favoritos con determinadas peticiones de modo de visualización. Una vez aclarado esto, se pintarán y es aquí, dónde entra el diseño en juego.

¿Qué hacer con esas gráficas? La respuesta es clara, facilitar su visualización. De tal modo que, se ofrecerá diferentes vistas acorde al gusto de cada usuario. Estas vistas estarán dentro de la aplicación para que el usuario pertinente, pueda navegar entre ellas. Para resolver este objetivo establecido, se estudió el SDK de iPhone para descubrir qué vistas puede ofrecer.

El SDK de iPhone viene previsto de cuatro tipos de aplicaciones visuales:

-*View-based Application*
-*Window-based Application*
-*Page View Controller*
-*Split View-based Application*
-*The Tab Bar Application*

A la hora de comenzar a visualizar en la mente este proyecto, las alternativas en cuanto a ponerlo en marcha eran las dos últimas mencionadas: *Split View-based Application* y *The Tab Bar Application*. De cualquier manera, explicaré brevemente cada vista.

Antes de sumergirnos en estas dos alternativas, hago un alto para esclarecer un punto que de otra manera, provocaría dudas posteriormente: Toda aplicación diseñada dispone de dos elementos: vista y controlador. Por un lado, la vista es la representación en sí de la interfaz y por otro, el controlador se encarga de realizar diferentes acciones asociadas a una vista determinada. Con todo esto, proseguimos con el proyecto.

Views

View-based Application

Ilustración 7-View-based Application

Como se puede observar en la ilustración 7, en este tipo de aplicación se dispone de una única vista. Esta vista es controlada por una clase de *View Controller*.

Debido a su simplicidad, se proseguirá con las demás aplicaciones visuales.

Window-based Application

Este tipo de aplicación a diferencia de la anterior, no incluye una *View Controller* pero sin embargo, sí que provee del esqueleto de la aplicación en el cual como desarrollador se deberá añadir las vistas necesarias y sus controladores respectivos.

Debido a lo anterior, ***Window-based Application*** es una gran y recomendada manera de comprender los controladores de vistas.

PageView Controller

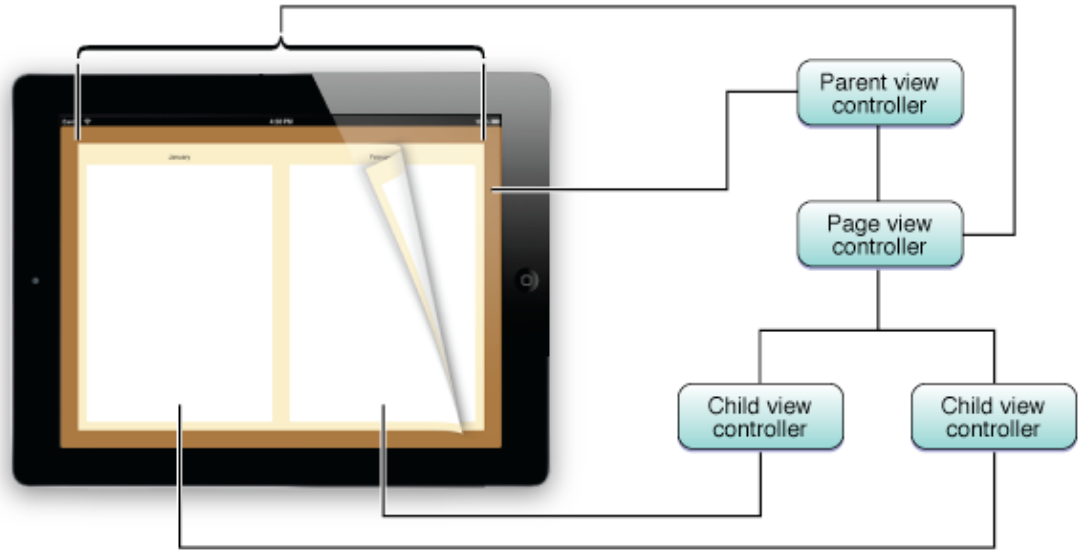


Ilustración 8-Page View Controller

En la ilustración 8, se puede observar la vista del interfaz *Page View*. Este tipo de controladores dispone de una única vista en cuál se almacena todo el contenido. La gran diferencia entre esta vista y las demás se produce cuando se pasa de página, haciendo ese mismo efecto al igual que si pasaras una página de cualquier libro.

El interfaz de este tipo consiste en:

- Un *delegate* opcional.
- Un data *source* opcional.
- Un *array* compuesto de los *view controllers*.

- Un *array* compuesto de los gestos reconocidos (esto permite pasar la página mediante volteo o agarre por ejemplo)

Y a continuación esta vista junto con sus objetos asociados:

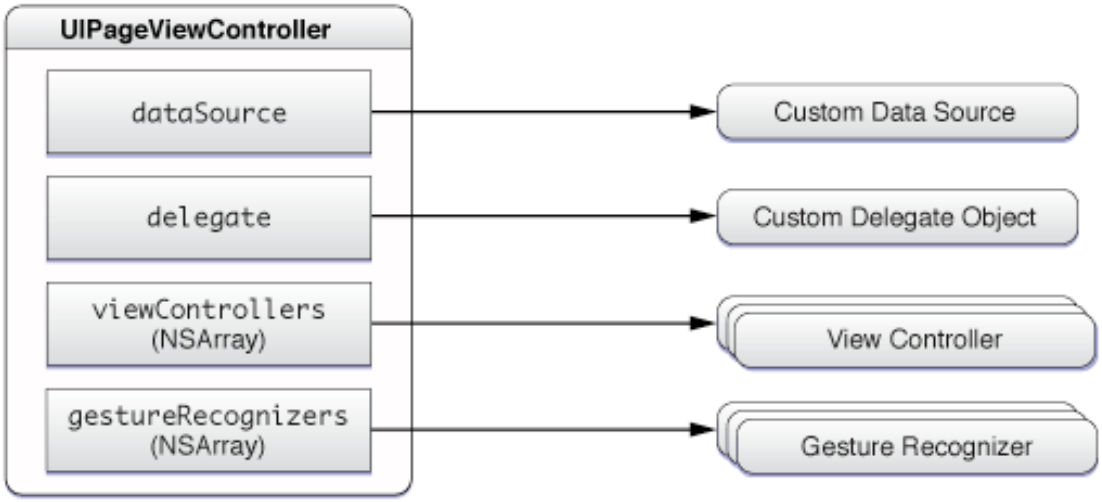


Ilustración 9-Page View Controller junto con sus objetos asociados

Tab Bar Application

Ilustración 10-Tab Bar Application

Esta clase contiene una colección de *View Controllers*.

En la ilustración 10 se puede observar dos vistas y, pulsando entre ellas, permite navegar entre diferentes vistas dentro de nuestra aplicación. Este tipo de vista se hace especialmente útil en situaciones donde se quiere ofrecer diferentes perspectivas sobre un determinado conjunto de datos. Para nuestro proyecto, que precisamente queremos ofrecer eso, este tipo de visualización toma un gran protagonismo.

Pero, ¿cómo funciona?

Ilustración 11-Interfaz de una Tab Bar

En la Ilustración superior se puede observar, las diferentes partes de una ***Tab Bar*** siendo, su interfaz compuesta de:

- Un objeto del tipo *UITabBarController*.
- Un *view controller* para cada *Tab*.
- Un objeto *delegate* opcional.

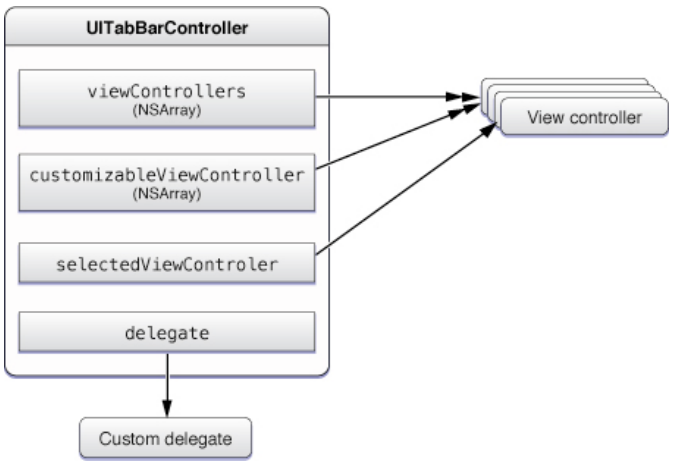


Ilustración 12-Tab Bar Controller asociado a sus view controllers

Un detalle importante es la separación entre ***Tab Bar*** y *Controller*. Éste último es el encargado de gestionar a la primera y decir qué vista se debe mostrar en cada momento mientras que el primero es el elemento de interacción.

Esta opción sin embargo necesita de un complementario: ***Nav Bar***

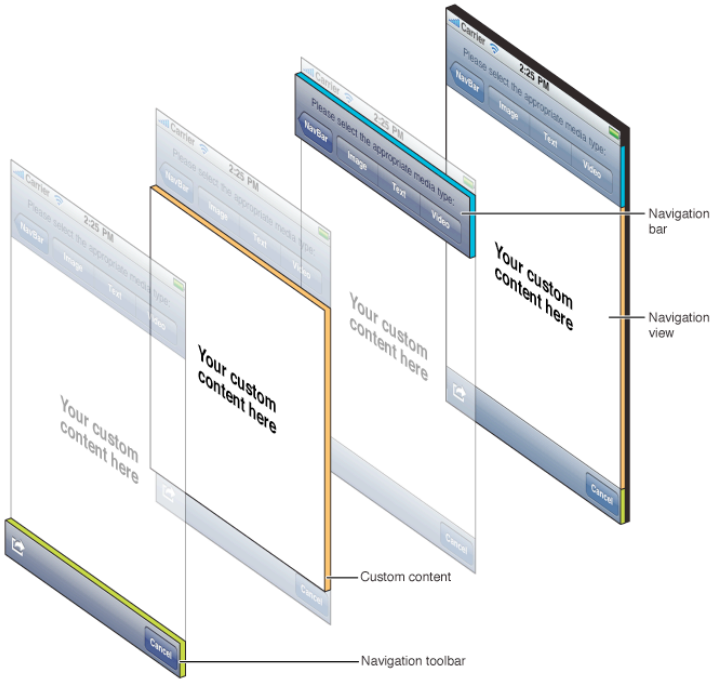


Ilustración 13-Nav Bar

En la Ilustración anterior se observa un ***Nav Bar***. Una vez que el usuario se encuentra en una vista que llama a otra a su vez, para poder navegar entre ellas, se hace necesaria esta herramienta. Su interfaz correspondiente:

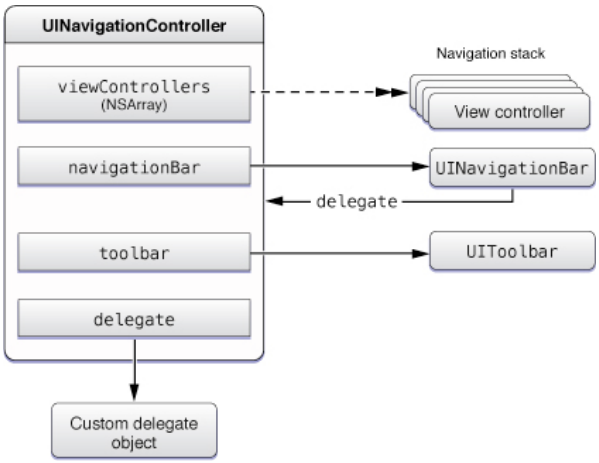


Ilustración 14-Objetos controlados por una Nav Bar

Las barras de navegación (Nav Bar) funcionan de manera diferente al basarse en jerarquías. Se indica cuál es la vista más alta en dicha jerarquía y el controlador de la barra indica que esa vista sea cargada dentro de ella. Esto funciona de manera sucesiva y se almacenada en una pila de la barra de navegación para poder moverse entre ellas.

Split View-based Application

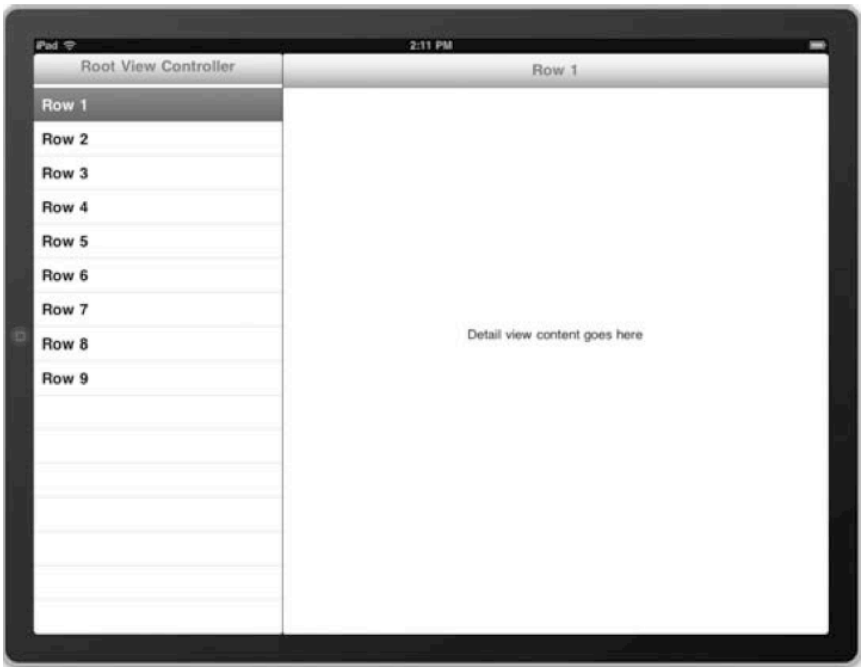


Ilustración 15-Split View landscape

Ilustración 16-Split View portrait

En la ilustración 15-16, se puede observar la vista ***SplitView***. La gran funcionalidad de esta vista se produce cuando se gira el dispositivo. Cuando se encuentra en posición *landscape* (ilustración 15) la aplicación muestra una lista de filas a su izquierda y sin embargo, cuando esta en posición *portrait* (ilustración 16) la lista de filas se encuentra recogida en un ***Popover View*** (esto se explicará antes de comenzar el siguiente apartado).

Otra manera más clara de visualización de lo explicado:

Ilustración 17-Split View

Cabe mencionar que, este tipo de vista siempre debe ser la raíz de cualquier interfaz que se cree. En otras palabras, siempre debes meter las vistas desde un objeto del tipo *UISplitViewController* como la raíz de la ventana de tu aplicación.

Antes de continuar, expliquemos un poco la vista *Popover.*

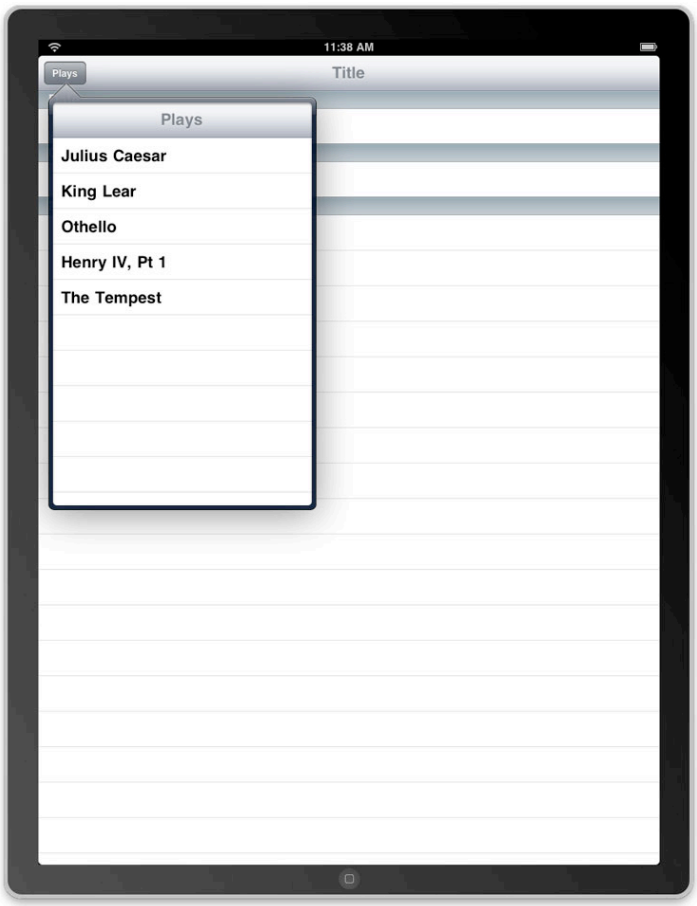


Ilustración 18-Popover

Cuando mencione todos los tipos de vistas que ofrece el SDK de iPhone no cité los **Popovers** debido a que no son una vista en sí, aunque eso no evita que pueda manejar la presentación de vistas de controladores. Este tipo de controlador, facilita la presentación de información mostrando una ventanita flotante en la ventana de tu aplicación. Las situaciones donde se puede hacer uso de este tipo de visualización sin dispares aunque, para este proyecto, la única que nos importa es cuando se usa en conjunto con el **SplitView**.

Como resumen de las vistas mencionadas:

Ilustración 19-Resumen de las vistas de controladores

Elección

Como se citó anteriormente, la elección sobre qué vista escoger se cerraba en torno a **Split View** y **Tab Bar** (junto con **Nav Bar**).

Por un lado, **Tab Bar** ofrece la mejor alternativa a la hora de manejar interfaces sobre dispositivos con una pantalla limitada en cuanto a espacio como puede ser el iPhone o el iPod touch.

Paralelamente a esto, la aparición en el mercado del iPad aportó entre otras cosas, un mayor espacio al iOS, por lo que nuevas UI aparecieron en respuesta. **Split View**, es una de ellas.

Ilustración 20-Ejemplo de Nav Bar y Split View

En la ilustración anterior se puede ver la diferencia principal entre estas dos vistas. **Split View** no contiene navegación hacia otras ventanas. Cuando un ítem es pulsado, sus datos aparecen en la misma pantalla usando un área específico. El iPhone, por otro lado, debido a su falta de espacio, presenta la alternativa de poder navegar entre vistas.

Pero, no sólo quedó relegado **Split View** para iPad únicamente. Muchos desarrolladores empezaron a trabajar también para adaptarlo a sus "hermanos" menores.

Ilustración 21-La navegación puede ser accedida por cada uno de los lados

Esto produjo que aplicaciones que comenzaron usando **Tab Bar** y **Nav Bar** se actualizarán a **Split View**.

Ilustración 22-Aplicación de Facebook en 2010 a la izquierda y en 2012 a la derecha.

Pero, ¿es posible esta adaptación para todas las aplicaciones?

Facebook como se muestra en la ilustración 22, es una aplicación bastante compleja, la cuál necesita acomodar cinco núcleos de secciones (News Feed, Messages, Nearby, Events and Friends) así como una cantidad indefinida de secciones secundarias (Pages, Apps, Groups, etc.). Teniendo un menú ayuda al usuario a obtener la funcionalidad que ellos esperan de una aplicación para móvil.

Por otro lado, hay otra multitud de aplicaciones más centradas en la experiencia que sacan un mayor beneficio de la **Tab Bar**. Y es que, teniendo una iconos centrales ayuda a reivindicar el núcleo de la función de la aplicación respectiva.

En comparación, dependiendo de cuál es nuestro objetivo con la aplicación, se debe escoger una u otra opción. Y posiblemente, **Tab Bar** es la mejor manera posible de centrar la atención del usuario.

Eso por un lado, por otro, al comienzo del desarrollo de la aplicación se habló de la portabilidad de ésta del iPad al iPhone en un futuro cercano. Y es que, viendo las posibilidades, no quedaba muy claro la visualización con **Split View** en iPhone debido a la gran cantidad de datos que podría haber en pantalla.

Concluyendo, en la reunión pertinente se acordó una vez vistas las alternativas que, la opción preferente a la hora de visualizar nuestra aplicación sería **Tab Bar**, tanto para el desarrollo del que se hablará en este proyecto, es decir para iPad, como para el futuro, iPhone.

Introducción

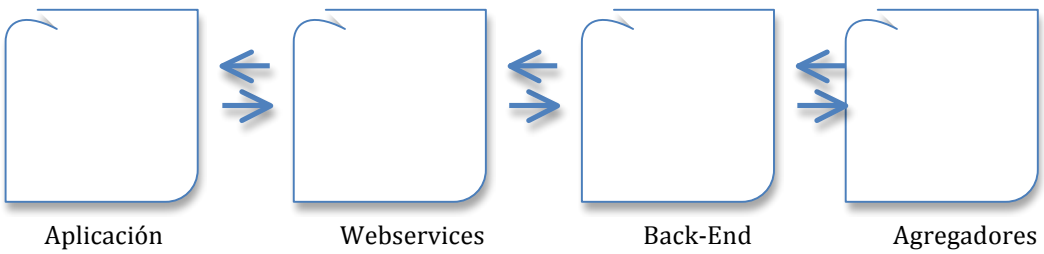
El objetivo de la aplicación es monitorizar el estado de unos agentes y de los recursos del sistema.

Esto permite ver qué hacen en su jornada los agentes de *call centers* para poder saber si hacen bien su trabajo o no. Haciendo esto, puedes optimizar la configuración del sistema por si hay muchos agentes libres por ejemplo. Controlando los recursos del sistema, puedes comprobar si hay pocos agentes atendiendo un determinado negocio, comprobar los tiempos de atención y el nivel de servicio para ver la calidad de tu servicio de atención por si la gente espera mucho en cola, por poner unos ejemplos. En todo esto, se engloba la aplicación desarrollada. Es decir, en agentes por un lado y estadísticas por otro.

En definitiva, los datos mostrados en este proyecto serán por un lado los agentes de *call centers* que estén usando un determinado recurso y el control de dichos recursos.

Arquitectura

Para entender lo que explicaré a continuación se observará esta ilustración:



Conectores

En esta figura se resume en líneas generales cómo funciona la aplicación. La aplicación que he creado se comunica con unos *webservices*, y éstos con los sistemas de back-end de la plataforma.

Estos sistemas de back-end, a su vez, están recibiendo, en tiempo real, la información de supervisión de diferentes sistemas de atención de llamadas, cada uno de ellos podría estar soportado por diferentes tecnologías (Genesys, Avaya, Cisco, etc..). Una vez recibida, se normaliza, se estructura y se ofrece a los usuarios, controlando los permisos y niveles de responsabilidad de cada uno de ellos.

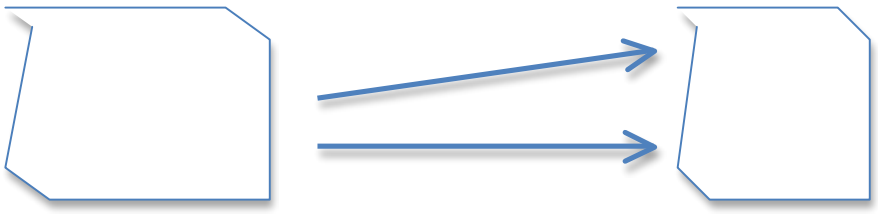
De esta forma, la plataforma de supervisión y monitorización, ofrece una capa de abstracción para independizar las infraestructuras y el equipamiento de los sistemas de atención de llamadas, y ofreciendo una visión global, estructurada de forma jerárquica en función de los negocios y roles de cada responsable, sin conocer la tipología de cada uno de los sistemas de atención de llamadas. Es decir, cada usuario ve a sus agentes y a sus centros pero no sabe si ese *call center* es Genesys, Avaya, Cisco, para él es transparente. Podría ver a dos agentes del que es responsable, uno de ellos en una plataforma y otro en otra y es totalmente transparente.

Básicamente, la arquitectura SW de la plataforma de monitorización y supervisión es la siguiente:

- Aplicaciones Conectores.
    - GconfigBridge: Conectores a la gestión de los sistemas de atención.
    - GSupbridge: Conectores a los sistemas CTI (AES Avaya, T-Server Genesys).
- Aplicaciones de Control de Estados.
    - Reciben los eventos de las anteriores, manteniendo es estado de actividad de cada agente y cada contacto.
    - Ejecutan las órdenes de monitorización: escucha silenciosa, modificación de estado, modificación de perfile de atención.
- Aplicaciones de Procesado de CDRs de actividad de los agentes
- Aplicaciones de Recogida y Estructuración de la información.

- Aplicaciones de Autentificación, Control de Acceso a la información y Control de Permisos.

De todos ellas, las más relevantes son las últimas dos para la realización de este proyecto, las cuales las podemos denominar también: CAGICC y CGUSU respectivamente. La capa de *webservice* que se usa desde la aplicación, redirige las peticiones a estos dos componentes. En el primer caso para la autenticación, validación de permisos de operación e información de recursos monitorizables. En el segundo caso, para la obtención de la información en tiempo real.



Y ahora, ¿qué es un *webservice*? *Webservice* es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como internet.

Los estándares empleados más comunes que nos interesan son:

- XML: (*Extensible Markup Language*): Es el formato estándar para los datos que se vayan a intercambiar.
- SOAP: (*Simple Object Access Protocol*) o XML-RPC (XML *Remote Procedure Call*): Protocolos sobre los que se establece el intercambio.
- WDSL:(*WebServices Description Language*): Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.

Para nuestro fin, nuestro *webservice* tiene varias interfaces que se deben mencionar, siendo no las únicas:

- SessionServices: permite autentificarte en la aplicación.
- AgentsSupervisionServices: detalla todos los agentes que se encuentran.
- SkillsManagementServices: especifica la habilidad de cada uno así como diferentes valores.

Monitorización

Introducción

En este apartado entraremos a ver cómo se recogen los datos y qué se hacen con ellos. Nuestro sistema **back-end** tiene los datos en Silverlight. Por lo que, un primer paso es procesarlos y transformarlos en el lenguaje que nuestra aplicación entiende: Objetive-C. Una vez hecho, hay tres partes diferenciadas:

- **Login**: es la parte de autentificación del usuario y acceso a los datos. Para ello se necesita el *webservice* SessionServices.
- **Favoritos**: aquí recogeremos estos mismos favoritos y explicaremos qué son. Para ello se necesita el *webservice* FavoritesManagementServices.
- **Representación**: se obtendrán el restante paquete de datos y se comenzará a representarlos. Para ello se necesita los *webservices*:
  - AgentsManagementServices: detalla todo los datos referentes a los agentes, es decir, nombre, apellidos, dni, teléfono, etc.
  - AgentsSupervisionServices: detalla el estado de los agentes.
  - SkillsManagementServices: detalla el puesto de cada agente.
  - StatisticsSupervisionServices: datos estadísticos obtenidos.

Login

Como se mencionó anteriormente, para poder autentificarnos en la aplicación, se necesita el *webservice* SessionServices. Por tanto, necesitamos el wsdl de ese *webservice* específico. En nuestro caso:

http://dirac/ImpacteBackEnd/SessionServices.svc?wsdl

¿Y ahora qué? Ahora, hacemos uso de una aplicación llamada WSDL2OBJC, el cual genera código en Objective-C procedente del WDSL para permitirte llamar a servicios SOAP.
Ejecutándolo:

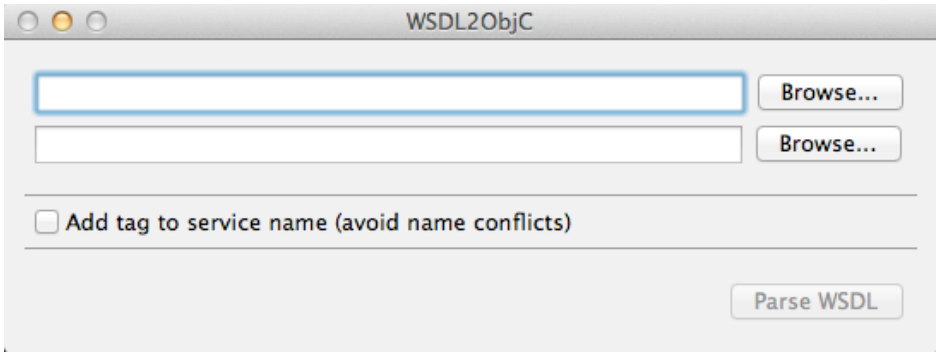


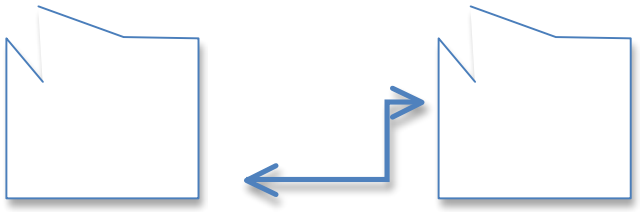
Ilustración 23-WSDL2OBJC

Simplemente, deberemos indicarle la dirección del WSDL. Podemos especificar un archivo local o una dirección web. En el segundo campo, especificamos dónde queremos que cree los nuevos archivos. Por último, dar a *Parse* WSDL para generarlos.

Una vez hecho, la aplicación nos devuelve gran cantidad de código destacando entre ellos SessionServices (ver Apéndice C). Dentro de dicho .m, nos encontramos con la función *LogOnRoot* al cual hay que invocar cuando introduzcamos el usuario y contraseña. Siempre será así para ese usuario en específico hasta que le cambien la contraseña. Como respuesta al *LogOnRoot* recibiremos un tsn1.SessionInfoItem con un campo llamado *Key*, siendo esto como una sesión de identificación,

cambiando cada vez que valides el usuario. Este identificador se lo deberemos pasar a todos los métodos que invoquemos en el *webservice*.

Después desde nuestra aplicación deberemos llamar a una función del SessionServicesWrapper (ver Apéndice C), siendo estos "Wrapper" librerías intermedias creadas para aislar diferentes tipos de clases del código. Estas librerías permiten que todo lo que reciba del *webservice*, la aplicación pueda comprenderlo. Es decir, es el intermediario entre la aplicación y el *webservice*.



En otras palabras, una vez generado los archivos procedentes del *webservice*, ya tengo el código en Objective-C por lo que puedo acceder a cualquier dato. Únicamente es necesario crear las funciones necesarias en la aplicación para poder recogerlo. En donde están todas esas funciones, son los wrappers. Y en vez, de mantenerlo todo en el código de la aplicación, las invoco como librerías. De tal manera, que la posible corrección o modificación se puede hacer de una manera más limpia. Para cada wsdl, hay una librería wrapper específica.

¿Qué quiere decir esto? Que las librerías wrappers hechas, son la parte del proyecto que traduce todo lo recibido del back-end y permite usarlo en la aplicación.

En este determinado caso, en la parte de **Login** , el wrapper SessionServicesWrapper, permite el acceso en el servidor a través del *webservice*, traduciendo su comunicación y abriendo los puntos de conexión necesarios para lograr la correcta autentificación del usuario pertinente.

Asentado eso, continuemos. Si ejecutamos la aplicación nos aparece una ventana de *login*:

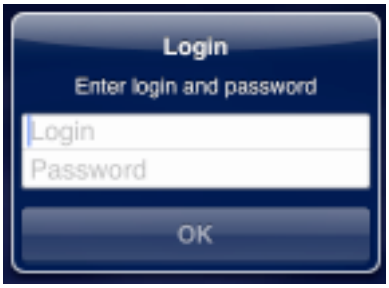


Ilustración 24-Pantalla de Login

Introduciendo el usuario y contraseña pertinente:

Ilustración 25-Pantalla de login

Si hemos conseguido autentificarnos satisfactoriamente accederemos a:

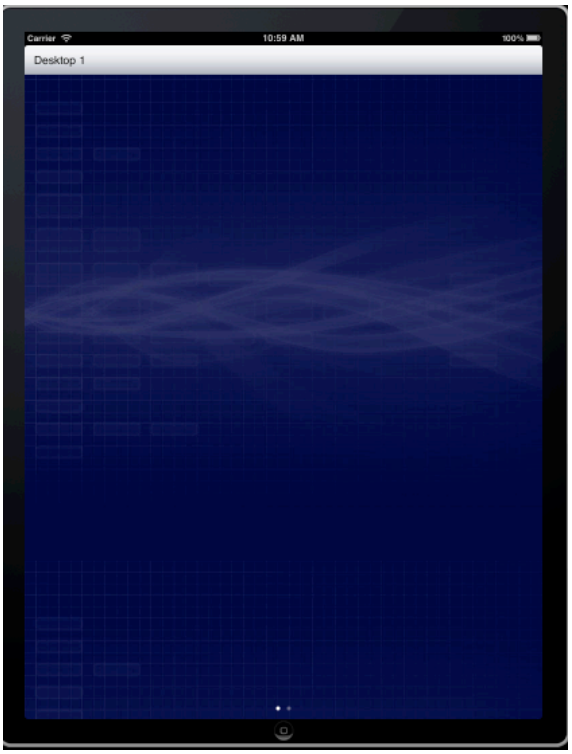


Ilustración 26 - Pantalla después de Login

Favoritos

Una vez, conseguido acceder es hora de recoger los favoritos, que serán los datos que tomaremos. Antes de continuar, explicaré qué son esos favoritos mencionados.

Los sistemas de back-end tienen una serie de recursos sobre los que se puede actuar (para gestionarlos, supervisarlos, consultarlos, etc.) sobre dos áreas distintas. Por un lado, está la estructura que se podría decir “física” con sus agentes, oficinas y centros. Por otro lado, las competencias que sería la estructura “lógica”, es decir, cómo el cliente quiere ordenar esos recursos físicos para una determinada función. Aquí se encuentran los negocios, modos de habilidad, etc. Para la supervisión se utiliza algo llamado "contenedor" donde se arrastra uno o varios de esos recursos, con unas determinadas operaciones y vistas, y eso da como resultado una "supervisión". Como este proceso es un poco farragoso, y normalmente un usuario acabará utilizando siempre los mismos "contenedores", se permite guardar esa configuración en la lista de favoritos para que pueda consultarla más tarde. Y, en el caso de iPad, es decir, en nuestro caso, sólo se permitirá consultar esa lista porque es lo que consultará más a menudo. Y esa lista, es lo que denominaremos, favoritos.

Para obtenerlos, se haría igual que antes, con la dirección del wsdl *webServices* de FavoritesManagementServices, lo parcheríamos obteniendo un conjunto de archivos y con nuestro, en este caso: FavoritesManagementServicesWrapper (ver Apéndice C) conseguimos dicho fin. Es decir, en este caso, permite la comunicación con el servidor abriendo los puntos de conexión necesitados y obteniendo los favoritos mencionados.

Ejecutando la aplicación obtenemos:

Favorito	Resource ID	Resource Type	Gráficos	Operaciones
Estad.Autónomos	1	18	3	7,3
Modo Otros	49	20	4	3
Agente Autónomos	1	18		7
Est y agentes	1	479	5	3,7
Hist estados	1	18	8	4
Barra acum	1	18	6	3
Colum acum	1	18	7	3
%BarrasPorEstado	1	18	1	2
%BarrasPorRecurso	1	18	2	2
Prueba Estados	87	304		7

Ilustración 27 - Tabla de favoritos

Después de obtenerlos, vemos acompañados de los nombres de los favoritos, diferentes términos nuevos: Resource Id, Resource Type, Gráficos y Operaciones junto con valores que difieren casi por cada favorito. Para saber qué significan dichos valores, hay librería para facilitar esta identificación de valores.

Para **Resource Type** tenemos que:

Resource Type	Valor
None	0
GenBusiness (negocio)	18
GenMode(modo)	20
GenSkill (habilidad)	1
GenProvider(proveedor)	479
GenOffice(oficina)	303
GenUnit(unidad)	299
GenAgentsFamily(familia agentes)	304
GenAgent(agente)	305
GenExtension(extensión)	315
GenVirtualGroup(grupo virtual)	38

Ilustración 28 - Tabla de Resource Type

Por tanto, si relacionamos nuestros **favoritos** junto con el **ResourceType** sabemos que:

Favorito	ResourceType
Estado Autónomos	GenBusiness
Modo Otros	GenMode
Agentes Autónomos	GenBusiness
Est y agentes	GenProvider
Hist estados	GenBusiness
Barra acum	GenBusiness
Colum acum	GenBusiness
%BarrasPorEstado	GenBusiness
%BarrasPorRecurso	GenBusiness
PruebaEstados	GenAgentsFamily

Ilustración 29 - Tabla Favorito - Resource Type

Después, tenemos el término Gráfica. Aquí se define el tipo de gráfica:

Gráfica	Valor
Ninguno	0
BarrasEstadosAgentes	1
BarrasRecursos	2
ColumnasEstadosAgentes	3
ColumnasRecursos	4
Quesos	5
BarrasAcumuladas	6
ColumnasAcumuladas	7
Lineal	8

Ilustración 30 - Tabla Gráficas

Y Operaciones, el tipo de operación:

Operación	Valor
Porcentaje	2
ValorAbsoluto	3
HistoricoEstadisticasAgentes	4
InterseccionEstadisticasAgentes	5
SupervisionAgentes	7
InterseccionSupervisionAgentes	8
NivelServicio	10
HistoricoNivelServicio	11
EstadisticasLlamadas	13
HistoricoEstadisticasLlamadas	14

Ilustración 31 - Tabla Operación

Si se echa un vistazo en operación hay una del tipo: SupervisionAgentes y como su propio nombre indica, supervisará los agentes dados de una *resource* determinada. Éste tipo de operación no tiene gráfica. Por tanto, cuando aparezca, el valor de gráfica será nulo o lo que pasaba anteriormente, ningún valor saldrá acompañando al término aunque se puede dar el caso que, nos pidan un tipo determinado de gráfica junto con una supervisión de agentes.
Juntando todos los valores:

Favorito	Resource ID	Resource Type	Gráficos	Operaciones
Estad.Autónomos	1	GenBusiness	ColumnaEstadosAgentes	ValorAbsoluto y SupervisionAgentes
Modo Otros	49	GenMode	ColumnaRecursos	ValorAbsoluto
Agente Autónomos	1	GenBusiness		SupervisionAgentes
Est y agentes	1	GenProvider	Quesos	ValorAbsoluto y SupervisionAgentes
Hist estados	1	GenBusiness	Lineal	HistoricoEstadisticasAgentes
Barra acum	1	GenBusiness	BarrasAcumuladas	ValorAbsoluto
Colum acum	1	GenBusiness	ColumnasAcumuladas	ValorAbsoluto
%BarrasPorEstado	1	GenBusiness	BarrasEstadosAgentes	Porcentaje
%BarrasPorRecurso	1	GenBusiness	BarrasRecursos	Porcentaje
Prueba Estados	87	GenAgentsFamily		SupervisionAgentes

Resumiendo, ya tenemos nuestros favoritos y ya sabemos qué hacer con ellos. Para identificarlos en la aplicación haremos uso de imágenes predefinidas para diferenciar cuándo se trate de un tipo de gráfica determinado o cuándo se trate de supervisión de agentes. Dichas imágenes son:

Ilustración 32 – Gráfica con historic de operación

Ilustración 33 – Cualquier tipo de gráfica

Ilustración 34- Gráfica + SupervisiónAgentes

Ilustración 35- Supervisión de Agentes

Representación

Ya hemos conseguido autentificarnos satisfactoriamente en la aplicación, hemos recogido también los favoritos y ahora, es turno de repetir el mismo procedimiento de antes con la dirección wsdl pertinente para:

- AgentsManagementServices: detalla todo los datos referentes a los agentes, es decir, nombre, apellidos, dni, teléfono, etc.
- AgentsSupervisionServices: detalla el estado de los agentes.
- SkillsManagementServices: detalla el puesto de cada agente.
- StatisticsSupervisionServices: datos estadísticos.

Con todo ello se consigue tanto los agentes, como cada especialidad, como cualquier valor referente a ellos. Haremos uso de las librerías intermedias para facilitar su recogida:

- AgentsManagementServicesWrapper
- AgentsSupervisionServicesWrapper
- SkillsManagementServicesWrapper
- StatisticsSupervisionServicesWrapper

Haciendo uso de todos estos wrappers, conseguimos obtener cualquier dato necesario para la representación de las gráficas pedidas. De igual manera que se ha hecho anteriormente, se consigue dicho fin a través del *webservice* y abriendo los puntos de conexión y por último, obteniendo los datos.

Una vez tenido todo dato, es hora de dibujar las gráficas (la supervisión de agentes se explicará a continuación). Para ello, existe una librería abierta completa llamada Coreplot [36] que permite visualizar datos en 2D. Por lo cual, implementaremos dicha librería en nuestra aplicación y haremos uso de ella, para dibujar las gráficas pedidas.

Un ejemplo de visualización para aclarar esto:

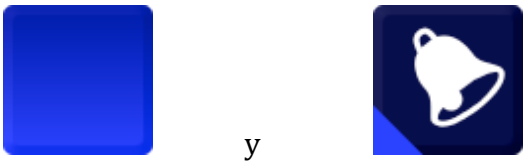
Ilustración 36-Coreplot

Supervisión Agentes

Para esta tarea, accediendo a la aplicación he creado:

Ilustración 37-Agent Info

En él se puede ver un icono predefinido de agente y cuatro iconos pequeños más. Los que nos importa en este caso son :



y

Ilustración 38 – Estado y sub-estado

El primero indica el estado actual del agente. Puede haber tres principales estados principales: libre, en pausa u ocupado. El segundo indica el sub-estado dentro del estado primero. A continuación, todos los tipos:

Libre

Este estado es el más sencillo. Únicamente tiene valor su estado actual, mientras que su sub-estado es nulo. En la pantalla saldría:

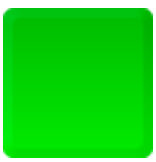


Ilustración 39 - Estado: libre

Pausa

Este estado sin embargo, ya tiene cierta complejidad. En el estado actual se observaría:

Ilustración 40- Estado: pausa

Y dentro de ese estado pueden suceder varias cosas:

- *Default*: pausa por defecto.
- *Scheduled*: pausa programada.
- *Other*: alguien ha forzado dicha pausa.
- *Training*: pausa debida a entrenamiento
- *Administrative*: pausa administrativa.
- *AgentInteractive*: el agente ha forzado la pausa.
- *Audits*: pausa debida a una auditoría.
- *Meeting*: pausa por reunión.

A continuación, los iconos de los sub-estados mencionados:

Ilustración 41- Subestado: default

Ilustración 42- Subestado: scheduled

Ilustración 43- Subestado: others

Ilustración 44- Subestado: training

Ilustración 45 –Subestado: administrative

Ilustración 46- Subestado: agentInteractive


Ilustración 47- Subestado: audits


Ilustración 48- Subestado: meeting

Ocupado

En este último estado posible, al igual que antes muchos sub-estados posibles pueden ocurrir. Por un lado, el estado actual sería:


Ilustración 49- Estado: busy

Y por otro lado, los sub-estados pueden ser:

- *Admin*: ocupado en trabajos administrativos.
- *Ring*: ocupado por tono de llamada.
- *Conversation*: ocupado por conversación.
- *Retention*: ocupado por pausa.
- *Consult*: ocupado debido a consulta.
- *Study*: ocupado debido a estudio.
- *AgentRetention*: ocupado forzado debido a agente.
- *ConferenceWithReatinedClient*: ocupado debido a conferencia con cliente retenido.
- *ClientAndAgentRetained*: ocupado debido a que tanto agente como cliente están en pausa forzada.
- *ConversationWithAgentRetained*: ocupado debido a que la conversación del agente está retenida.
- *Multiconference*: ocupado debido a multiconferencia.
- *Consulted*: ocupado debido a consulta

Y sus iconos respectivos:


Ilustración 50- Subestado: admin


Ilustración 51- Subestado: ring


Ilustración 52- Subestado: conversation


Ilustración 53- subestado: retention

Ilustración 54- Subestado: consult

Ilustración 55- subestado: study

Ilustración 56- Subestado: agentRetention

Ilustración 57- Subestado: conferenceWithRetainedClient

Ilustración 58 - Subestado:ClientAndAgentRetained

Ilustración 59: ConversationWithAgentRetained

Ilustración 60. Subestado: multiconference

Ilustración 61- Subestado: consulted

Otro caso que se podría dar que no hemos mencionado todavía es, ¿qué ocurre si no hay datos en los favoritos?

Nuestra aplicación debe darse cuenta y no pintar algo equivocado. Por tanto en nuestra pantalla al dar a un favorito "nulo" saldría:

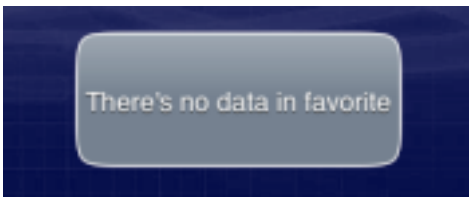


Ilustración 62- Pantalla de favoritos nulos

Ya tenemos por tanto, las gráficas pedidas y la supervisión de agentes con todas sus sub-clases.

En un primer *tab* , dibujaremos los favoritos sin ningún tipo de efecto. En los otros dos, daremos cierta sofisticación visual a nuestra aplicación. El siguiente apartado cubrirá esto. En definitiva, se hablará de una **Normal View**, **Launcher View** y **Coverflow View**.

Vistas

Normal View

Esta visualización será la primera al ejecutar la aplicación. En ella descargaremos los favoritos y los tiempos de carga, permitiendo una mayor fluidez después en las demás vistas. **Normal View** será la manera más completa de mostrar los favoritos, tanto de monitorización y supervisión de agentes, convirtiéndola en la protagonista de nuestra aplicación.

En un primer momento, se observará esto:

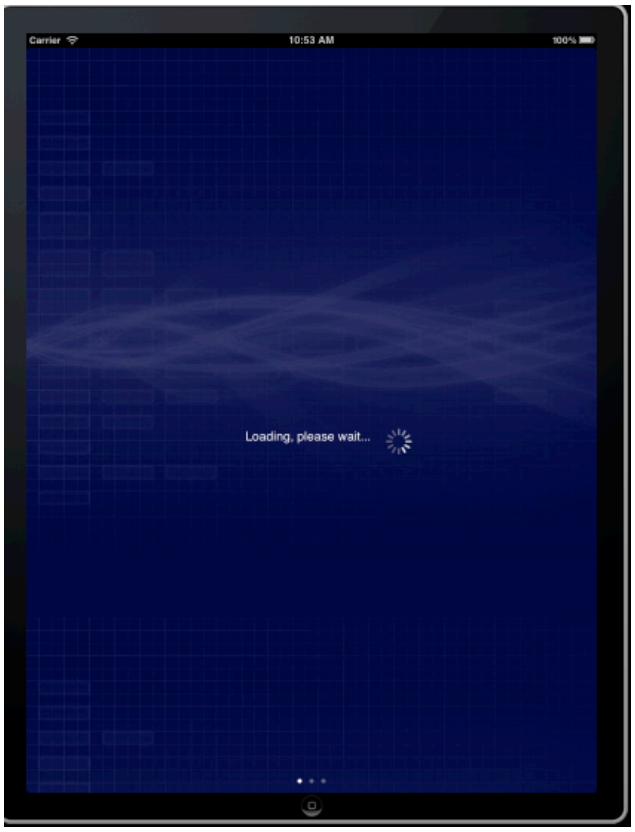


Ilustración 63 -Normal View

Esta vista indicará el tiempo de carga pertinente. Una vez se hayan recogido todos los datos necesarios, desaparecerá permitiendo acceder a la vista que hemos llamado **Normal View**.

La vista en sí:

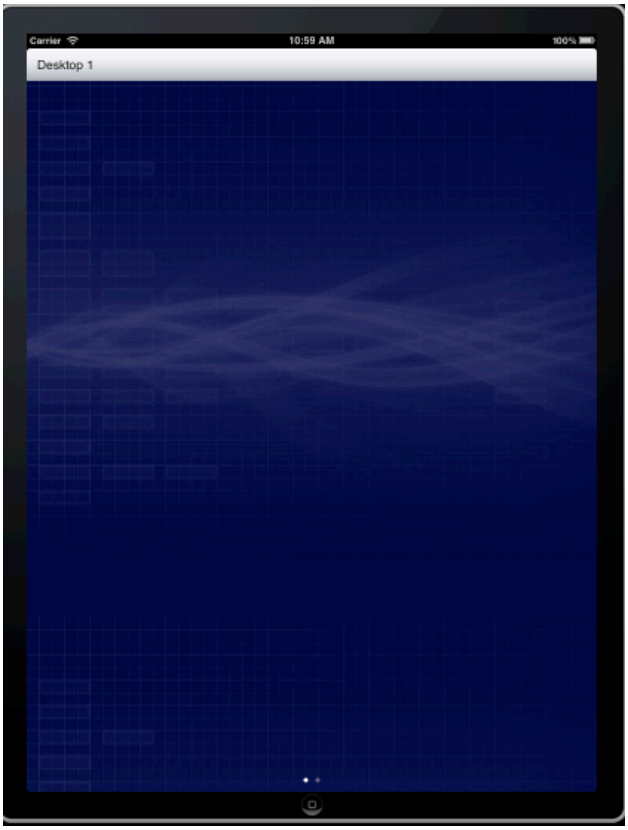


Ilustración 64 - Normal View2

En la figura anterior se puede apreciar un nuevo concepto: Desktop, acompañado de una numeración, en este caso: 1. Esto indica el número de escritorio en el que nos encontramos. Por defecto, habrá dos escritorios cada vez que ejecutemos la aplicación. Esto nos permitirá seleccionar un favorito en un primer escritorio y otro diferente en el siguiente escritorio, favoreciendo la visibilidad. Para acceder al otro *desktop*, basta con arrastrar el dedo para desplazarte de uno a otro.

Para contemplar los favoritos, se hará *click* sobre la barra superior donde se encuentra Desktop 1 (en nuestra figura).

Pulsando:

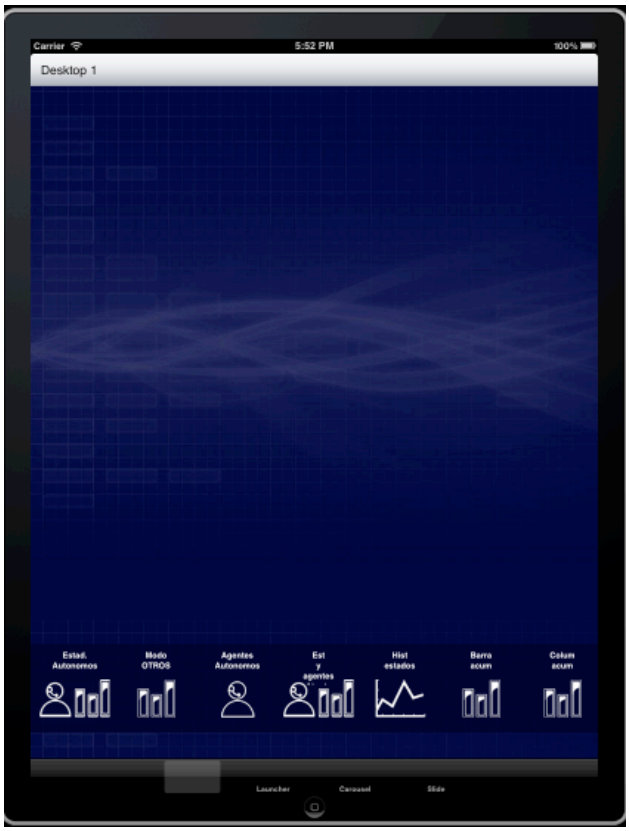


Ilustración 65 - Normal View3

En la imagen superior se aprecia la lista de favoritos que aparece al pulsar sobre la barra superior del *Desktop*. En un primer vistazo se aprecian los primeros siete favoritos. Bastaría con deslizar el dedo sobre la plataforma donde se sustentan los favoritos para hacer aparecer los restantes tres.

Para observar ya el favorito en sí, haciendo *click* sobre cualquiera, aparecería la oportuna supervisión de agentes, monitorización o ambas.

A continuación, se mostrará cada favorito al pulsarlo. Cabe recordar que para las gráficas hicimos uso de la librería **Coreplot** explicando anteriormente.

Ilustración 66 – Normal View4: Primeros dos favoritos

En la figura de la izquierda se puede ver el favorito titulado Estad. Autónomos que como averiguamos anteriormente, se pedía una gráfica del tipo Columnas Estados Agentes y operaciones de Valor Absoluto y Supervisión Agentes. Por otro lado, en la figura de la derecha era una gráfica del tipo Columna de Recursos.

Ilustración 67 – Normal View5: Tercer y Cuarto favorito

En la figura de la izquierda es el favorito titulado Agentes Autónomos y se pedía una Supervisión de Agentes y en la figura de la derecha era una Supervisión de Agentes junto con gráfica del tipo Quesos.

Antes de continuar mostrando los diferentes favoritos y su representación, voy a mostrar de nuevo el primer favorito para entender mejor qué se está mostrando en la gráfica.

Ilustración 68 – NormalView6: Primer Favorito

Las gráficas que estamos llevando a cabo representan la volumetría de lo que se esté midiendo. En nuestro ejemplo de arriba la gráfica es de estados de agente, por tanto, representará qué número de agentes se encuentran en cada estado. Es decir, cuántos estarán ocupados, cuántos libres, cuántos ocupados en otros servicios, etc.

Haciendo uso de nuevo de la librería Constans podemos ver mejor esos estados:

Estado	Valor
Registrados	0
LibSal	1
LibEnt	2
GestDiferidas	3
OcupSal	4
OcupEnt	5
OtraHabilidad	6
Pausa	5
LlamadasCola	8

Ilustración 69 - Tabla Estados

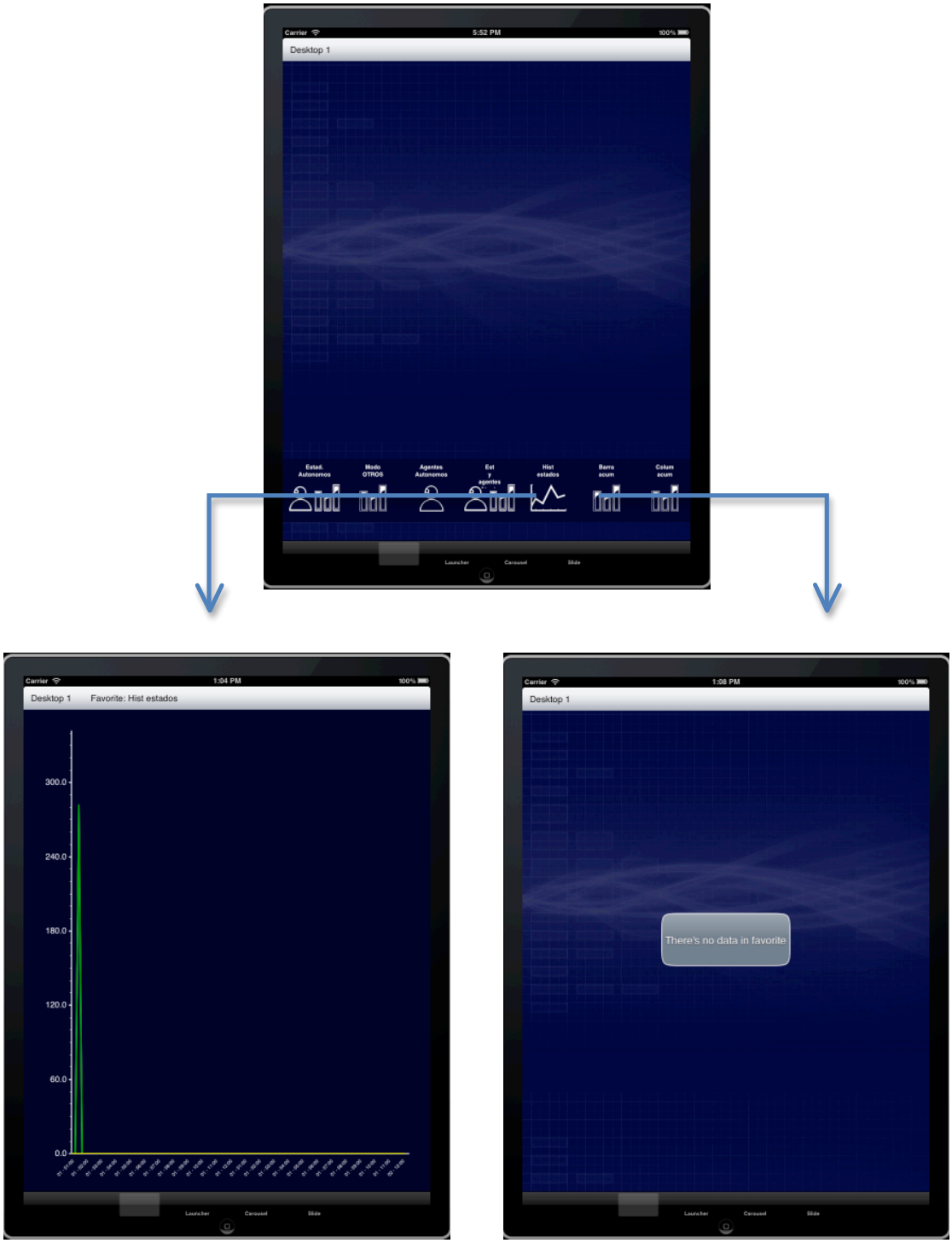


Ilustración 70 - NormalView7: Quinto y Sexto favorito

En este caso, tenemos un favorito del tipo Lineal y uno sin datos. Como se explicó anteriormente, aparecerá en pantalla un aviso indicando que no hay datos.

Ilustración 71 - Normal View8: Séptimo y Octavo favorito

En la figura de la derecha hay una gráfica con operación Porcentaje de Estados de Agentes. Y nos encontramos con otro favorito sin datos.

Ilustración 72 - Normal View9: Noveno y Décimo favorito

Por un lado, tenemos una gráfica del tipo Barra de Recursos con una operación de Porcentaje. Por otro, otra sin datos.

Con este último par de favoritos, terminamos **Normal View.** A continuación las siguientes vistas.

Launcher View

Esta vista como la posterior que contaré, cualquier usuario de iPhone o iPad es familiar sobre ella. Es la vista por defecto, donde se muestran todas las aplicaciones de las que se disponen con su representación por iconos. Una gran característica que tienen es que, si mantienes pulsado el dedo sobre cualquiera de ellas, puedes modificar su posición a voluntad junto con la posibilidad de eliminar cualquiera en la misma acción. Esta vista es francamente útil y fácil, aunque viene con un problema añadido: es de Apple.

Ser de Apple quiere decir sencillamente que, no estás autorizado a usarlo debido a que es una API privada. Por tanto, para este proyecto, se ha creado dicha vista teniendo claramente como objetivo su similitud, añadiendo también la función de posicionamiento y eliminación de este caso, nuestros favoritos.

Un primer acercamiento visual mostrando los primeros seis favoritos:

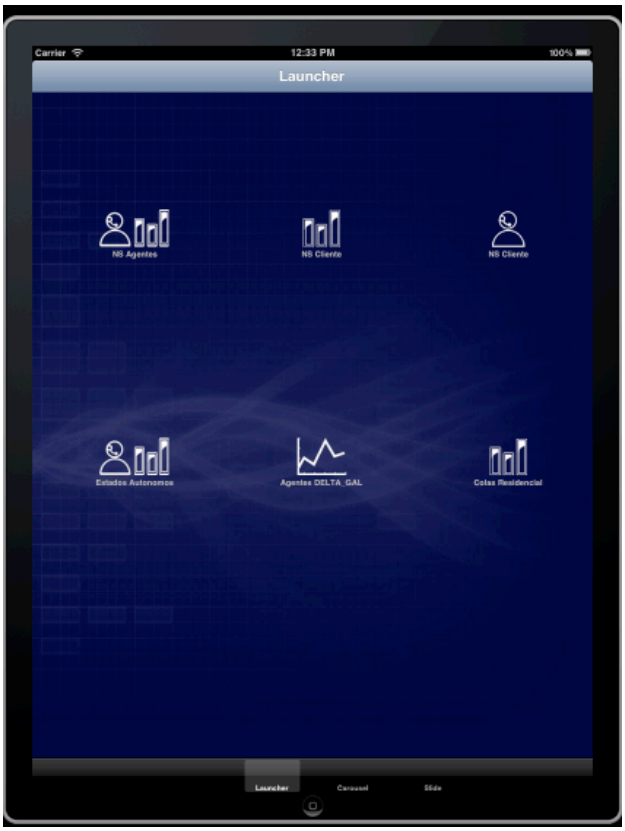


Ilustración 73- Launcher View

Y manteniendo pulsado un icono:

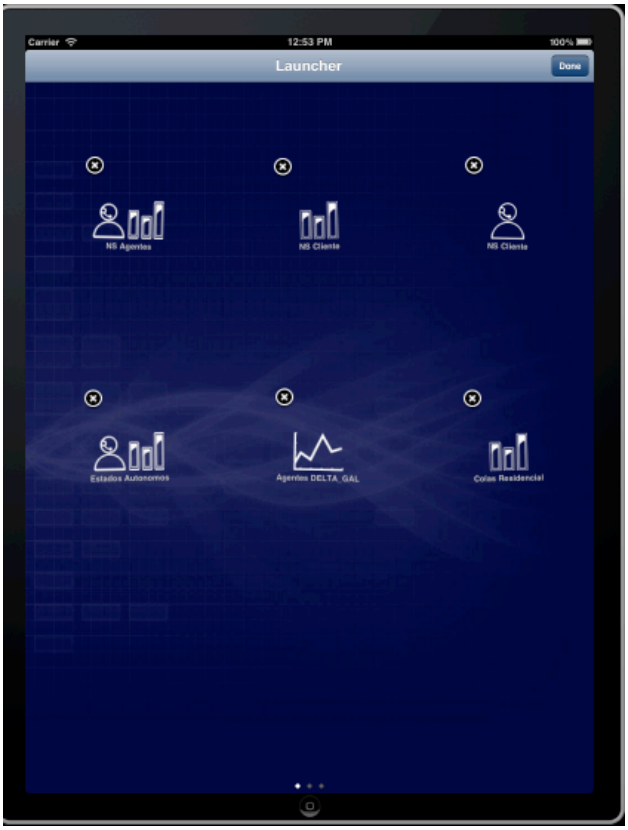


Ilustración 74 - Launcher View2

En la pantalla, aún no apreciándose el movimiento de los iconos, aparecen junto con los iconos unas 'x' las cuales nos permitirían eliminar cualquier favorito y en la *navBar* un botón de nombre "Done". Este susodicho botón permite terminar esta acción volviendo al estado anterior.

Por último, cuando se pulsa un icono que tenga gráfica, salta hacia otra vista, mostrándola.

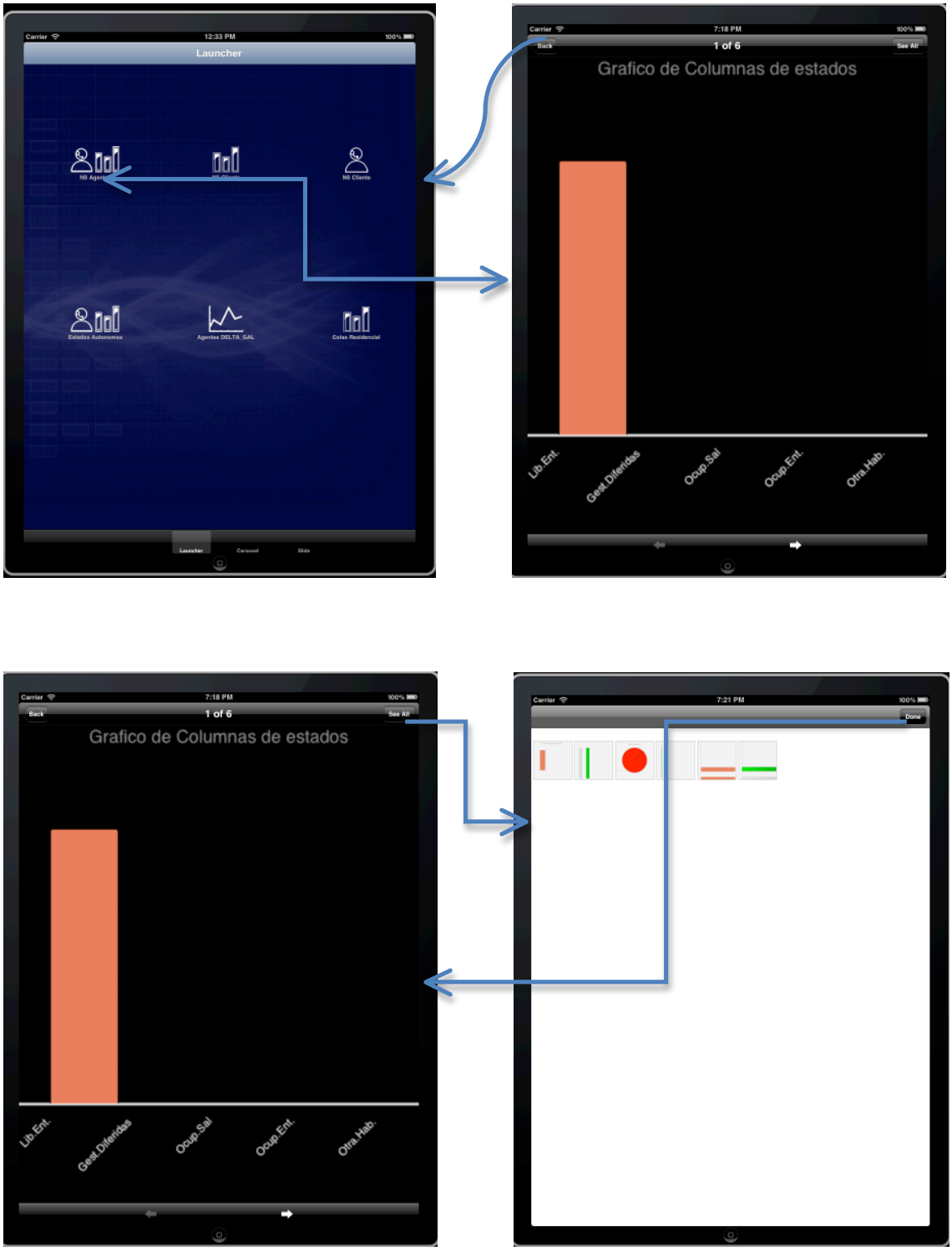


Ilustración 75 - LauncherView3

En la imagen superior se observa, dos botones en la barra de navegación superior. La primera es “back” que permite volver a la vista anterior y la segunda es “see all” la cual permite, ver todas las imágenes de las gráficas en pequeño, para poder seleccionarlas mejor, pudiendo volver a la vista anterior mediante un “done”. También es importante mencionar la flecha situada en la parte inferior de la imagen la cual facilita el desplazamiento de una imagen a otra pero, de cualquier manera, el desplazamiento con el dedo es posible también. Otra característica es el zoom. Al pulsar dos veces sobre la imagen ésta se amplia y con otros dos *clicks* se deshace. Valiéndome de la gráfica anterior, ejemplos de zoom serían:

Ilustración 76 - Launcher View zoom

Coverflow View

Este modo de visualización como el anterior, es también sumamente conocido. Aparece al poner el dispositivo en modo *landscape* cuando accedes a la música en la que aparecen todos los álbumes o autores de tal modo que puedas navegar entre ellos deslizando el dedo.

Ilustración 77 - Coverflow View

Este efecto fue añadido por primera vez al iPhone permitiendo visualizar las carátulas de los álbumes en vez del texto como se producía anteriormente. Funciona utilizando la barra de desplazamiento de la pantalla usando en nuestro caso, el dedo.

Lamentablemente, este API también es propiedad de Apple, por lo que la implementación de este efecto en nuestra aplicación se creó a partir de librerías abiertas y gran imaginación, siempre teniendo como objetivo el creado por ellos.

La principal diferencia en cuanto a otras librerías que hacen uso de este efecto, es que la nuestra puede trabajar con cualquier tipo de vistas, no se encierra en únicamente imágenes, por lo que se puede presentar los datos de una manera fluida. También, se incluyeron diferentes tipos de efectos, los cuales se pueda añadir o quitar con cambios mínimos en el código.

Un ejemplo de este efecto en nuestra aplicación :

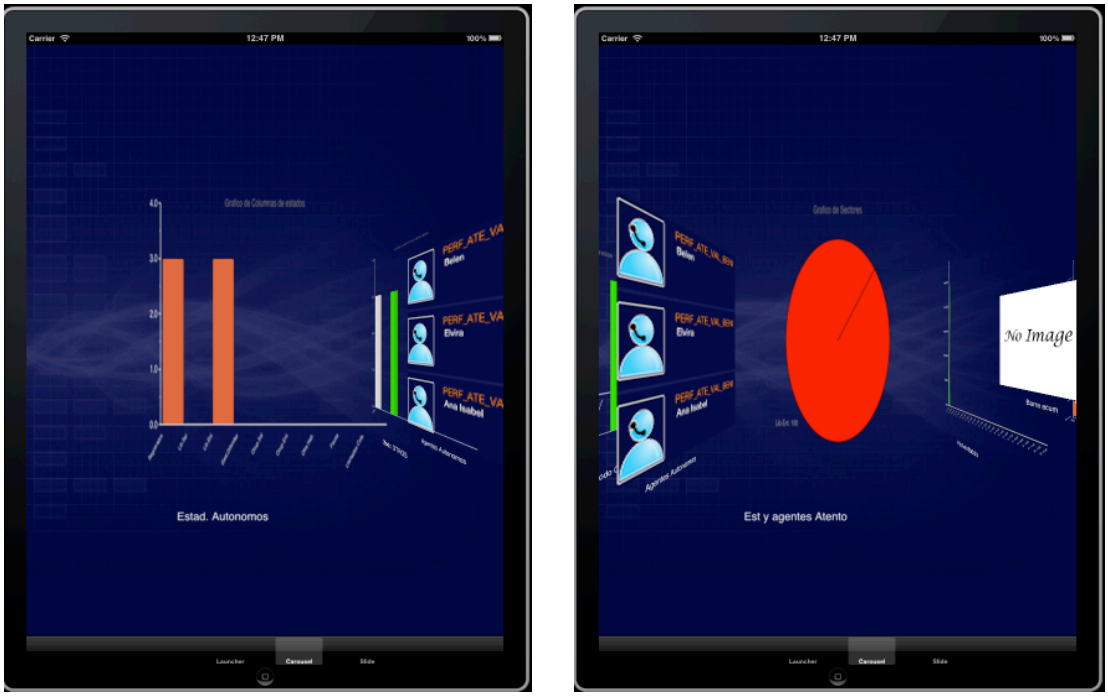


Ilustración 78 - Coverflow View2

En la figura superior se muestra dos situaciones de **Coverflow View** con las gráficas que hemos visto en las demás vistas. Como se ha mencionado, para desplazarse entre ellas, basta con deslizar el dedo para conseguir el movimiento fluido tan conocido.

Funcionalidad Añadida

Otra funcionalidad que hemos añadido es el poder desplazarse por cada vista sin la necesidad de usar la **Tab Bar** haciendo uso del reconocimiento de *clicks* en pantalla. Para variar respecto de lo visto, hemos incluido un carrusel de imagénes (**Slide View**) que se desplazan automáticamente sin la aparición de las flechas vista en **Launcher View.** Todo nace desde ésta última.

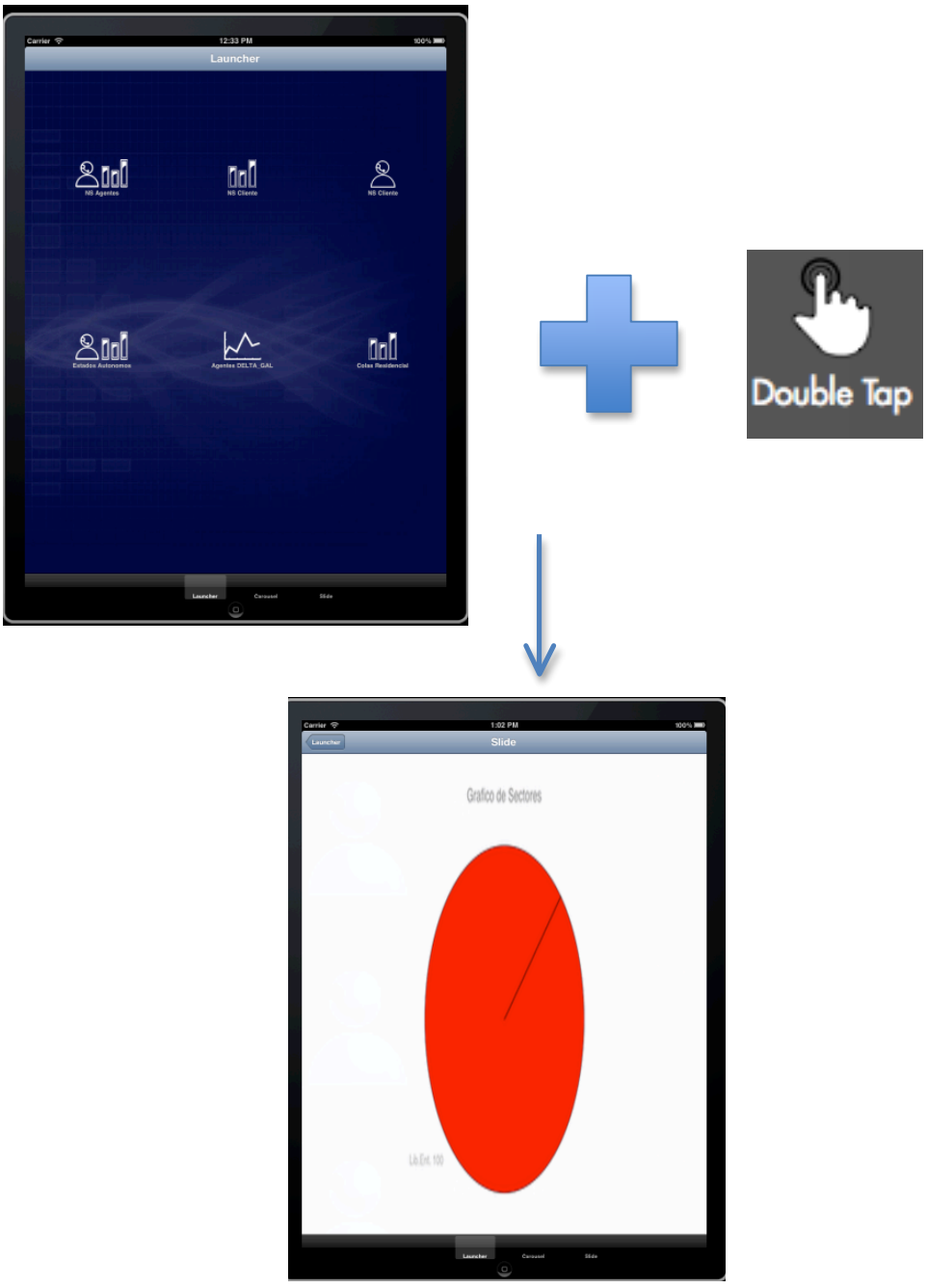


Ilustración 79- Double Tap

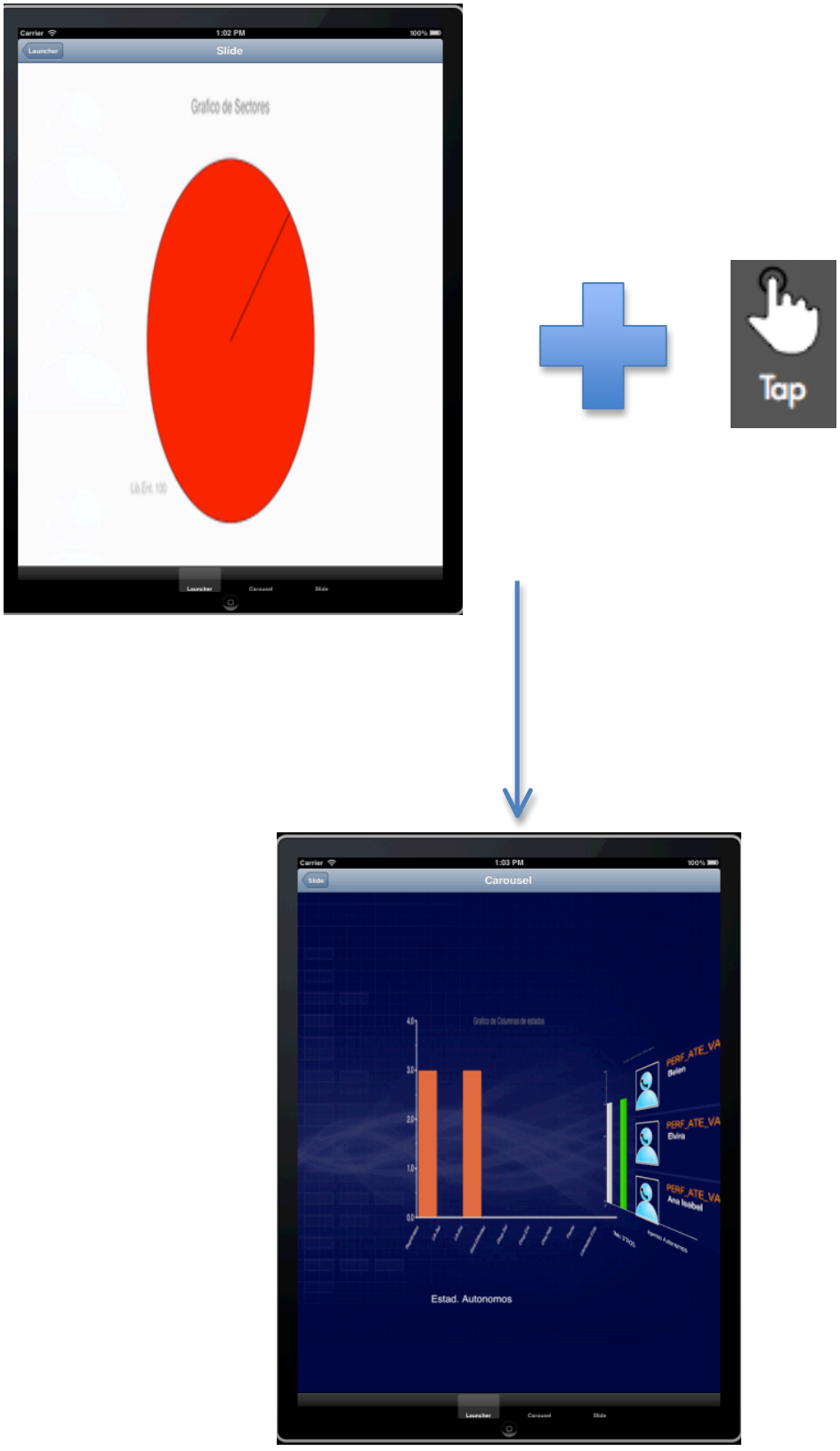


Ilustración 80 - Tap

En el apartado anterior se incluyó la parte de Integración de manera que fuera más fácil la comprensión de qué se está haciendo. Aquí, volveremos a recoger las gráficas y veremos los resultados obtenidos, partiendo del hecho de que como se mencionó al principio de la memoria, los datos los recogemos sobre un servidor el cual es nutrido por unos favoritos y agentes ficticios. Se hará uso de la vista **Normal View**.

Ilustración 81 - Favorito1

En esta gráfica se aprecian tres agentes: Belén, Elvira y Ana Isabel, los cuales están de estado: Libre. La gráfica de columnas de estados indica que están **libres** como indicaba la supervisión de agentes al igual que **registrados**. Por tanto, el valor de cada columna es tres, el equivalente al número de agentes.

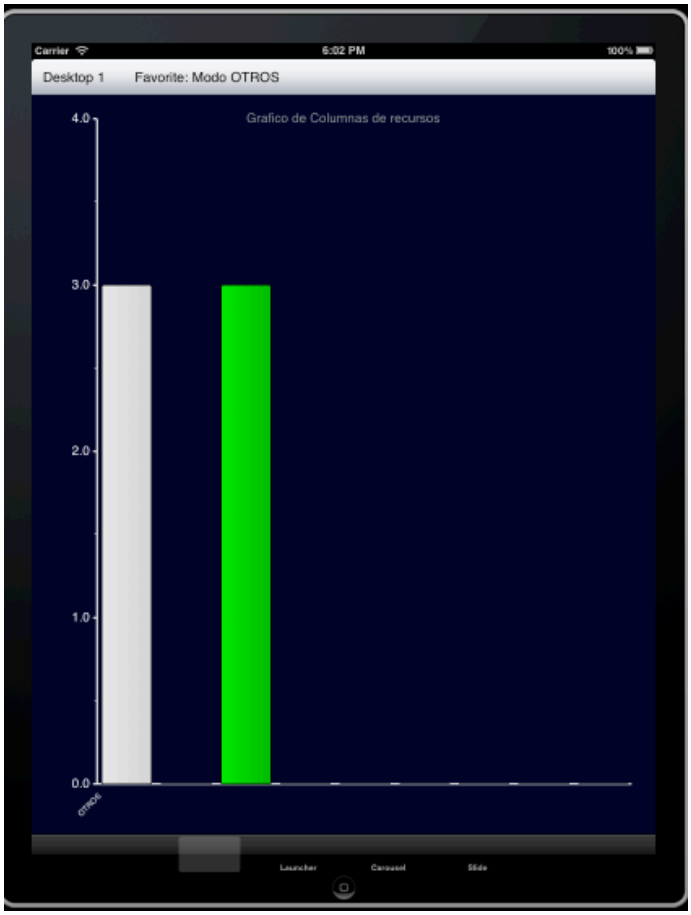


Ilustración 82 - Favorito 2

En este caso, tenemos el favorito titulado Modo OTROS. Es una gráfica de columnas de recursos con tipo de operación de valor absoluto. El número de agentes totales es tres igual que antes, con un ResourceName: OTROS. Este recurso dispone de dicho número de agentes registrados y libres.

Ilustración 83 - Favorito 3

Es el favorito Agentes Autónomos. Se puede apreciar que son tres: Belén, Elvira y Ana Isabel, los tres libres desde un determinado tiempo. Para Belén desde hace 06:13:49, para Elvira: 06:14:01, para Ana Isabel: 06:13:37. Al estar libres, no tienen ningún subestado.

Ilustración 84 - Favorito 4

En el favorito 4 llamado Est y Agentes disponemos de nuevo de los tres agentes citados pero esta vez, una representación de quesos, en el cual se aprecia un estado: Lib Ent. Éste podemos averiguarlo gracias a mi librería Constans tantas veces mencionada.

Estado	Valor
Registrados	0
LibSal	1
LibEnt	2
GestDiferidas	3
OcupSal	4
OcupEnt	5
OtraHabilidad	6
Pausa	5
LlamadasCola	8

Ilustración 85 - Tabla Estados

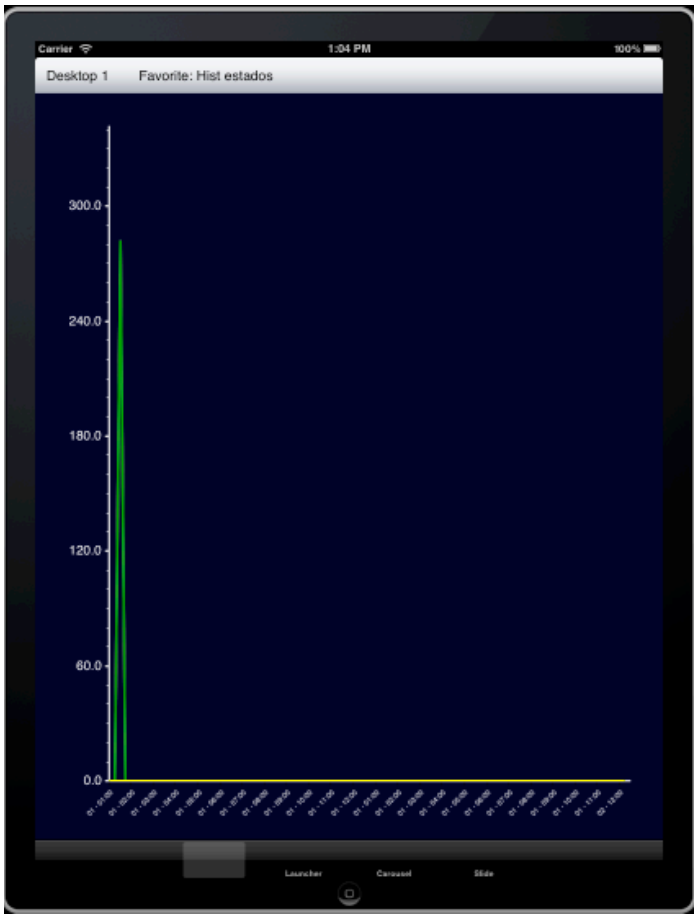


Ilustración 86 - Favorito 5

En esta gráfica se representan los valores históricos de los agentes y sus estados. Es decir, se toma como inicio un determinado tiempo hasta un tiempo final de un nodo específico, en este caso: **rusk** que es el nodo de nuestro webservice **dirac**.

Para el estado de agentes se hace uso de :

Estado	Valor
BlockedAgent	0
CorpAplicInputBusyAgent	1
CorpAplicOutputBusyAgent	2
InputBusyAgent	3
InputFreeAgent	4
OutputBusyAgent	5
OutputFreeAgent	6
PauseAgent	7
InputBusyAgentsOtherSkill	8
OutputBusyAgentsOtherSkill	9

Ilustración 87 - Tabla Estados Agentes

Y también:

EstadoHistorico	Valor
Reg	0
Lib	1
GestDif	2
Ocup	3
Pau	4

Ilustración 88 - Tabla Estado Histórico

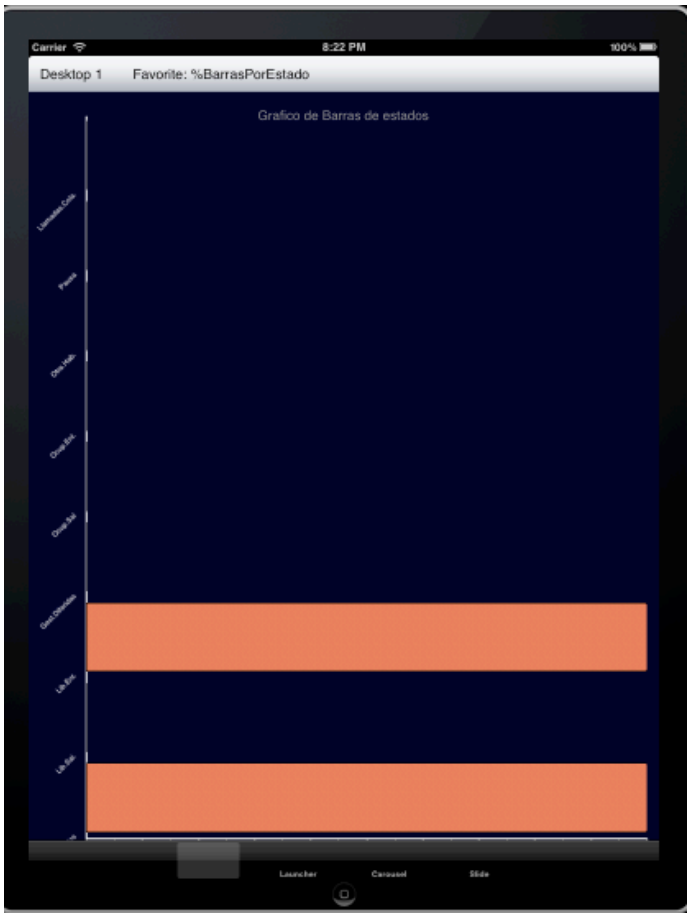


Ilustración 89 - Favorito 8

Se representa barras de estados pero a diferencia del caso anterior que se hacía uso también de ese tipo de gráfica, ésta no es de valor absoluto, es de porcentaje. Se observa que está en el estado de LibreEnt y Registrado. Los estados son:

Estado	Valor
Registrados	0
LibSal	1
LibEnt	2
GestDiferidas	3
OcupSal	4
OcupEnt	5
OtraHabilidad	6
Pausa	5
LlamadasCola	8

Ilustración 90 - Tabla Estados

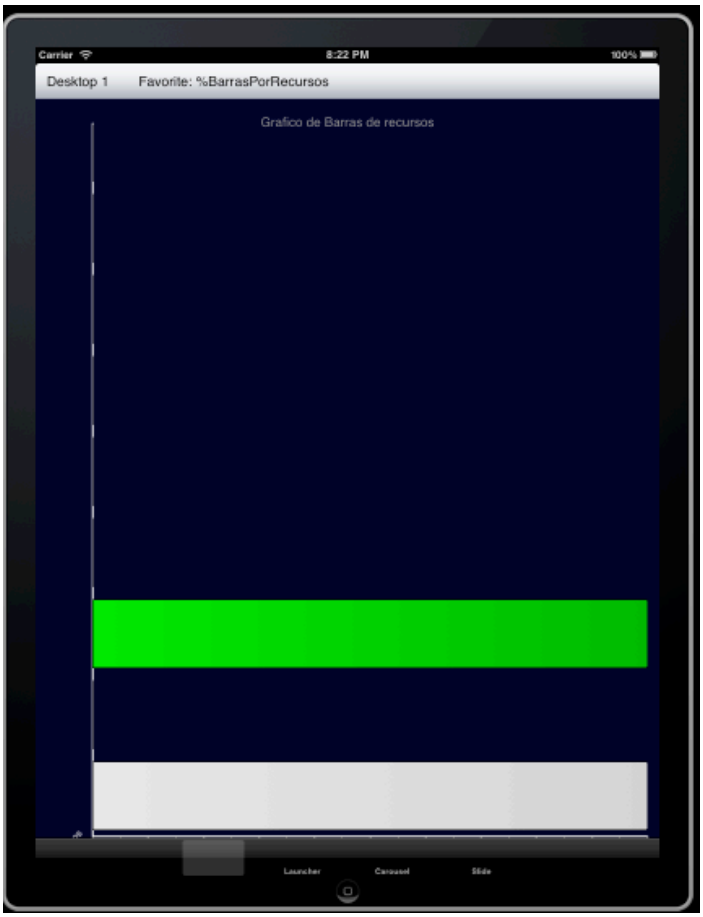


Ilustración 91 - Favorito 9

Como antes, el tipo de operación es porcentaje, pero esta vez de recursos en vez de estados. Nuestro recurso como se menciono es OTROS únicamente siendo en estado libre y registrado los agentes de dicho recurso.

Conclusiones

Al comienzo de la parte de **Diseño**, enumere qué requisitos debería cumplir la aplicación:

1. Atractivo visualmente: Es lo primero que ofreces. Debe provocar cierto interés.
2. Facilidad de uso.: Toda aplicación debe regirse por este dogma.
3. Adaptación a cada usuario: La aplicación debe adaptarse de forma automática al perfil del responsable.
4. Rápida: La aplicación puede ser atractiva, fácil etc. Si es lenta, el interés se reduce y con ello puedes perder a un cliente potencial.

La **primera** se ha cumplido más o menos de una manera bastante aceptable. Se ha ofrecido tres vistas bien diferenciadas, cada una con sus propias características, siempre persiguiendo su atractivo visual.

La **segunda** se ha cumplido perfectamente. El manejo de vistas mediante **Tab Bar** es totalmente sencillo e intuitivo, gracias además a la costumbre nacida de aplicaciones similares. Después las funciones añadidas de pulsar la pantalla, son también fáciles de manejar e intuir.

La **tercera** no ha podido probarse debido al usuario utilizado. Esto ha provocado que funciones destinadas a esa adaptación no hayan podido usarse.

La **cuarta** funciona a la perfección. Las pruebas realizadas en ningún caso han producido una demora importante en la recopilación y procesamiento de datos. En cualquier caso, se avisa al usuario que se están cargando los datos en la pantalla del iPad.

En conclusión, se ha desarrollado en este proyecto una aplicación que permite obtener información de monitorización y supervisión desde una sistema de back-end a través de una interfaz *webservice* determinado. Una vez obtenidos, se representan dichos favoritos, ya bien siendo una monitorización de datos o una supervisión de agentes. Después se permite la visualización de la aplicación de tres maneras diferentes. La primera es por defecto, la segunda es **LauncherView** y la tercera es **Coverflow View**. La primera es la encargada de recoger todos los datos y mostrar los favoritos de manera clara y sencilla, ya sean gráficas o ya sea una supervisión de agentes. La segunda muestra todos los favoritos mediante iconos y haciendo *click* sobre cada icono muestra la gráfica relacionada. Por otro lado, **Coverflow View** hace uso de dicho efecto tan conocido para presentar las gráficas de una manera más estética.

Trabajo Futuro

Este proyecto tiene gran potencial ya sea por la portabilidad que se puede hacer a iPhone y hacia Android, si no por la implementación dada. Hay muchos ajustes que aún se pueden hacer para favorecer la rapidez de recogida de datos o actualizar con mayor frecuencia los valores. Otra implementación posible sería capaz de escuchar en tiempo real las conversaciones de los agentes con el cliente sin afectar a su rendimiento como modo superior de supervisión.

Por otro lado, la adaptación del usuario no ha podido realizarse de la mejor manera posible debido al campo de trabajo. Hay infinitas líneas de código para comprobar el usuario y su autorización para poder desempeñar funciones que provocado al estancamiento de proyectos en mi empresa, no he podido probar como debiera haciendo únicamente uso de un usuario predeterminado con total acceso para poder realizar las pruebas y acceder a la base de datos.

Esto unido al servidor nutrido por agentes ficticios impiden obtener una visión mas general que de otra manera, con mayores pruebas, se podría haber logrado mejores resultados.

Bibliografía

[1] http://www.xatakamovil.com/mercado/la-carrera-de-las-plataformas-moviles-actuales-en-una-completa-infografia

[2] http://www.ticbeat.com/sim/plataforma-movil-domina-pais/

[3] Android http://developer.android.com/about/index.html

[4] http://arstechnica.com/gadgets/2009/02/an-introduction-to-google-android-for-developers/

[5] J. Eisenstein, J. Vanderdonckt, and A. Puerta. Applying model-based techniques to the development of UIs for mobile computers. In Proceedings of the 6th international conference on Intelligent user interfaces, pages 69–76. ACM, 2001.

[6] Open handset alliance. http://www.openhandsetalliance.com/, 2010.

[7] Android market. http://www.android.com/market/, 2010.

[8] Samsung http://www.samsung.com/es/index.html

[9] http://onlamp.com/pub/a/onlamp/2007/11/12/google-calling-inside-the-gphone-sdk.html

[10] Java http://www.java.com/es

[11]RIM http://www.rim.com/

[12] http://es.wikipedia.org/wiki/Teclado_QWERTY

[13] P. Saco et al. La telefonía celular en la era blackberry. Negotium, 5(13):71–79, 2009

[14] Blackberry App World http://es.blackberry.com/services/appworld/download.jsp

[15] http://www.infoworld.com/d/hardware/microsoft-phase-out-pocket-pc-smartphone-brands-232

[16] http://channel9.msdn.com/Events/TechEd/NorthAmerica/2010/WPH201

[17] H.R.G. Brito. Estudios de las plataformas principales en la informática móvil. 2006.

[18] Windows marketplace. http://marketplace.windowsphone.com/

[19] http://tecnologia.elpais.com/tecnologia/2011/02/11/actualidad/1297418462_850215.html

[20] Apple store. http://www.apple.com/es/iphone/apps-for-iphone/, 2010.

[21] Distimo app store analytics. http://www.distimo.com/, 2010.

[22] Objective c. http://developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC/, 2010.

[23] http://www.todocelular.es/Apple/noticias/n26016/ios-historia.html

[24] http://www.applesfera.com/apple/la-app-store-consigue-25-mil-millones-de-descargas

[25] SimplyCT http://simplyct.com/

[26]Ring Central http://www.ringcentral.com/

[27]Oracle CRM On Demand http://www.oracle.com/us/products/applications/crmondemand/index.html

[28] http://leerenpantalla.com/tabletas-en-el-mercado-cual-es-la-mas-vendida/

[29] http://www.indracompany.com/

[30] http://www.zenbrains.com/blog/2010/06/configurando-una-tab-bar/

[31] Instagram http://instagram.com/

[32] Foursquare http://foursquare.com

[33] Soundcloud http://soundcloud.com/

[34] Shazam http://www.shazam.com/

[35] Barclays http://www.barclays.es/

[36] Coreplot http://code.google.com/p/core-plot/

[37] Cocoa programming for Mac Os 3rd Edition, Aaron Hillegass.

[38] Introduction to Cocoa Programming for Mac OS X, Arthur Clemens

[39] Learning Cocoa with Objective-C, James Duncan Davidson

[40] The Mac OS X Solutions Guide, Rob Griffiths

[41] Programming in Objective-C 2.0, Stephen G. Kochan

Apéndice A-Presupuesto

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... 2.000 €
- Alquiler de impresora láser durante 6 meses..... 50 €
- Material de oficina..... 150 €
- Total de ejecución material..... 2.200 €

2) Gastos generales

- 16 % sobre Ejecución Material..... 352 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material..... 132 €

4) Honorarios Proyecto

- 1200 horas a 15 € / hora..... 18000 €

5) Material fungible

- Gastos de impresión..... 60 €
- Encuadernación..... 200 €

6) Subtotal del presupuesto

- Subtotal Presupuesto..... 22959 €

7) I.V.A. aplicable

- 16% Subtotal Presupuesto..... 3673,44€

8) Total presupuesto

- Total Presupuesto..... 26632,44 €

Madrid, Septiembre de 2012

El Ingeniero Jefe de Proyecto

Fdo.: Ángel Manuel Pérez Villar
Ingeniero Superior de Telecomunicación

Apéndice B - Pliego de Condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un Aplicación sobre Dispositivos móviles para la Monitorización de Contact Centers Multitenant. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se

discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de

la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha

aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.

```
#import "SessionServices.h"
#import <libxml/xmlstring.h>
#if TARGET_OS_IPHONE
#import <CFNetwork/CFNetwork.h>
#endif
@implementation SessionServices_LogOnRoot
- (id)init
{
	if((self = [super init])) {
	  user = 0;
	  password = 0;
	  ip = 0;
	  language = 0;
	}

	return self;
}
```

De SessionServices.m: Nos permite identificarnos en la aplicación

---

```
- (id) logOnRoot:(NSString*)user password:(NSString*)password ip:(NSString*)ip
language:(NSNumber*) language inAddress:(NSString*) address
logLevel:(NSNumber*) log{

	id logOnResult = nil;

	if ( nil == self.binding ){
	  self.binding = [SessionServices BasicHttpBinding_SessionServicesBinding];
	  [self.binding initWithAddress:address];

	  if ( 1 == [log intValue] || 3 == [log intValue] ){
		self.binding.logXMLInOut = YES;
	  }
	}

	SessionServices_LogOnRoot *request = [[SessionServices_LogOnRoot new]
autorelease];

	request.user = user;
	request.password = password;
```

```
        request.ip = ip;
        request.language = language;

        BasicHttpBinding_SessionServicesBindingResponse *response = [self.binding
LogOnRootUsingParameters:request];

        NSArray *responseBodyParts = response.bodyParts;

        for(id bodyPart in responseBodyParts) {
          if([bodyPart isKindOfClass:[SessionServices_LogOnRootResponse class]]) {
                SessionServices_LogOnRootResponse *body =
(SessionServices_LogOnRootResponse*)bodyPart;

                logOnResult =
(tns1_SessionLogOnResultItem*)body.LogOnRootResult;
          } else if([bodyPart isKindOfClass:[SOAPFault class]]) {

                logOnResult = (SOAPFault*) bodyPart;
          }
        }

        return logOnResult;
}
```

De SessionServicesWrapper: es la librería intermedia que permite que nuestra aplicación se conecte al *webservice*.

```
- (void)getFavoritesManagementConfiguration{
        self.favoritesManagementServicesWrapper =
[[FavoritesManagementServicesWrapper alloc] init ];
   NSNumber *log = [ NSNumber numberWithInt:3];
   NSNumber *lang = [ NSNumber numberWithInt:1];
   NSNumber *time = [ NSNumber numberWithInt:60];
    //conseguir favoritos
   [self.favoritesManagementServicesWrapper
getFavoritesConfigurationByUser:self.user withKey:self.key withLanguage:lang
inAddress:@"http://dirac/ImpacteBackend/FavoritesManagementServices.svc/ba
sicHttp"  logLevel:log timeout:time];

   NSLog(@"La lista de favoritos vale: %@",
self.favoritesManagementServicesWrapper.favorites);
 }
```

Función dentro de nuestra aplicación que a través de FavoritesManagementServicesWrapper obtenga los favoritos del *webservice*.