UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Oportunidad de la tecnología móvil para colaboración en desarrollo software

Autor: Víctor Martín Aguilar

Director: Dr. Patricio Letelier Torres

Julio del 2014

Trabajo Final de Máster presentado para cumplir con los requisitos finales para la obtención del título de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información, de la Universidad Politécnica de Valencia, 2014.

Agradecimientos

A Carlos del Fresno Canales por su colaboracion resolviendo cuantas dudas me surgieron en el desarrollo de la aplicación móvil.

Tabla de Contenidos

Tabla de Figuras

Capítulo 1. Introducción

1.1. Motivación

Los últimos años la penetración de internet y un aumento en el ancho de banda en los dispositivos móviles ha permitido la utilización de dispositivos móviles para algo más que el uso de llamadas o aplicaciones básicas. Estas nuevas aplicaciones permiten explotar los recursos de las empresas, lo que aumenta la productividad y mejora de la competitividad.

En la figura 1, podemos ver un gráfico de ONTSI (**O**bservatorio **N**acional de las **T**elecomunicaciones y de la **S**ociedad de la **I**nformación), en el que se refleja la gran penetración del equipamiento de tecnología móvil en las empresas.

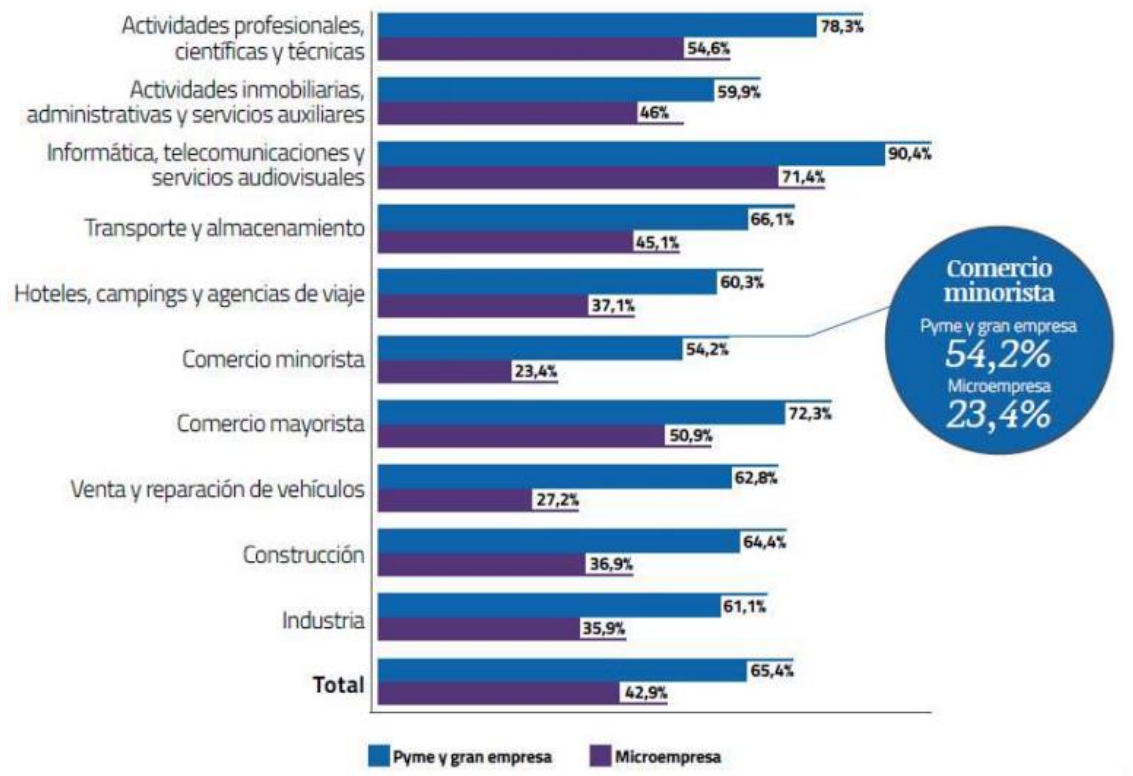


Figura 1. Equipamiento TIC básico, banda ancha móvil. Fuente ONTSI (Datos 2012)

En concreto el sector de la informática es el que más avance tiene, destacando incluso a nivel de microempresas, muy por encima que en otros sectores. Refleja la importancia de esta nueva tecnología en las TIC.

Los dispositivos móviles son cada vez más potentes, la posibilidad de conectarse desde cualquier lugar y en cualquier momento facilita las tareas a los empleados. Es necesario ofrecer proyectos con el mínimo coste, en el menor tiempo posible y máxima calidad. Para conseguirlo hay que utilizar metodologías flexibles, con alto grado de productividad y que puedan desplegarse sobre plataformas heterogéneas, ya que permite adaptarse a las necesidades cambiantes de las empresas y no estar ligado a ninguna tecnología móvil.

En el contexto del desarrollo software, los equipos de desarrollo que utilizan metodologías ágiles para mejorar su productividad, deben poder acortar los tiempos de respuesta, aumentar la fluidez de comunicación con el cliente, adelantarse a los problemas antes de que ocurran, en definitiva encontrar una oportunidad en la tecnología móvil para mejorar el desarrollo software.

1.2. Objetivos

El objetivo de esta tesis, es establecer criterios respecto de qué funcionalidades serán interesantes trasladar al contexto de dispositivos móviles con el propósito de mejorar el trabajo en equipo en el contexto de desarrollo software.

Se establecerán funcionalidades a partir de los conceptos de Context Awareness para los agentes implicados en un proyecto, no solamente asociados a tareas tales como análisis y programación, sino especialmente para conocer cómo evoluciona el proyecto a partir de decisiones propias o de terceros, lo que se traducirá en una mejora de los resultados. El trabajo en equipo es fundamental, pero para conseguir el máximo beneficio el equipo tiene que estar implicado, disponiendo de la información del proyecto en cualquier momento y lugar.

Este análisis lo reflejaremos construyendo una aplicación integrada en TUNE-UP, pero queremos ir un paso más, esta aplicación móvil no debe ser exclusiva de una plataforma, y para conseguirlo haremos uso de las tecnologías multiplataforma, para escribir el código una vez y tener desplegada la aplicación al menos con las plataformas más representativas del mercado.

1.3. Estructura del trabajo

A continuación se presenta la organización del trabajo:

El Capítulo 1 incluye una introducción a la motivación de nuestro trabajo y presenta los objetivos de nuestra propuesta.

En el Capítulo 2 explica los aspectos básicos de la metodología TUNE-UP y de su herramienta de apoyo TUNE-UP Software Proccess.

En el Capítulo 3 analizaremos el estado del arte de herramientas similares a TUNE-UP desde el punto de vista de dispositivos móviles.

El Capítulo 4 Shared Context Awareness explica los conceptos básicos de Shared Context Awareness tomando como referencia los trabajos de la investigadora Mica Endsley.

En el Capítulo 5 Tecnologías para el desarrollo de aplicaciones móviles, recoge las herramientas con mayor empuje del momento y realiza un análisis de cada una de ellas. Las herramientas propuestas han sido PhoneGap, Titanium, Genexus y RhoMobile Suite.

En el Capítulo 6 Desarrollo móvil en Shared Context Awareness, pondremos de manifiesto cómo aplicaremos el Shared Context Awareness para la aplicación móvil de TUNE-UP.

El Capítulo 7 PhoneGap, explicaremos sobre la herramienta seleccionada, los pasos necesarios de instalación hasta obtener un primer ejemplo ya operativo sobre un terminal. Desarrollo de la propuesta realizada en el Capítulo 3 con PhoneGap.

En el 0 Conclusiones, damos nuestras conclusiones y los trabajos futuros que complementan el trabajo actual.

El Anexo I, mostraremos la documentación recopilada a través de un cuestionario realizado a las distintas tecnologías multiplataformas.

Capítulo 2. Proceso y herramientas

2.1. Metodología TUNE-UP

TUNE-UP (http://www.tuneupprocess.com/) ha sido desarrollada en la Universidad Politécnica de Valencia, en el grupo de investigación ISSI (**I**ngeniería del **S**oftware y **S**istemas de **I**nformación), con una estrecha colaboración de empresas en diversos proyectos.

TUNE-UP es una metodología que da soporte a la gestión ágil de proyectos para su desarrollo y mantenimiento, basándose en un proceso iterativo e incremental, además de seguir una planificación y control de tiempos. TUNE-UP Software Process es la herramienta que da apoyo a la metodología TUNE-UP.

La metodología TUNE-UP, tiene como objetivo, la entrega de un producto software de calidad y en los tiempos establecidos, para ello incluye los siguientes elementos: [15,16]

- ***Modelo de proceso iterativo e incremental***, las tareas se dividirá en unidades de trabajo (WUs) que serán asignadas a versiones. Una versión tendrá asignada unas fechas de trabajo que dependiendo del tamaño del producto pueden variar en el tiempo entre 3 y 6 semanas.

- ***Proceso de desarrollo dirigido por las pruebas***, el desarrollo de una WU gira en torno a las pruebas de aceptación que serán consensuadas con el cliente. Las pruebas de aceptación reflejaran el nivel de cumplimiento y porcentaje de completitud de la WU. Con la definición de estas pruebas se estima el tiempo de implementar y aplicar las pruebas de la WU. Figura 2.

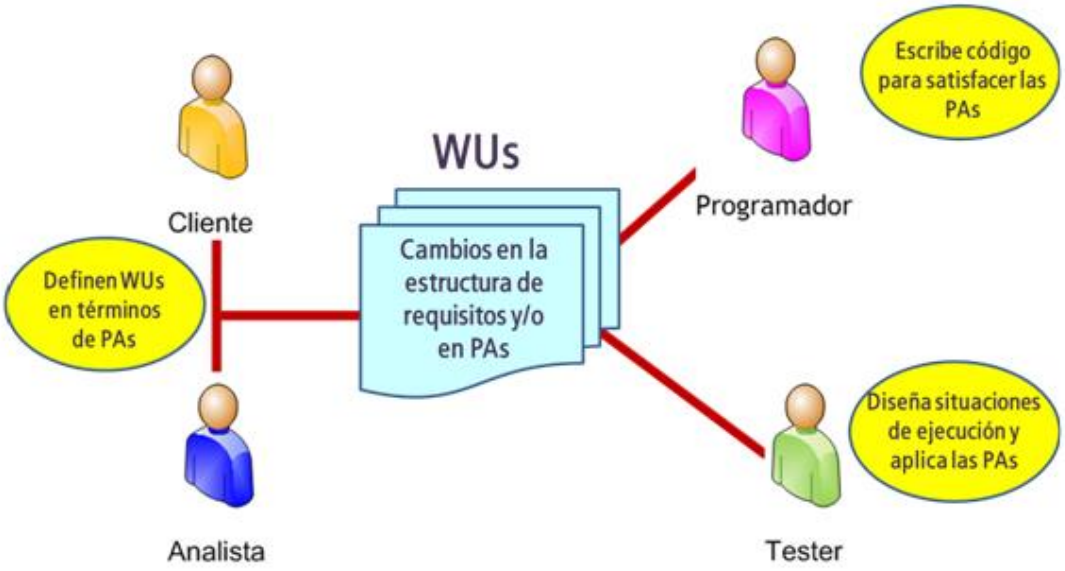


Figura 2. Proceso dirigido por pruebas de aceptación. [24]

- ***Workflows flexibles para la coordinación del trabajo asociado a cada unidad de trabajo***, dependiendo del producto este tendrá asociados un conjunto de workflows, que determinaran los pasos necesarios para la realización de una WU. La flexibilidad permite avanzar o retroceder en el workflow, e incluso si fuera necesario añadir nuevas actividades. En el propio workflow podremos ver los roles que hay para cada una de las actividades. Figura 3.

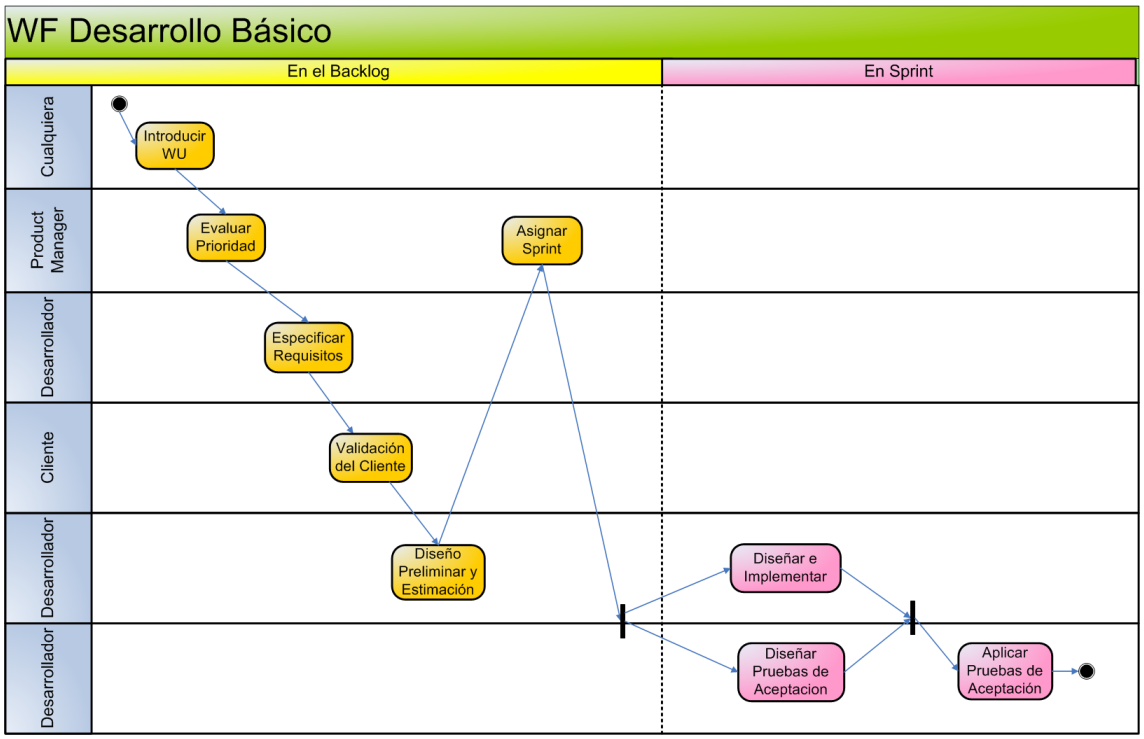


Figura 3. Un ejemplo de workflow básico. [24]

- ***Planificación y seguimiento***, permite conocer el estado una versión, de las WUs de una versión o del trabajo asignado a los agentes. Con esta información se puede volver a planificar la iteración con el cliente (cambio de plazos de entrega o cambio de WUs entre versiones), redistribuir cargas de trabajo entre los agentes, etc.

- ***Control de tiempos***¸ hay un registro del tiempo dedicado por los agentes a las WUs asignadas, lo que permite conocer los posibles desfases entre la estimación y la realidad. Es útil para la toma de decisiones (como hemos visto en el punto anterior de planificación y seguimiento), de igual modo que sirve de referencia para los agentes para mejorar sus estimaciones futuras.

2.2. Herramienta TUNE-UP Software Process

La herramienta TUNE-UP Software Process es una herramienta que da soporte a la metodología TUNE-UP. La herramienta se divide en 6 módulos: [15,17]

- ***Planificador personal (PEP)***, figura 4, presenta todo el trabajo que tiene pendiente un agente. La información se divide en el área del kanban, el área de las comunicaciones y en el área del detalle, esta última área mostrara el seguimiento de las actividades del kanban o la información del área de las comunicaciones. Esta presentación permite al agente poder visualizar las tareas y establecer prioridades. En el área del kanban el agente puede ver un resumen de las actividades en las que el agente está implicado y el número de unidades de trabajo que se encuentran en estado ToDo y Doing.

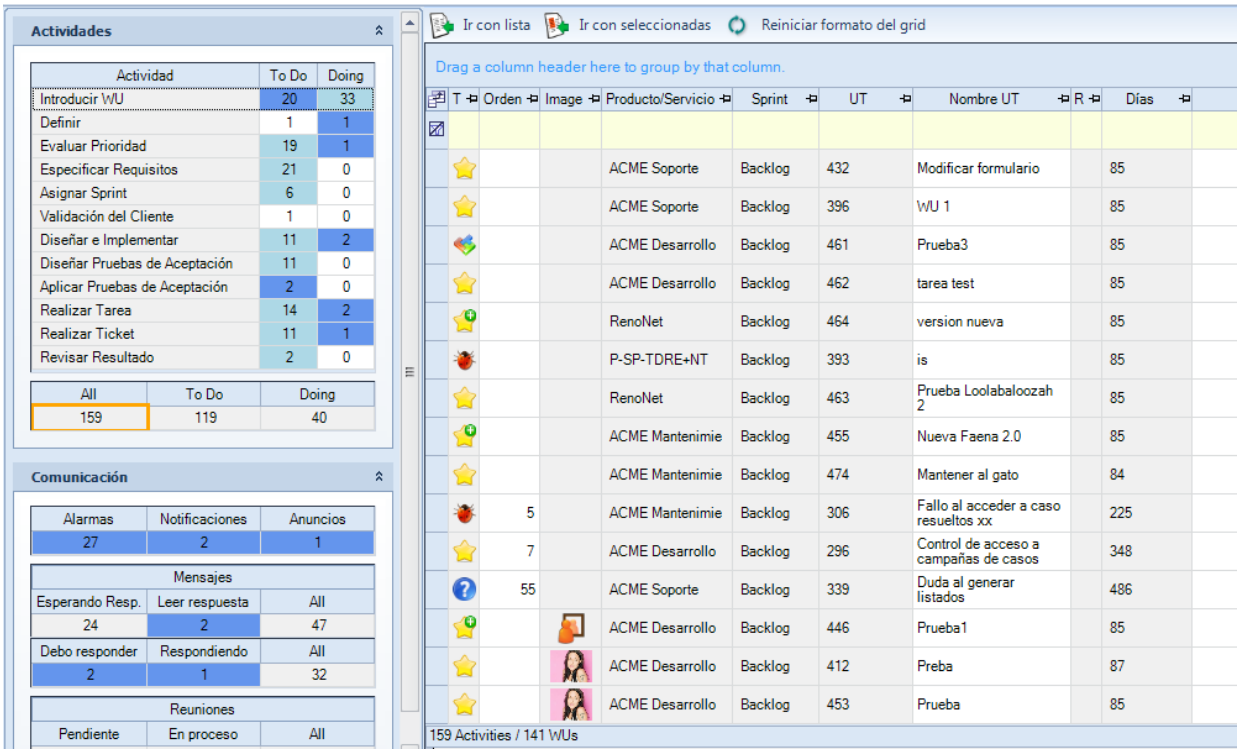


Figura 4. Planificador Personal

En el área de comunicaciones, el agente puede ver alarmas que son procesos automáticos que se generan ante situaciones anómalas y que solo desaparecen poniendo solución a estas (por ejemplo, se ha excedido el tiempo para desarrollar una WU, la solución es finalizar la WU o ampliar el tiempo marcado inicialmente), además el agente recibirá notificaciones (generados automáticamente) y avisos (generados por otros agentes), no tienen posibilidad de réplica pero si de confirmación de lectura. Los mensajes es el medio que utilizan los agentes para comunicarse sobre una WU, que posibilita la retroalimentación.

- ***Workflows***, TUNE-UP Software Process incorpora un motor de workflow que permite que una unidad de trabajo siga el flujo del workflow, de forma que a medida que los agentes terminan la actividad de una unidad de trabajo, esta pasara a la siguiente actividad según el workflow, y el nuevo responsable puede ser el mismo agente u otro. La flexibilidad se consigue, permitiendo añadir nuevos estados al workflow, y permitiendo avanzar o retroceder en el flujo de datos, e incluso cambiar de workflow. El cambio de agentes y el trabajo en paralelo son otras características que maximizan la flexibilidad del workflow. En la figura 3 tenemos un ejemplo de 3 roles y 9 actividades.

- ***Gestor de unidades de trabajo (WUM)***, es el espacio de trabajo de los agentes para una WU, facilita el trabajo colaborativo y recoge toda la información de una WU en un solo lugar, permitiendo que todos los agentes que en algún momento tengan que trabajar en esta WU encuentren la información, el estado, y su evolución. En la figura 5 podemos ver un ejemplo, la parte superior corresponde a la información genérica de la WU, después tenemos varias pestañas que nos permiten saber el estado de la WU y la actividad en que se encuentra, los agentes involucrados, documentos asociados a la WU, mensajes entre los agentes, registro de tiempos, que permitirá saber la evolución de la WU, y un punto muy importante, las pruebas de aceptación, que será donde se refleje el nivel de finalización de la WU.

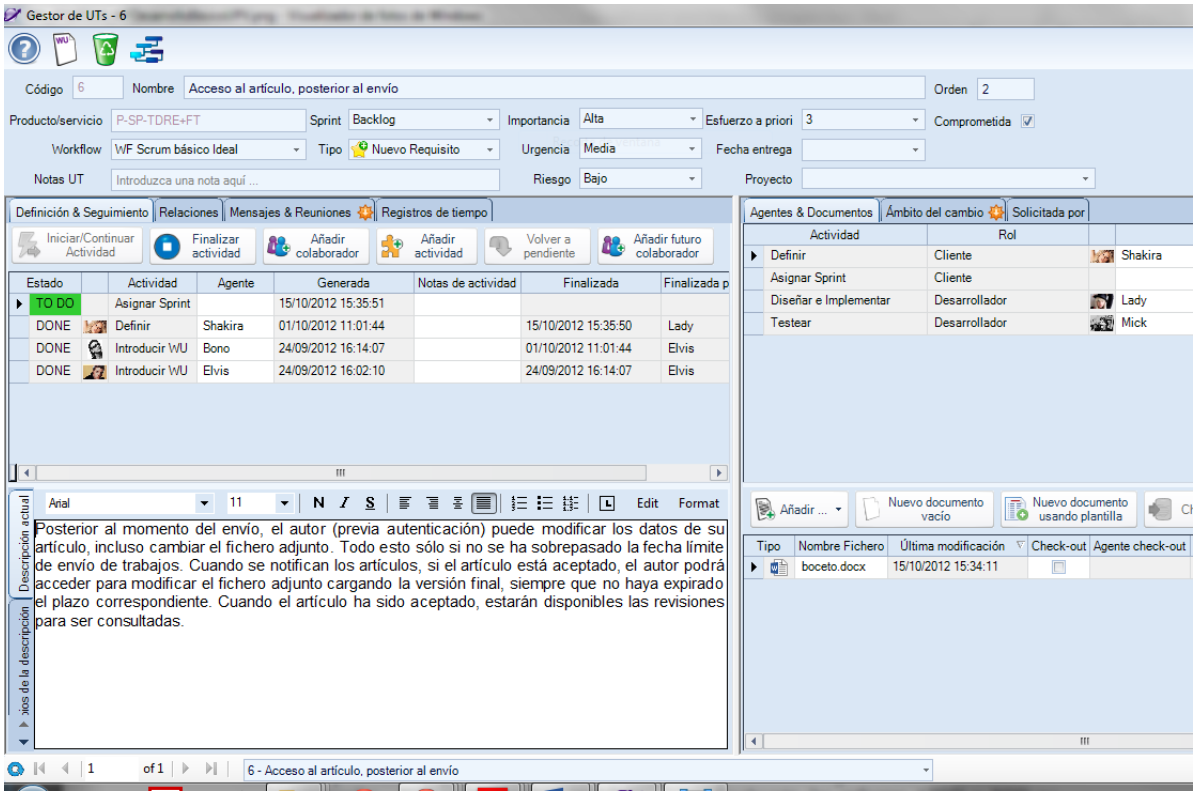


Figura 5. Gestor de unidades de trabajo

- ***Versiones y seguimiento (VCT)***, permite realizar la gestión de los productos. Podemos ver el estado de cada una las WUs para una versión, tanto el estado de finalización de programación como de testeo. En las distintas pestañas podremos ver la carga de trabajo de los agentes, las relaciones entre las WUs y los nodos afectados por las WUs.

- ***Cuadro de mando TUNE-UP (Dashboard)***, permite en una sola pantalla tener una instantánea de la situación del producto, apoyándose en las gráficas de Burn Down, el esfuerzo por tipo de actividad, el estado de finalización de las WUs y el estado de finalización de las pruebas de aceptación. Como vemos en la figura 6, la gráfica de Burn Down muestra información diaria de la evolución del proyecto, la línea roja indica el tiempo restante para finalizar la versión, la línea azul indica el tiempo de esfuerzo invertido, la línea verde corresponde a la duración estimada y la línea morada corresponde al esfuerzo restante de referencia. La gráfica de esfuerzo por actividad permite ver desglosado para cada WU el esfuerzo destinado a responder mensajes, participar en reuniones y resto de esfuerzo. La gráfica estado de finalización de las WUs, nos muestra de manera ponderada el porcentaje de WUs no finalizadas respecto a las finalizadas. Finalmente la gráfica de finalización de las pruebas de aceptación nos muestra para cada WU, el estado de aceptación de las pruebas, porcentaje de correctas, de incorrectas, o si se necesitan revisar de nuevo las pruebas.

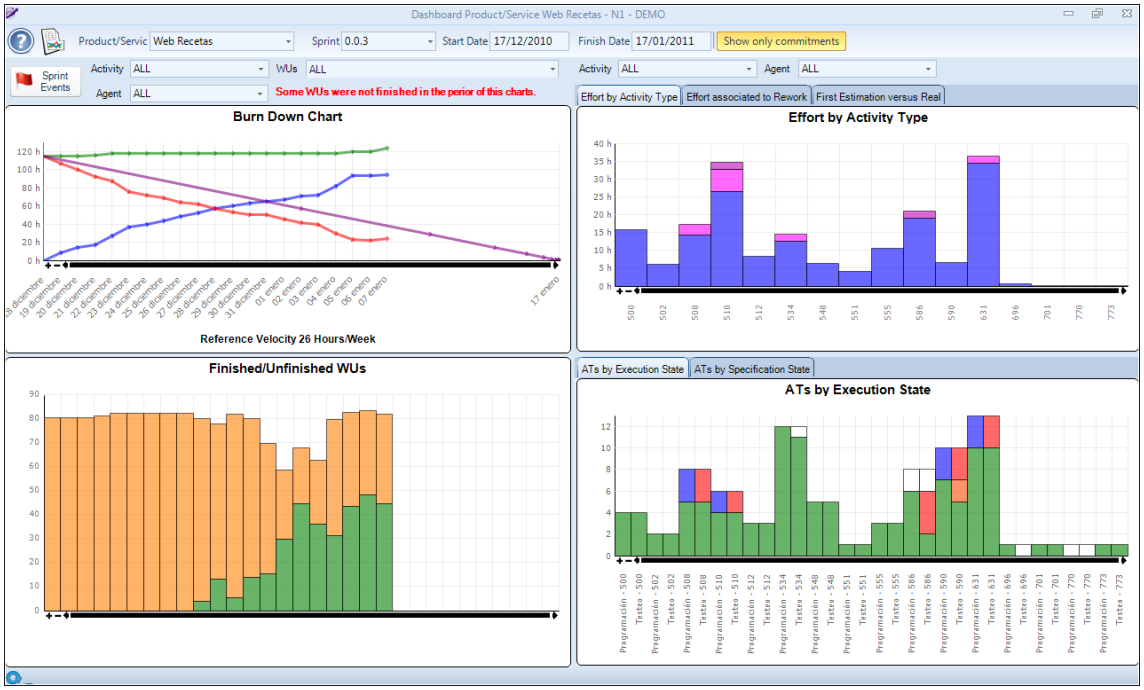


Figura 6. Gráfica Dashboard

- ***Gestor de requerimientos (REM)***, nos permite ver como los requisitos que definimos para las WUs se van cumpliendo a partir de las pruebas de aceptación consensuadas con el cliente. Se podrá visualizar los nodos que se ven afectados por una WU.

Capítulo 3. Aplicaciones móviles para la gestión de proyectos

En el mercado hay muchas aplicaciones que ofrecen servicios similares a TUNE-UP Software Process, pero lo que vamos a revisar es cuales de ellas ofrecen además un servicio móvil, e intentar conocer que funcionalidades están implementadas y si detrás de la aplicación podemos conocer los criterios de implementación. Algunas de la aplicaciones que comentaremos a continuación no tienen aplicación móvil, lo que ofrecen a cambio es una interfaz web adaptada a los terminales móviles.

- ***JIRA***, www.atlassian.com/software/jira/, [20] muestra sobre la interfaz web, una versión de la aplicación. La figura 7 ilustra que podemos hacer en JIRA móvil. Jira móvil es compatible con mobile safari y el navegador por defecto de Android 4.0.3.
    - Visualización de temas, comentarios, archivos adjuntos, enlaces de emisión y sus filtros favoritos.
    - Realización de operaciones básicas como la adición de comentarios (incluyendo menciones y restricciones), viendo o votando en los asuntos y la asignación de asuntos a los usuarios.

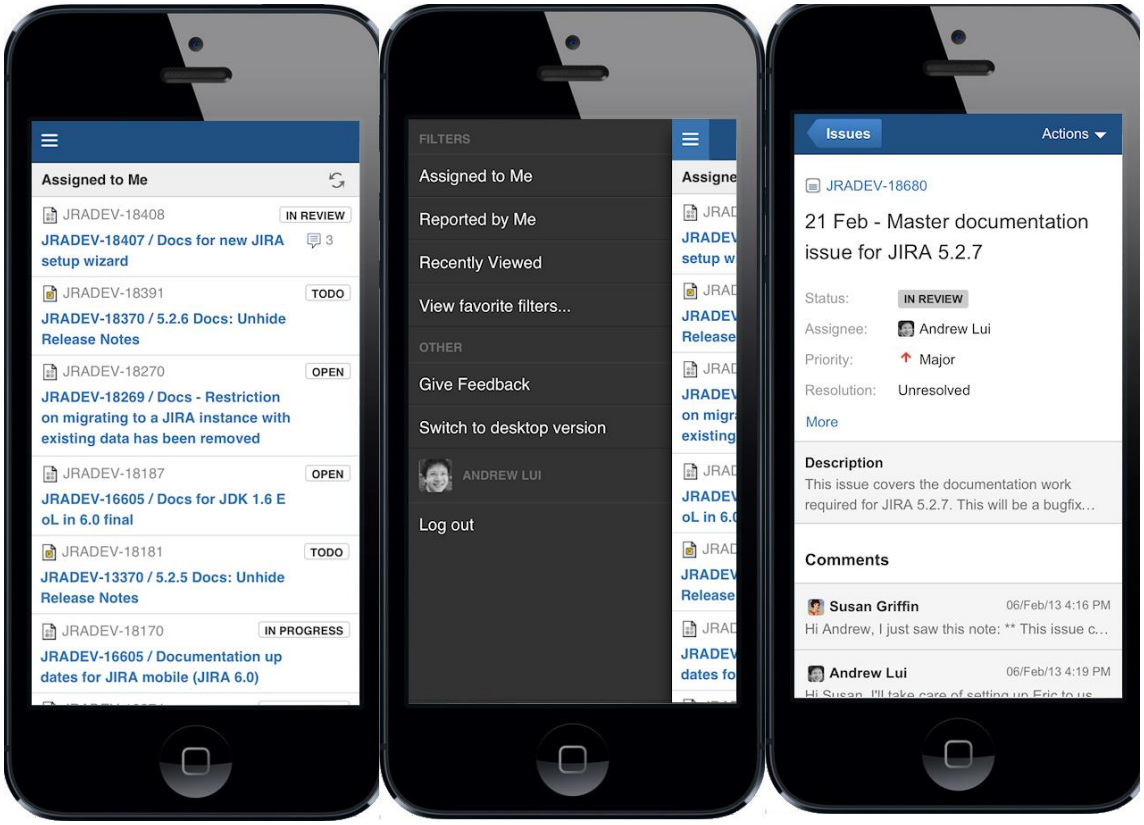


Figura 7. JIRA en dispositivo móvil

- ***OnTime***, www.axosoft.com/ontime, no se ha encontrado ninguna referencia al soporte en dispositivos móviles ni con interfaz web ni como aplicación.

- ***Rally***,www.rallydev.com/,[21] tiene versión para móvil sobre IPhone (figura 8), las características más destacadas son:

    - Indicador de estado de la salida / iteración
    - Gráfico de barras de aceptados y listas de tareas pendientes
    - Historial programados en una iteración
    - Gestión de requisitos, cartera de pedidos, incluyendo drag-and-drop re-clasificación en una iteración
    - Posibilidad de añadir y editar nuevas historias o defectos
    - Vista de tareas de usuario individual
    - Consulta y edición de tareas, incluyendo la descripción y las estimaciones

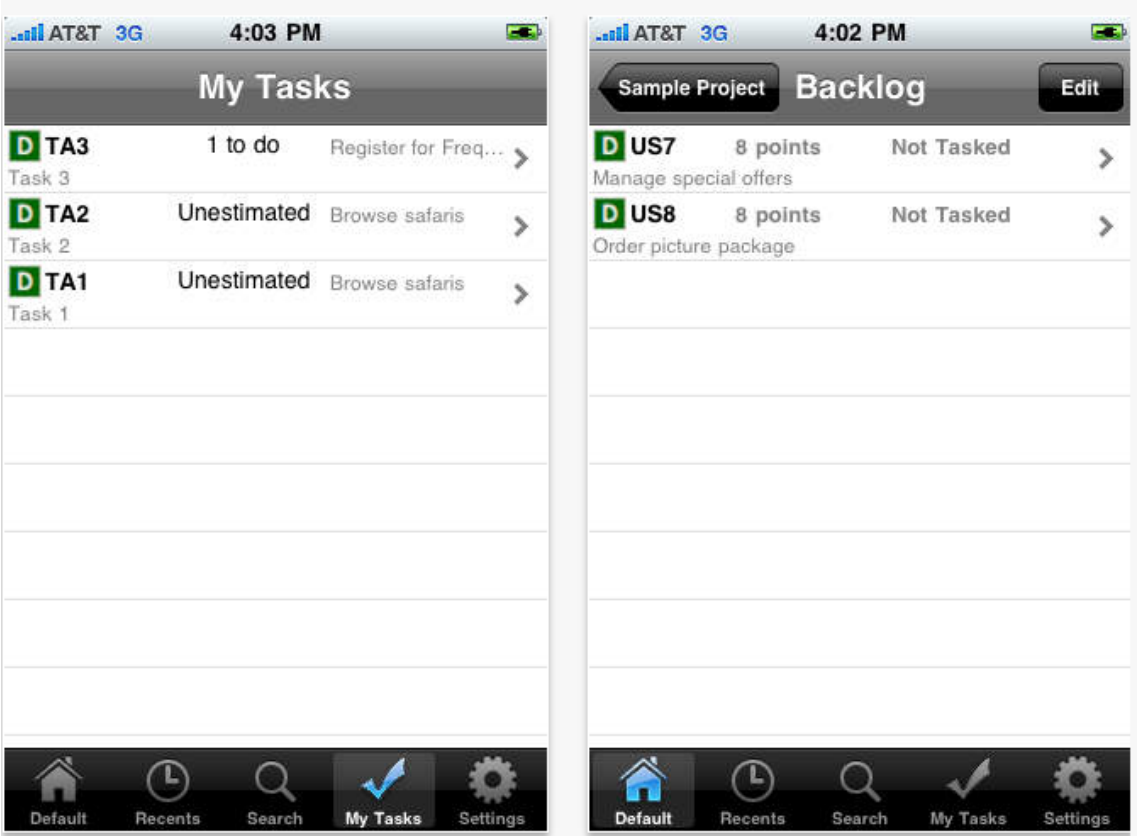


Figura 8. Rally en dispositivos móviles

- ***TargetProcess***,www.targetprocess.com/, [22] ofrece una versión para dispositivos iOS (figura 9) y está trabajando en los dispositivos Android. Las funcionalidades más destacadas son.
    - Seguimiento del progreso
    - Añadir nuevo trabajo (Historias de usuario, funciones, etc.)
    - Marcar como finalizadas
    - Buscar cualquier detalle, por ejemplo, los comentarios, las tareas, las tareas relacionadas.

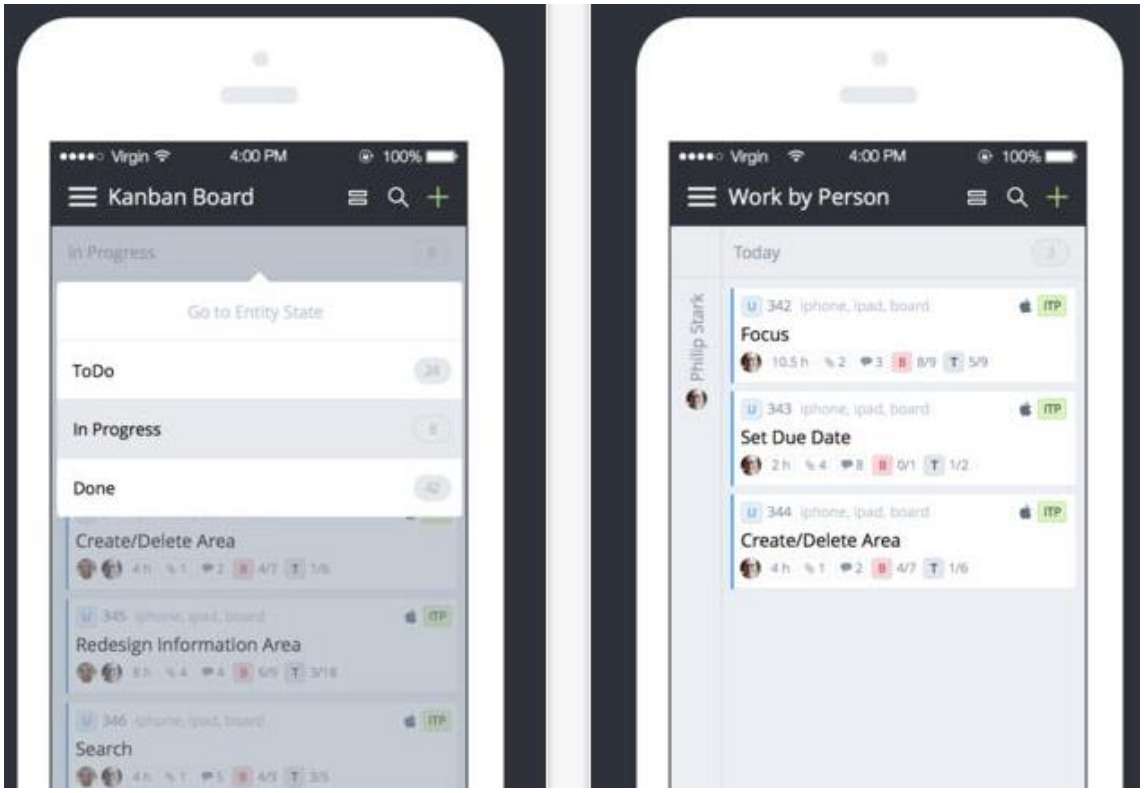


Figura 9. Targetprocess para móviles

- ***Team Concert***, www-01.ibm.com/software/rational/products/rtc/, no se ha encontrado ninguna referencia al soporte en dispositivos móviles ni con interfaz web ni como aplicación.

- ***VersionOne***,www.versionone.com/, versión para iOS (figura 10), las principales funcionalidades son:

    - Todas las funciones de apoyo de tareas, pruebas, historias y defectos. Estado de actualización, información de seguimiento

    - Detalles de la opción de los padres de Tareas y Análisis de su propiedad

    - Añadir seguimiento de la información para todos los usuarios asignados

    - Profundizar en la descripción del artículo para la vista de pantalla completa del texto

    - Visualización de la configuración VersionOne para mostrar listas de estado mediante la configuración del espacio de trabajo.

- Utiliza el modelo de seguridad VersionOne para determinar si tiene permisos para actualizar el estado o los elementos cercanos.

Figura 10. Versione para móviles

Aunque algunas herramientas disponen de aplicación para móvil, tratándose de una versión más ligera que la de escritorio, no ha sido posible conocer las funcionalidades, ya que no existe documentación asociada.

Capítulo 4. Shared Context Awareness

En el desarrollo software es importante saber lo que ocurre en nuestro entorno de trabajo, si además trabajamos en equipo se requiere un buen nivel de ***Situational Awareness*** [4], para que los cambios significativos en tareas interconectadas y que por lo tanto afectan a varios miembros del equipo reciban la información necesaria para poder actuar en consecuencia.

Si además, este desarrollo software, es con metodologías ágiles, los miembros del equipo deben ser capaces de percibir los cambios y predecir cómo evolucionará el proyecto [2]. De no ser así, los tiempos de desarrollo y pruebas aumentarán, la calidad del producto se verá afectada y la probabilidad de rechazo por parte del cliente se incrementará.

La definición de ***Situational Awareness*** más aceptada es la de la investigadora en factores humanos Mica Endsley, que cita:

La consciencia situacional es la percepción de los elementos existentes en el entorno en un volumen de tiempo y espacio, la comprensión de su significado y la proyección de su estatus en el futuro cercano. [4]

4.1. Modelo de Endsley

Los componentes esenciales de ***Situational Awareness*** según ***Endsley***: [1,4]

- ***Percepción***, es el nivel más básico de ***Situational Awareness***, incluye el conocimiento de los estados, eventos, gentes, sistemas y factores ambientales. En este nivel tenemos los datos relevantes del dominio. Este nivel se conoce como ***Level 1 SA***. Un ejemplo de la vida cotidiana, sería percibir los estados de un vehículo a través del cuadro de mando y sonido del motor.

- ***Comprensión***, en este nivel se procesará la información para la consecución de las metas y los objetivos. Demuestra la habilidad para la comprensión de la información y la creación de patrones. Este nivel se conoce como ***Level 2 SA***. Siguiendo con el ejemplo anterior, la comprensión nos permite tomar medidas según el nivel de gasoil.

- ***Proyección***, consiste en predecir las acciones futuras en base al conocimiento del estado y la creación de un modelo mental. Este nivel es el más alto de ***Situational***

Awareness. Un ejemplo relacionado con los anteriores puntos, consiste que a partir del recorrido que queramos realizar y la reserva de gasoil, prevemos realizar repostaje en una u otra gasolinera.

4.2. Medidas de Situational Awareness

Podemos dividirlo en tres categorías [1, 2, 4]:

- ***Explicitas***, la técnica más conocida es la denominada ***SAGAT (S****ituation* ***A****wareness* ***G****lobal* ***A****ssessment* ***T****echnique)* sugerida por ***Endsley***, en la que durante una tarea esta se para, y se realiza un cuestionario al individuo para conocer su conciencia actual. Para evitar parar las tareas, se han planteado técnicas alternativas que hacen uso de un cuestionario verbal.

- ***Implícitas***, son medidas objetivas, en cuanto que la consecución de unas determinadas tareas, en un tiempo y una corrección, permiten evaluar de forma objetiva el nivel de ***Situational Awareness***. La ventaja de esta medida, es que se puede realizar sin interrumpir la tarea, en cambio, esta medida por sí sola no es suficiente, y debe apoyarse en otras medidas, ya que la relación de un buen nivel de ***Situational Awareness*** no siempre tiene que ir asociada a un buen resultado, aunque sí que es lo deseado.

- ***Subjetivas***, existen varias técnicas, todas ellas complementarias, una seria preguntar directamente a los individuos para que se autocalifiquen, Un ejemplo es ***SART*** (***S****ituation* ***A****wareness* ***R****ating* ***T****echnique*) desarrollada por Taylor en el año 1989. Otra forma seria medir a través de expertos, en este caso tienen un conocimiento más completo de la situación, pero carecen de información sobre el estado mental del individuo, para lo que deben centrarse en las acciones y la verbalización para conocer su nivel de ***Situational Awareness***.

4.3. Team Situational Awareness

En el desarrollo software, las categorías y medidas antes expuestas, no se pueden entender como individuales, ya que la interacción en grupo es vital y en el desarrollo con metodologías agiles, es más importante esta comunicación debido al alto dinamismo de los procesos.

Nos encontramos un escenario donde, como se muestra en la figura 11, en el que hay una serie de tareas que se ven solapadas entre los miembros del equipo. En estos solapamientos es donde cobra fuerza el ***Situational Awareness*** ya que se debe producir la mayor coordinación entre los miembros para que cualquier cambio que se pueda producir llegue a todos los miembros implicados de forma homogénea. ***Endsley*** indica que en estos puntos los miembros implicados deben tener un alto grado de conocimiento, para que la interpretación de la información sea homogénea, ya que distintas interpretaciones dan lugar a errores y retrasos [4].

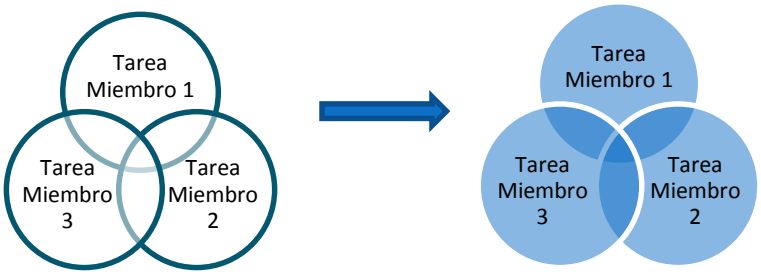


Figura 11. Solapamiento de tareas [4]

Es importante conseguir que esta información llegue en su justa medida. Un exceso de información a los miembros del equipo es nociva, porque podría generar errores al intentar solucionar tareas ajenas, o bien porque mucha información genera pérdida de tiempo en su procesamiento. El caso opuesto, también es malo, falta de información o información imprecisa puede provocar tomar acciones erróneas. El medio a través del cual se comparta la información debe ser fluido y homogéneo, para facilitar toma de decisiones rápidas si las necesidades lo requieren y a su vez permitir detectar malas decisiones.

Capítulo 5. Tecnologías para el desarrollo de aplicaciones móvil

En este capítulo realizaremos un estudio de una serie de tecnologías para el desarrollo de aplicaciones móviles, y al final mostraremos en una tabla un resumen de las características principales de estas tecnologías.

Se han elegido para comparar herramientas que entre ellas ya existen una diferencias marcadas, lo que puede dificultar la toma de decisión, pero nos permite conocer distintas maneras de acometer la misma necesidad, una herramienta de desarrollo, múltiples plataformas de despliegue.

Respecto al rendimiento, entendemos que no será mejor que las aplicaciones nativas, pero tampoco hemos encontrado ningún estudio que avale lo contrario. Por supuesto las aplicaciones nativas que utilizan las características de los terminales, deben ser más rápidas, pero también los terminales son cada vez más potentes y una buena programación puede hacer estas diferencias imperceptibles a los usuarios.

5.1. PhoneGap

Introducción

Desarrollado por Nitobi en 2011, PhoneGap se convirtió en una distribución de Apache Cordova para mantenerse como código libre, y formar parte de la casa Adobe. Así tenemos que Apache Cordova nutre a PhoneGap con el marco de trabajo básico, por eso el desarrollador podrá encontrar en PhoneGap herramientas extras para facilitar el desarrollo de aplicaciones móviles.

Para los nuevos desarrolladores que se introduzcan en el mundo de PhoneGap puede resultar confuso encontrar que pueden trabajar con las librerías de PhoneGap o Cordova, pero investigando un poco más en estas librerías veremos que hasta la fecha su contenido es idéntico. Mientras Cordova solamente trabajo en el framework, PhoneGap tiene una serie de herramientas satélite para mejorar el desarrollo de las aplicaciones multiplataforma, como pueden ser PhoneGap Build, para el despliegue del desarrollo en la nube, PhoneGap Empresa, dando soporte a aplicaciones ya existentes (nativas o

hibridas) o PhoneGap Developer App para testear las aplicaciones sobre cualquier plataforma soportada.

Características

La documentación que aporta PhoneGap es bastante extensa, y viene clasificada para cada una de las versiones que van sacando, y como soporte añadido hay foros donde los desarrolladores intercambian sus experiencias.

PhoneGap está formada por las API que permiten utilizar las funcionalidades nativas de los terminales sobre las que se podrá desplegar la aplicación desarrollada. Actualmente PhoneGap en su versión 3.4 (Marzo 2014) permite desarrollar para iOS, Android, BlackBerry, Windows 8, WindowsPhone 7 y 8, Amazon-fireos, Firefox OS, Ubuntu y Tizen.

Para hacer uso de estas APIs el desarrollo se realiza sobre tecnologías web estándar, como son HTML5, JavaScript y CSS. La aplicación final no correrá sobre un navegador sino que irá empaquetada, gracias a esta idea podrá invocar características nativas de los terminales, como contrapartida se necesita que el navegador del terminal soporte estas funcionalidades.

Para conseguir este encapsulamiento, podemos realizar el despliegue de la aplicación sobre el IDE con el que desarrollamos, muy útil para realizar desarrollo y pruebas, pero poco práctico para el despliegue en distintas plataformas móviles, ya que cada dispositivo móvil tiene un entorno de desarrollo distinto (para dar soporte a tres de las tecnologías más utilizadas, tendríamos que tener un PC con eclipse y Visual Studio, y un MAC con el IDE Xcode). Otra opción nos la proporciona PhoneGap Build, que nos dará la posibilidad de desplegar la aplicación en la nube para los dispositivos que soporte la versión de PhoneGap seleccionada. Como inconveniente, soporta las plataformas más importantes, pero no todas las que indican en la documentación, para estos casos hay que utilizar el correspondiente IDE.

En primer lugar decidiremos sobre qué plataforma queremos empezar a trabajar, y en un segundo paso decidiremos sobre qué otros sistemas queremos extender nuestra aplicación. Los entornos de desarrollo serán diferentes en función de la plataforma que elijamos:

Para Android, usaremos eclipse (sobre cualquiera de sus arquitecturas, Windows, Mac y Linux). Para iOS, usaremos Xcode, solo disponible para Mac, para BlackBerry, usaremos eclipse y finalmente para WindowsPhone, usaremos Visual Studio.

Si bien para el resto de plataformas no hemos encontrado información al respecto, la situación más habitual para un desarrollador es desplegar las aplicaciones pensando en Android y iOS, donde se encuentra actualmente concentrada la mayor cuota de mercado.

Una vez elegido el IDE y el paquete de PhoneGap sobre la plataforma base a desarrollar, empezamos a crear nuestra aplicación con HTML5, JavaScript y CSS. Para desplegar la aplicación la forma más cómoda es utilizar PhoneGap Build, debido a que evitamos tener que compilar en nuestra máquina, ganamos tiempo y minimizamos recursos, además evitamos el IDE correspondiente para cada una de las plataformas móviles. En caso de no utilizar PhoneGap Build, si desarrollamos para Android e iOS deberíamos disponer de un Mac (ya que desde un PC no podemos compilar para los dispositivos iOS), y compilar con la librería de PhoneGap para Android y después con la librería para iOS.

Respecto a la depuración, por un lado tenemos la ventaja de que el desarrollo es sobre plataformas conocidas, HTML5, JavaScript y CSS, y se pueden desarrollar y testear sobre herramientas tales como Dreamweaver o Aptana. Si bien todas estas pruebas no serán suficientes ya que tenemos características nativas que solamente se pueden testear sobre el propio terminal o con los emuladores. Para poder realizar estas pruebas sobre los emuladores necesitaremos el eclipse, JDK y Android SDK (En el Capítulo 7 explicamos cómo hacer uso de estas herramientas).

Arquitectura

Podemos dividir la arquitectura en tres capas (Figura 12). El lenguaje en que desarrollaremos la aplicación, en PhoneGap es la combinación de HTML5 + CSS3 + JavaScript. Las dos capas siguientes son las que van a permitir la comunicación de ambos entornos, la capa superior (HTML5 + CSS3 + JavaScript) con la capa de funcionalidades o características de los terminales (Windows Phone, iOS, Android). La capa Cordova JS API, estará desarrollada en JavaScript, y en cambio la capa de Cordova nativa, será específica del terminal, es decir, para el caso de Android esta librería estará implementada en Java. Además, podemos añadir plugins a nuestra aplicación como una librería más, lo que facilita la reutilización de código.

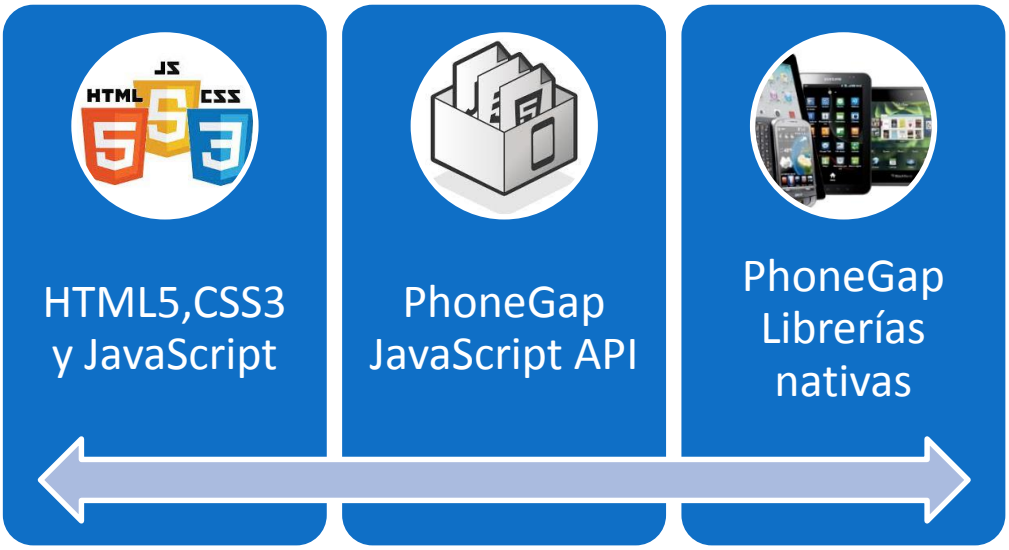


Figura 12. Arquitectura PhoneGap

5.2. Titanium

Introducción

AppCelerator Titanium es una plataforma para el desarrollo de aplicaciones móviles y de escritorio, libre y de código abierto. Su salida al mercado fue en 2008, y permite desarrollar para Android, iOS y BlackBerry, Tizen, Denso y Web móvil.

Características

A diferencia de otras herramientas, aunque AppCelerator basa su programación en JavaScript, HTML y CSS, el código final es un código nativo.

AppCelerator se divide en tres bloques:

- ***Analítico***, permite tener una visión real de la cartera de aplicaciones móviles. Para ello dispone de cuadros de mando para los ejecutivos de TI, responsables de negocios, desarrolladores, tester y miembros del equipo del proyecto.

- ***APIs***, Titanium proporciona un Mbaas (mobile backend as a service), que permite acceder a base de datos heterogéneas a través de servicios en la nube. Se divide en dos grupos:

    - Conectores para empresas, crear nuevas integraciones empresariales permitiendo la movilidad de los datos de la empresa
    - Conectores públicos, que facilitan el acceso a los servicios móviles comunes tales como Linkedin, PayPal, DropBox, Facebook, etc. eliminando la complejidad de la gestión de múltiples APIs

- ***Desarrollo***: El IDE utilizado está basado en eclipse, y los lenguaje utilizados son HTML, CSS y JavaScript. Titanium dispone de una plataforma cruzada de JavaScript APIs. La idea es que el desarrollador realmente está programando una aplicación nativa, ya sea Java para Android o Objective-C para iOS, pero a través del lenguaje JavaScript.

Las divisiones de Titanium son las siguientes:

- ***SDK Titanium***: comprenden un conjunto de utilidades basadas en Node.js y herramientas de apoyo para las herramientas del SDK nativo. La combinación del código JavaScript generado por el desarrollador, el intérprete de JavaScript y los assets (objetos) estáticos da lugar a un código binario que se podrá ejecutar sobre un emulador o dispositivo móvil.

- ***APIs de Titanium***: conjunto de APIs desarrolladas en JavaScript, que permiten tener acceso a un gran número de UI nativas y componentes no visuales. Se dividen en varias áreas destacando Titanium UI y Titanium Network.

- ***Titanium Studio***: es la herramienta para crear, probar y depurar las aplicaciones móviles. Integra plantillas y aplicaciones ejemplo.

Los servicios en la nube se proporcionan en el back-end, e incluyen una serie de análisis para el desarrollador, como frecuencia de uso de aplicaciones y plataformas. Mediante una API central, el desarrollador puede registrar eventos de análisis personalizados.

Respecto a la depuración, Titanium se queda un poco lejos de lo esperado, no presenta ninguna herramienta, así que hay que armarse de paciencia y utilizar la llamada T.API.info(), para realizar el seguimiento del código.

Si es cierto, que dentro del market de Titanium, hay una herramienta, Cloudebug, diseñada para una depuración remota AppCelerator Titanium.

Arquitectura

En la figura 13 vemos una descripción a alto nivel de la arquitectura. El desarrollador programa en JavaScript y HTML, y son las APIs de Titanium las que se encargan de convertir estas llamadas en nativas.

Debido a que el resultado final es una aplicación nativa, el rendimiento obtenido es el mismo que si estuviésemos desarrollando con las herramientas nativas.

Las plataformas para las que está disponible Titanium son Android, iOS y BlackBerry, Tizen, Denso y Web móvil.

Figura 13. Arquitectura Titanium

Otras características:

AppCelerator Alloy, sigue la arquitectura MVC, y proporciona independencia entre el interfaz de usuario, los modelos de datos de aplicación y lógica de negocio, facilitando la creación de aplicaciones móviles de alta calidad.

5.3. Genexus

Introducción

Genexus es un producto de la empresa Artech, fundada en 1989. Creada por Breogán Gonda (Facultad de Ingeniería de UdelaR, en la Pontificia Universidad Católica de Porto Alegre (Brasil) y en la Universidad Católica del Uruguay) y Juan Nicolás Jodal (fue profesor en la Universidad Católica del Uruguay). [10]

El proyecto Genexus surge como necesidad de resolver un proyecto de gran envergadura, donde se asumen cambios continuos y el reto fue, crear una herramienta que creara las aplicaciones y reducjera el mantenimiento.

Características

Genexus tiene un enfoque distinto a la mayoría de las herramientas o frameworks que podamos encontrar en el mercado. Las herramientas ofrecen un encapsulamiento como PhoneGap o generan código nativo como Titanium, pero Genexus sube un escalón más.

Genexus ofrece a los desarrolladores no solamente una única manera de programar las aplicaciones, sino que el lenguaje utilizado es principalmente declarativo. Más tarde, se podrá implementar en los diferentes ambientes (servidores, PCs y *dispositivos móviles*.) y lenguajes (Ruby, Java, C#, Cobol.). [11]

Centrándonos en el mundo móvil, actualmente Genexus ofrece soporte para las plataformas Android, BlackBerry e iOS. Ofrece una serie de características comunes a estas tres plataformas, de las que podemos destacar:

- Integración con redes sociales
- Un único login para todas las aplicaciones
- Posibilidad de generalas aplicaciones en HTML5

El entorno de desarrollo para plataformas móviles es el mismo IDE (figura 14) que para el resto de entornos. La herramienta Genexus no es de uso libre, y debido a que no tenemos que aprender un nuevo lenguaje de programación, la curva de aprendizaje es pequeña.

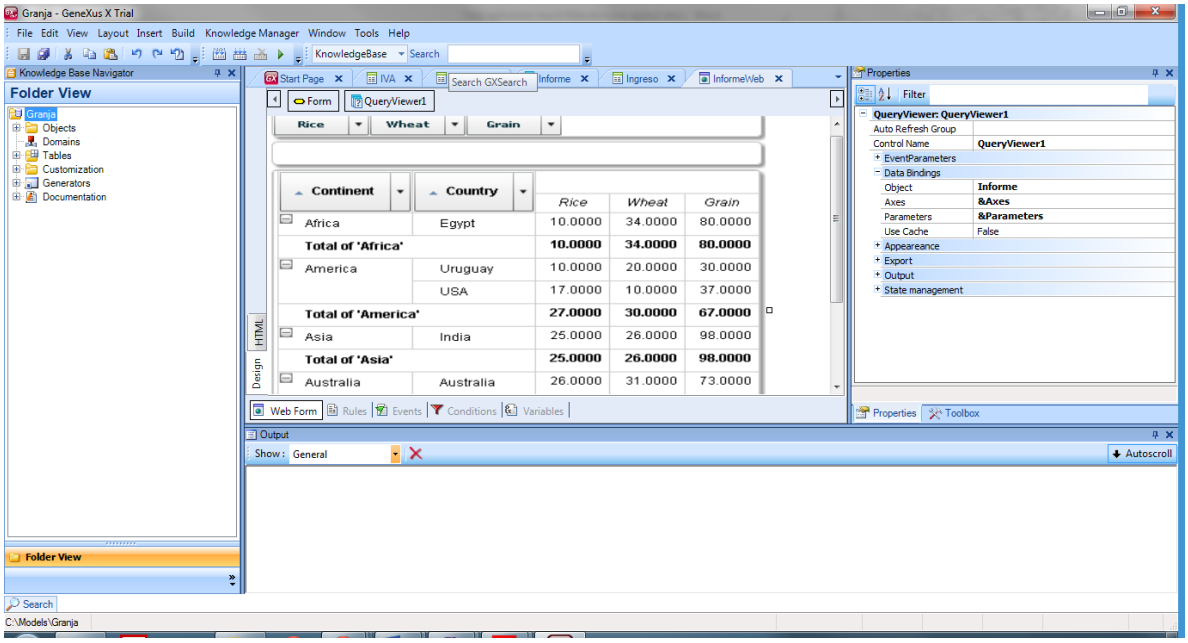


Figura 14 : IDE de Genexus

Su principal valor añadido es el que para crear las aplicaciones, el código será lo mínimo posible, el desarrollador tendrá que crear lo que en Genexus llaman base de conocimiento, en base a la visión de cada uno de los usuarios. Ante cambios en el modelo Genexus genera un informe de impacto, para conocer como las nuevas funcionalidades pueden modificar la base de datos y las aplicaciones, informando de restricciones de integridad o redundancia.

Arquitectura

La arquitectura de Genexus para dispositivos móviles realmente no difiere del resto de entorno, si bien es cierto que la información fluirá de otra manera, incluso dependiendo del dispositivo, Smartphone, Tablet, en este caso el tamaño sí que importa.

Detallaremos brevemente la arquitectura de Genexus que aparece en la Figura 16.

El primer punto de inicio para crear un prototipo, sería crear los objetos básicos:

- ***Transacciones***, nos permitirán crear los objetos de la realidad que el usuario conoce. Las distintas visiones de los usuarios deben verse reflejadas en estas transacciones. Genexus a partir de las transacciones deducirá que tablas crear. Por ejemplo, si creamos la transacción factura, incluiremos los campos de línea de factura, pero Genexus genera 2 tablas, una para facturas y otra para líneas de

factura, con su correspondiente relación de uno a muchos. Figura 15.

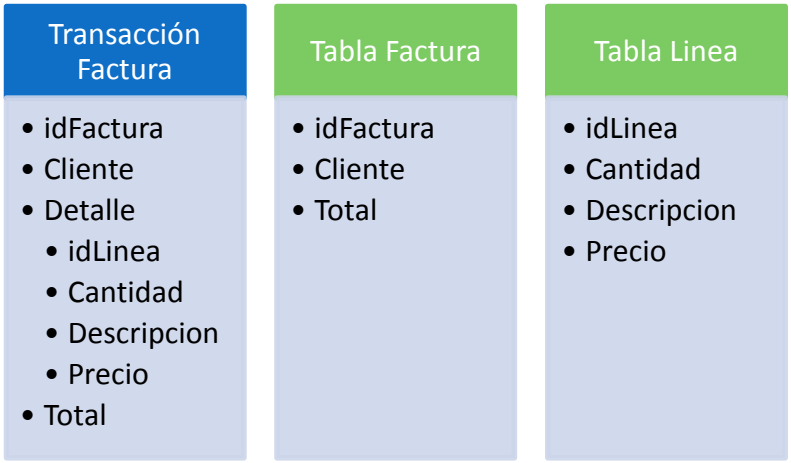


Figura 15. Transacciones Genexus

- ***Procedimientos***, es un objeto que permite navegar sobre las tablas y bajo determinadas condiciones modificar sus valores, además de poder generar un listado de sus datos a partir de algún criterio.

- ***WebPanels***, es un objeto que permite realizar consultas interactivas con la base de datos, podemos indicar los criterios de filtrado de la información, y será Genexus quien infiera sobre que tablas debe recuperar la información.

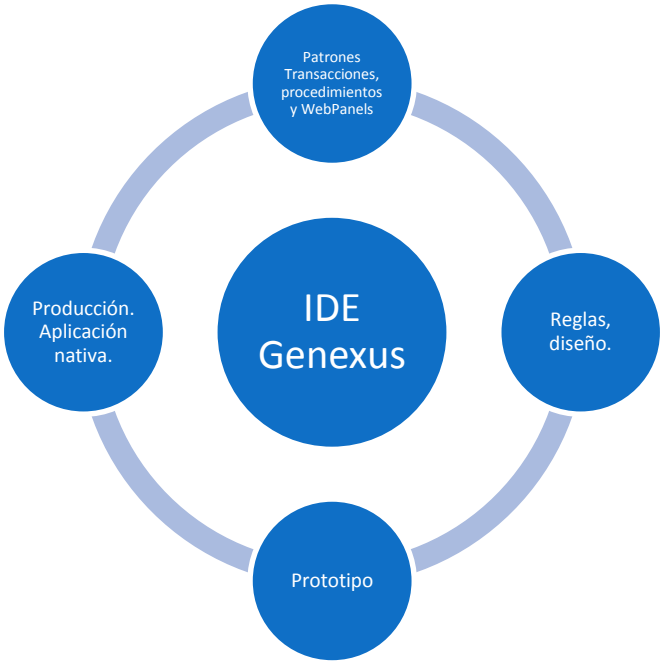


Figura 16. Arquitectura Genexus

Una vez creado, podemos añadir reglas, como por ejemplo que cuando se pida una determinada fecha tenga por defecto la fecha actual. Lo interesante es que el desarrollador solamente tiene que describir la regla y Genexus se encargará de lanzar la regla (también depende si la regla se asocia a un objeto o a todo el proyecto). Referente al diseño, Genexus proporciona una interfaz por defecto, que se podrá modificar de acuerdo a las necesidades de nuestra aplicación.

Durante el desarrollo de la aplicación, Genexus nos solicitará sobre qué ambiente y lenguaje vamos a desarrollar, más tarde podremos cambiar estos parámetros, pero es necesario especificar unos valores durante el desarrollo para generar los prototipos y que los usuarios puedan validar el desarrollo. Esta metodología incremental asegura que el producto cuando llega a explotación ya ha sido validado por los usuarios durante todo el ciclo.

El despliegue de la aplicación se realizara una vez tengamos todos los requerimientos implementados y los usuarios hayan dado el visto bueno a la aplicación. Pero como las empresas son cambiantes, podemos volver al punto inicial, y modificar el modelo con los nuevos requerimientos.

5.4. RhoMobile Suite

Introducción

RhoMobile, es un conjunto de herramientas (RhoStudio, RhoElements, RhoHub, Rhodes, RhoConnect y RhoGallery). Es un marco de trabajo de código abierto desarrollado por Motorola.

Al tratarse de un desarrollo independiente del sistema operativo, las aplicaciones pueden estar orientadas tanto para el mundo empresarial como para el público en general, si bien es cierto que dan un gran soporte a una gran parte de los dispositivos Motorola, cuyo uso es empresarial.

Características

De la suite RhoMobile, nos centraremos en el producto Rhodes, ya que sobre el recae el peso del desarrollo y generación de aplicaciones. Detallaremos brevemente el resto de elementos de la suite RhoMobile. [12]

- *RhoStudio*, simulador que permite probar y depurar las aplicaciones de las distintas plataformas móviles desde la misma herramienta.

- *RhoElements*, a través del conjunto de herramientas de HTML5, permite elementos que tengan un mismo comportamiento independientemente del hardware (plataforma y tamaño de pantalla).

- *RhoHub*, conjunto de servicios en la nube, que permiten el desarrollo, compilación y acceso a datos remotos.

- *Rhodes* es un framework de código abierto basado en Ruby. Permite crear aplicaciones nativas para las principales plataformas existentes en la actualidad (Android, iOS y Windows Phone).

- *RhoConnect*, motor de sincronización de dispositivos y base de datos transparente al usuario.

- *RhoGallery*, sistema de administración de aplicaciones, permitiendo gestionar por dispositivos las aplicaciones que pueden instalarse, con un enfoque empresarial.

La suite RhoMobile instala sus componentes sobre el IDE eclipse si la instalación es sobre un PC o Xcode si trabajamos sobre un Mac. Además será necesario el componente de Ruby para poder hacer uso de la suite RhoMobile.

La principal característica es que es el uso de MVC (**M**odelo **V**ista **C**ontrolador), es un patrón de arquitectura que permite separar los datos y la lógica de negocio de la interfaz de usuario, con el objetivo de ser más fáciles de entender y mantener. [13,14].

Arquitectura

Nos centraremos en la arquitectura del módulo Rhodes, para simplificar el diagrama y centrarnos en la parte de desarrollo.

Podemos dividir la arquitectura del módulo Rhodes, en tres niveles como aparece en la figura 17. La etiqueta definida como Rhodes app, será la capa donde el desarrollador

implementara la aplicación haciendo uso del lenguaje html y Ruby.

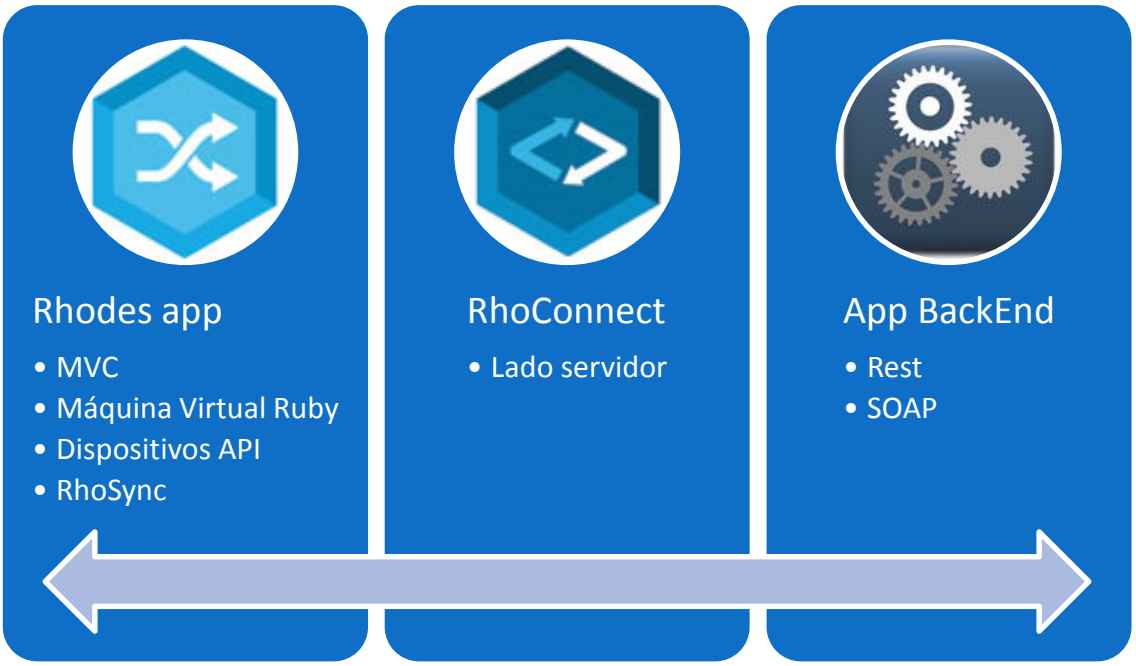


Figura 17. Arquitectura Rhodes

El proceso de desarrollo seguirá el patrón MVC, como ya hemos comentado. Para poder acceder a las funcionalidades internas de cada dispositivo, como son cámara, agenda entre otros, la máquina virtual de Ruby será la encargada de esta faena.

El acceso a la información se realizará con el módulo RhoConnect, que permitirá sincronizar la información y trabajar con los datos incluso sin conexión. Esta capa permite el acceso a servicios web de manera sencilla y transparente.

5.5. Elección de herramienta

El mercado nos ofrece una variedad importante de herramientas con solución multiplataforma, (PhoneGap, Titanium, Genexus, RhoMobile, Adienc, Velneo, Telerik, Xamarin, Appery.io entre otras) y ante inviabilidad de estudiarlas todas para decidir las más atractivas y que aportarán un valor añadido, hemos seleccionado por un lado las más reconocidas por su trayectoria: (PhoneGap y Titanium), muy similares entre ellas, las otras dos son herramientas con una estructura distinta al resto, (Genexus con una programación descriptiva y RhoMobile con un paquete de herramientas y muy pensada para el mundo empresarial).

PhoneGap ofrece un entorno sencillo, se centra en proporcionar el acceso a las características de los dispositivos, dejando en manos del desarrollador que IDE utilizar (Eclipse, Xcode, Adobe Dreamweaver), que herramienta para el interfaz de usuario trabajar (JQuery, Sencha, Kendo). Este abanico de posibilidades es un arma de doble filo, por un lado permite al desarrollador seleccionar las herramientas con las que se encuentra más cómodo, por otro tenemos muchas combinaciones a las que deberíamos tener soporte. Titanium es la más parecida a PhoneGap, su principal diferencia, que no es poca, es que PhoneGap no genera un código nativo y Titanium sí, respecto a su programación (HTML y JavaScript) sobre un IDE que suele ser eclipse, las hace similares, pero además Titanium incluye módulos para programar en Ruby. En cambio Genexus tiene un planteamiento totalmente distinto con su programación con modelos, siendo la única de las comparadas que es de pago. RhoMobile tiene un planteamiento en que la herramienta está formada por un conjunto de módulos bien diferenciados, que aunque da robustez, para pequeños proyectos podría resultar complejo.

La curva de aprendizaje, a priori, favorece a PhoneGap y Titanium, con un desarrollo más similar a la programación web, lo que permite que muchos desarrolladores se sientan más a gusto con estas herramientas. Genexus basado en un lenguaje descriptivo y de reglas, supone un nuevo reto, aunque estamos en un nivel de abstracción superior, permite dedicar más tiempo a la lógica y olvidarnos de programar código, o con RhoMobile con Ruby. Durante el desarrollo de la tesis pude asistir a un taller de Genexus, y la apreciación personal es que no debe ser ningún obstáculo desarrollar en Genexus, ya que con unas nociones básicas implementamos una aplicación con unas funcionalidades básicas en una sola mañana, es cierto que el problema no era complejo, pero tareas rutinarias como son navegación de pantallas, alta, borrado, modificar y filtros sobre base de datos, no tuvimos que escribir ni una sola línea de código.

Documentación y comunidad es otro punto interesante a tener en cuenta. Todas ellas ofrecen un alto soporte con documentación de cada una de las versiones, ejemplos con acceso al código, e igualmente tienen sus propios foros para la resolución de dudas y planteamiento de nuevos retos. La diferencia que apreciamos es que PhoneGap y Titanium ofrecían una comunidad más amplia, ademas PhoneGap incluso tiene su propia comunidad de habla hispana. Genexus en cambio no esta tan expandido a nivel mundial, y en España es poco conocida. RhoMobile, carece igualmente de ese seguimiento masivo

en sus foros. Por el contrario, estas dos herramientas tienen un enfoque más empresarial, y su comunidad parece ser más especializada.

Finalmente decidimos realizar la aplicación con PhoneGap, valorando positivamente el uso de un lenguaje bien conocido y consolidado y de su gran comunidad.

En la tabla siguiente vemos una comparativa de los ítems más interesantes para cada una de las herramientas.

	PhoneGap	*Titanium*	*Genexus*	*RhoMobile*
IDE	Eclipse/ XCode/ Visual Studio/ Dreamweaver	Eclipse/ XCode	Genexus	Eclipse/ Xcode
Lenguaje	HTML5 + JavaScript + CSS3	HTML5 + JavaScript + CSS3	Lenguaje declarativo	Ruby
Plataformas	Android/ iOS/ BlackBerry/ Winodws Phone	Android/iOS/BlackBerry	Android/iOS/ BlackBerry	Android/iOS/ BlackBerry
Depuración	No integrado (Weinre)	No integrado (Pasos en la documentación)	DebuGx	RhoStudio
Documentación	Documentación por versiones, ejemplos, foro	Documentación por versiones, ejemplos, foro	Documentación por versiones, ejemplos, foro, videotutoriales	Documentación por versiones, ejemplos, foro
Coste	Libre	Libre	De pago	Libre

Capítulo 6. Desarrollo móvil en Shared Context Awareness

En este capítulo presentaremos la propuesta de TUNE-UP para dispositivos móviles, atendiendo a los requisitos del Shared context awareness. Facilitaremos el uso de la aplicación centrándonos en la información más importante para los agentes en situaciones de desplazamiento o cuando el agente esta fuera de su entorno de trabajo, pero necesita de conectividad.

Además a este requisito no hay que perder de vista que estamos sobre dispositivos con tamaños de pantalla limitados, por lo que hay que analizar cuál es la información necesaria y suficiente para el agente.

6.1. Aplicación modelo Endsley a TUNE-UP

Como encajan estos niveles según el modelo ***Endsley*** en una herramienta como TUNE-UP.

- ***Percepción***, En la herramienta de TUNE-UP podemos ver en su pantalla principal, el estado de las unidades de trabajo, alarmas, mensaje y notificaciones. (Figura 18). La información es referida a todos los agentes o a uno determinado, y de las unidades de trabajo relacionadas con un agente. En otras secciones de TUNE-UP podemos acceder a las unidades de trabajo y ver cómo están relacionadas con otras unidades de trabajo (módulo Gestor de requerimientos de TUNE-UP Software Process), o conocer los márgenes de tiempo de las unidades de trabajo.

- ***Comprensión***, en el caso concreto de las unidades de trabajo, podemos atender a las asignaciones de las unidades de trabajo, conocer si han ocurrido cambios, y realizar nuestra tarea asignada en los tiempos recomendados. Una unidad de trabajo que parece no cumplir los plazos, puede ser reasignada, cambiada de versión, o dividida en varias unidades de trabajo. Respecto a mensajes, podemos ver tanto los que nos involucran de forma concreta y responder, quedando toda la información registrada, para que cualquier miembro del equipo que trabaja en el workflow pueda conocer toda la información que le pueda ser de su interés.

- ***Proyección***, TUNE-UP dispone de un seguimiento de iteración, el *Workflow* figura 3, que ya hemos comentado con más detalle en el Capítulo 2, que facilita el modelo mental compartido, para que todos los miembros del equipo puedan interpretar la información de forma homogénea. Esto dota al sistema de mayor flexibilidad y dinamismo, pudiendo actuar sobre actividades antes de que sufran retrasos. Otros puntos importantes que ayudan a este modelo mental de equipo son el *Cuadro de mandos* y *Versiones y seguimiento*.

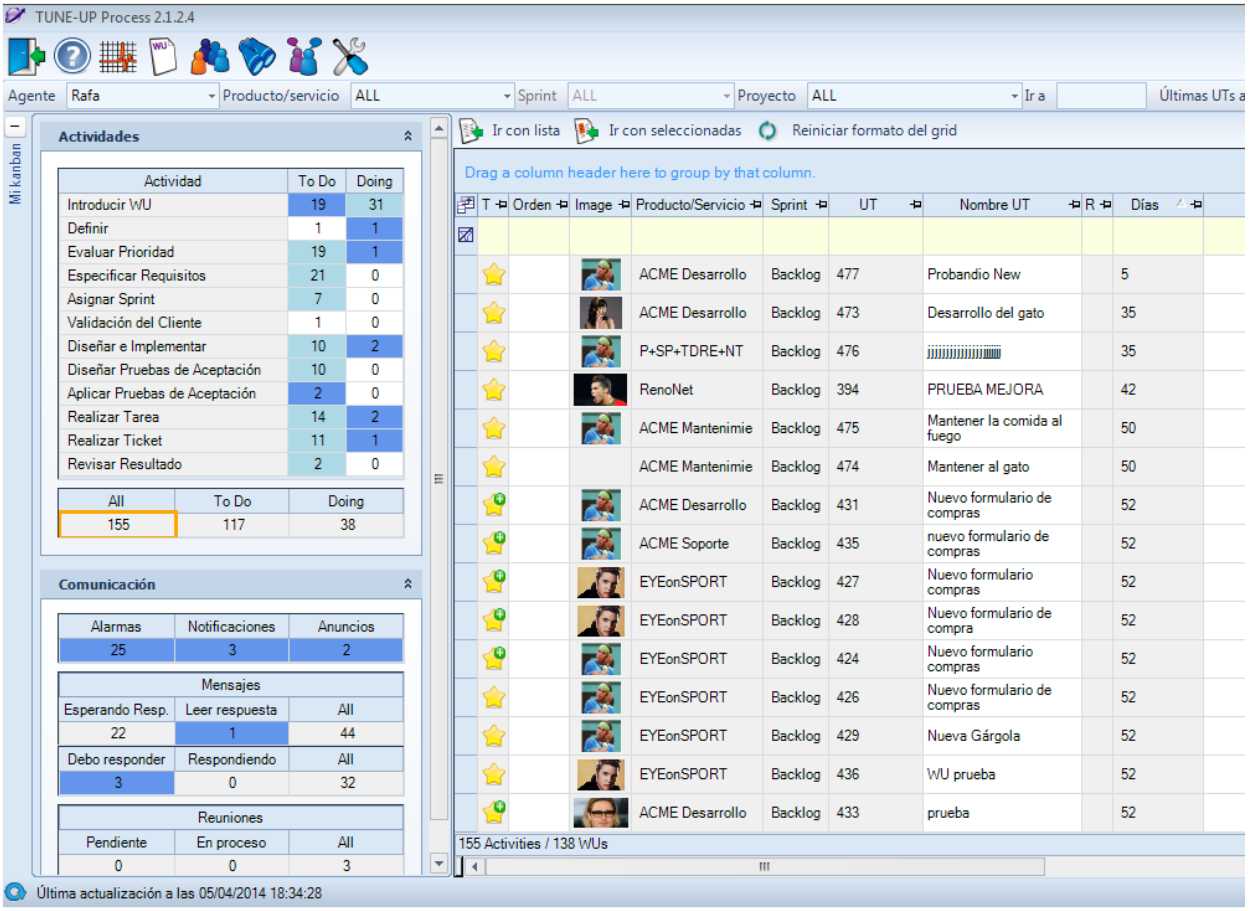


Figura 18. Pantalla principal TUNE-UP

6.2. Aplicación medidas de Situational Awareness a TUNE-UP

TUNE-UP no dispone de un seguimiento para el conocimiento de medidas, y tampoco era objetivo el estudio de estas medidas sobre TUNE-UP. Entendiendo que las metodologías ágiles son altamente dinámicas y cambiantes, no resultaría tampoco recomendable parar las tareas para realizar evaluaciones. Lo que sí que ofrece es una serie de herramientas para conocer el estado de proyectos, sprint, tareas.

TUNE-UP Software Process permite conocer en cualquier momento el estado de la versión y de cada una de las WU de la versión, otra forma es con las pruebas de aceptación, podemos evaluar objetivamente el nivel de corrección y finalización de las WUs, pero además podemos saber si los plazos a los que se comprometieron el equipo de trabajo son adecuados para la finalización de la versión.

Veamos con más detalle los elementos que definen el nivel de corrección y finalización que implementa TUNE-UP

- ***Gestión del producto***, la herramienta TUNE-UP ofrece la posibilidad de conocer en tiempo real la situación del proyecto mediante la "Gestión de estructuras", "Backlog", "Flujo acumulado", "Esfuerzo invertido", "Contenido del sprint", "Dashboard del sprint", "Tendencias de sprints". Podemos encontrar tablas y gráficas que permiten evaluar el rendimiento de forma objetiva de cada una de las unidades de trabajo para cada uno de los agentes o revisar a nivel de versión, podemos conocer el tiempo estimado para implementar una unidad de trabajo, y el tiempo que hay dedicado, la actividad en que se encuentra la unidad de trabajo, y como se ha movido dentro del workflow (ya que la propia dinámica del workflow permite navegar adelante, saltando actividades si así se cree conveniente por los requisitos propios de la unidad de trabajo, o navegar hacia atrás, para correcciones, o nuevos requisitos que se deben abordar).

- ***Pruebas de aceptación***, nos permiten saber el nivel de cumplimiento de la unidad de trabajo, extrapolándolo a todo el proyecto, se conoce con cada versión el porcentaje de producto acabado, información de vital importancia para el cliente, y para el grupo de desarrollo, ya que permite tomar medidas para corregir eventuales retrasos (por ejemplo, reasignar agentes a tareas aun no finalizadas).

6.3. TUNE-UP-móvil en el contexto Situational Awareness

Ahora que hemos contextualizado la ***Situational Awareness*** en TUNE-UP, vamos a dar un paso más, y llevar este nuevo contexto a la propuesta de la herramienta TUNE-UP para dispositivos móviles. Tenemos que buscar en las funcionalidades de TUNE-UP aquellas que son importantes desde el punto de vista del grupo de trabajo, es decir, información que afecta al desarrollo de las tareas de los diferentes miembros del equipo de desarrollo.

Por otro lado, como estamos con una aplicación móvil multiplataforma, debemos ser precisos de la información que se quiere mostrar. De igual modo, que no resulta razonable, llevar toda la información que se muestra en la plataforma de escritorio de TUNE-UP a la plataforma móvil, hay que redefinir la información que se quiere dar adaptándola a los formatos de pantalla móvil.

Un módulo como es el gestor de unidades de trabajo (WUM), muy importante en TUNE-UP, no tiene la misma relevancia en un dispositivo móvil, este módulo ofrece un conjunto de tareas que en su mayoría implica estar sentado en una mesa con un entorno adecuado. En cambio encontraremos tareas que podemos realizar de camino a la oficina o a una reunión mientras nos desplazamos (en tren, avión bus, etc...). Tendremos que encontrar funcionalidades importantes de TUNE-UP y que sean compatibles en un contexto móvil y en ese escenario dar solución al contexto situacional.

Por otro lado podemos diferenciar los roles que serían más adecuados para determinadas funcionalidades. No hay que perder de vista que en los desarrollos software con metodologías ágiles los agentes desempeñan ambos roles .

- *Desarrollador*, su papel se centra en realizar las tareas asignadas dentro del workflow, interactuar con el resto de agentes en las tareas compartidas o interdependientes.

- *Supervisor*, el papel es de conocer el estado de la versión, la evolución de la unidades de trabajo, cumplimiento de las pruebas de aceptación y plazos previstos, su visión es más abstracta, la información que principalmente debería manejar sería la que hoy en día se conoce como cuadro mandos.

En los siguientes puntos planteamos la solución para las distintas funcionalidades, esta puede sufrir alteraciones referente a la cantidad y tipo de información mostrada para cada funcionalidad, por eso no debe resultar determinante la solución propuesta hasta que no pase todos los filtros y pruebas de aceptación necesarias.

Funcionalidades generales

1. ***Kanban***, Mostraremos las actividades pendientes y en progreso (figura 19 y figura 20), diferenciando con colores aquellas que solo pertenecen al agente (azul), actividades que realiza el agente y otros miembros del equipo (celeste) y el fondo

blanco para las que realizan los miembros de su equipo. El detalle de las tareas solo se mostrara del agente logueado. También puede suceder que tengamos actividades que no estén asignadas a ningún miembro del equipo, sería interesante poder verlas, porque puede suceder que hayan actividades que sean más prioritarias que las que tenemos asignadas en ese momento, y esta información es importante tenerla presente para cuando nos sentemos a trabajar. Además, se propone quedarse a un nivel superficial respecto a la información de las unidades de trabajo, de forma que sea la necesaria y suficiente para cuando se siente a trabajar tener claro como acometer el trabajo (Percepción-Comprensión-Proyección). Podemos observar en la figura 20, cómo la información de las actividades se ha reducido a tres campos para los dispositivos móviles respecto a los 21 campos que tiene la aplicación de escritorio, hay campos totalmente innecesarios, como es el campo imagen y nombre del agente, ya que solo veremos actividades del usuario logueado, por lo que sería una información redundante, por otro lado vemos que en la aplicación de escritorio visualizamos el kanban y las actividades, en cambio en la aplicación móvil solo vemos o el kanban (figura 19), o las tareas de la actividad seleccionada.

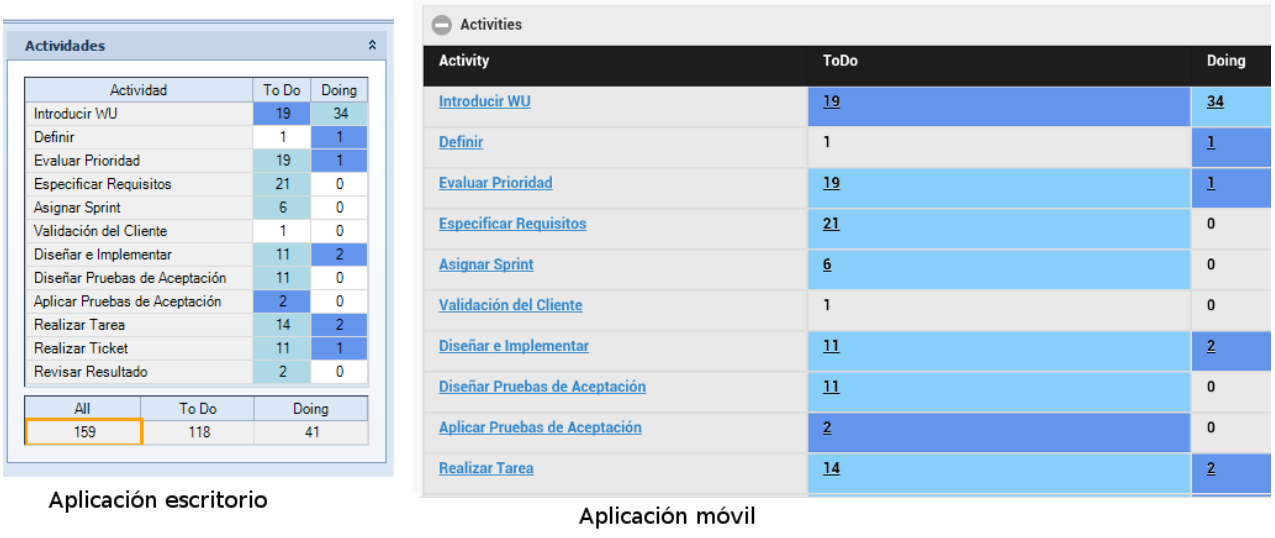


Figura 19. Kanban

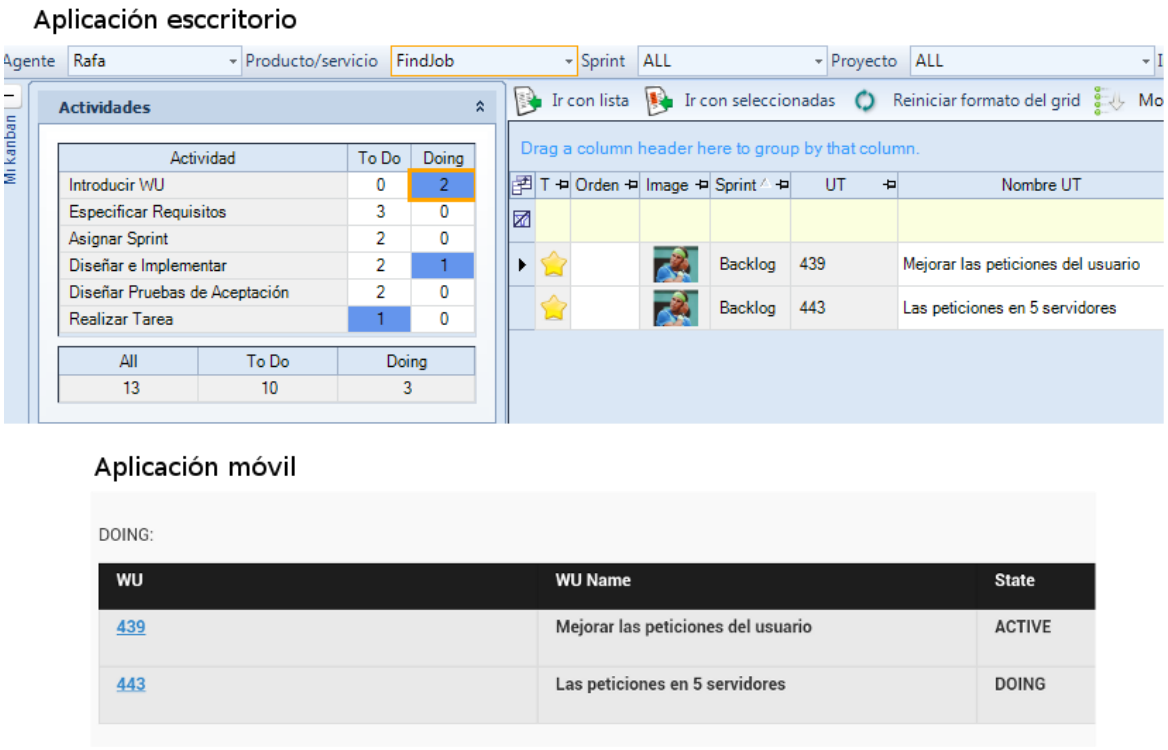


Figura 20. Actividades

No podemos olvidar el envió de mensajes para conseguir esa interrelación con los distintos agentes de un proyecto. La diferencia principal existente entre la versión escritorio y la versión móvil es que la segunda solo admite un agente, así

como que se considera que desde la aplicación móvil no es tan importante adjuntar documentos al proceso de envío de mensajes.

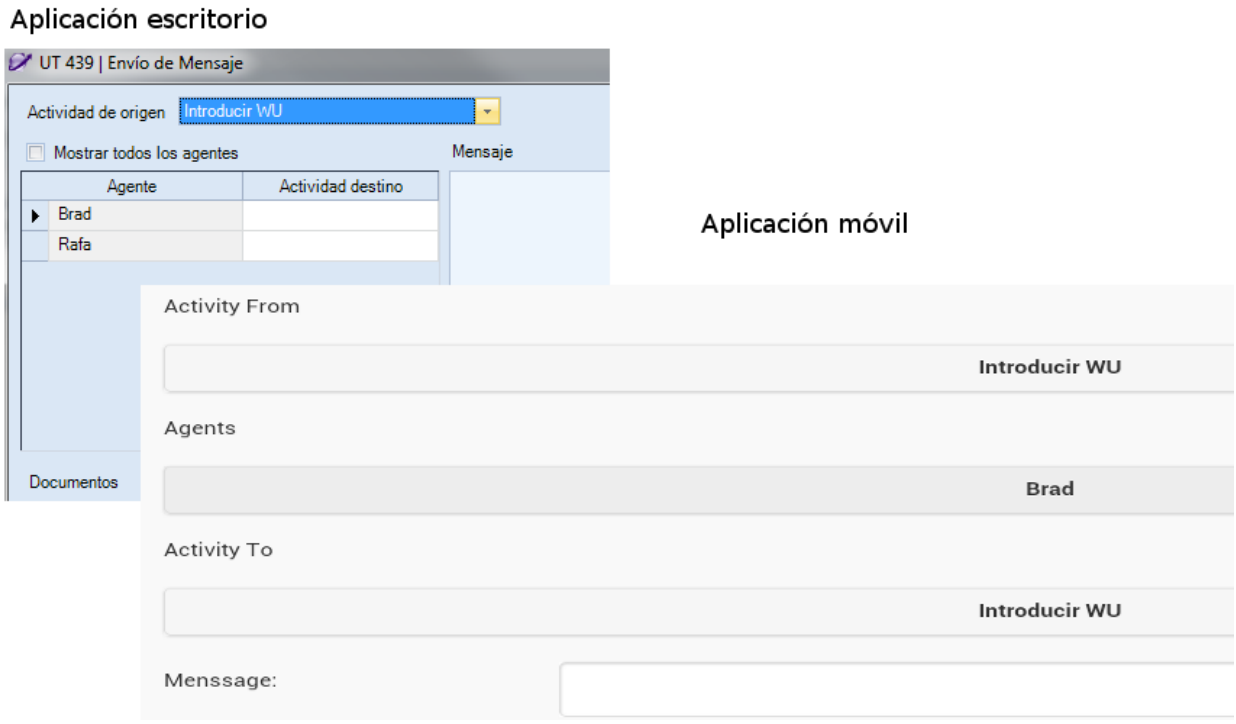


Figura 21. Envió de mensaje

2. ***Alarmas***, es un elemento muy importante a considerar, porque nos avisa de situaciones anómalas (figura 22), que en algunos casos podemos resolver inmediatamente. Las alarmas son generadas automáticamente, podemos dividir aquellas que podríamos resolver desde el dispositivo móvil, y las que necesitaríamos del lugar de trabajo para poderlas atender. Describiremos las situaciones que se pueden dar con la división comentada. [18]
    a. Solucionar a través del dispositivo móvil
        i. *"Sobrepasado el tiempo estimado en completar una actividad"*, podemos desde el dispositivo móvil, reestimar el tiempo.
        ii. *"El agente ha excedido el plazo para responder un mensaje"*, podemos responder al mensaje si contamos con toda la información necesaria.

iii. *"Se ha excedido el plazo para que un agente responda un mensaje enviado por el agente conectado"*, podemos volver a enviar un nuevo mensaje, o bien resolverlo con una llamada de teléfono.

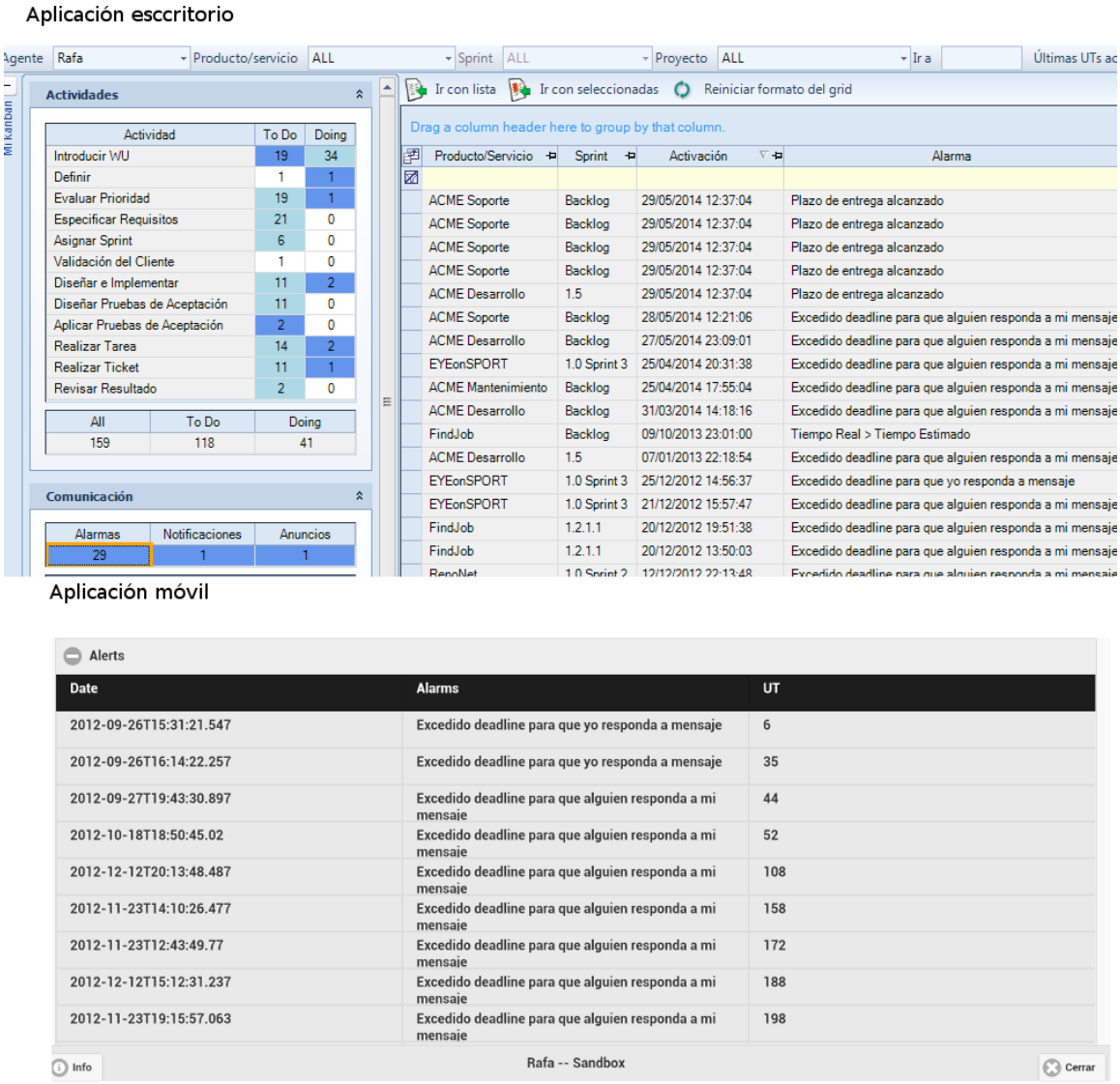


Figura 22. Alarmas

b. Solucionar desde el puesto de trabajo

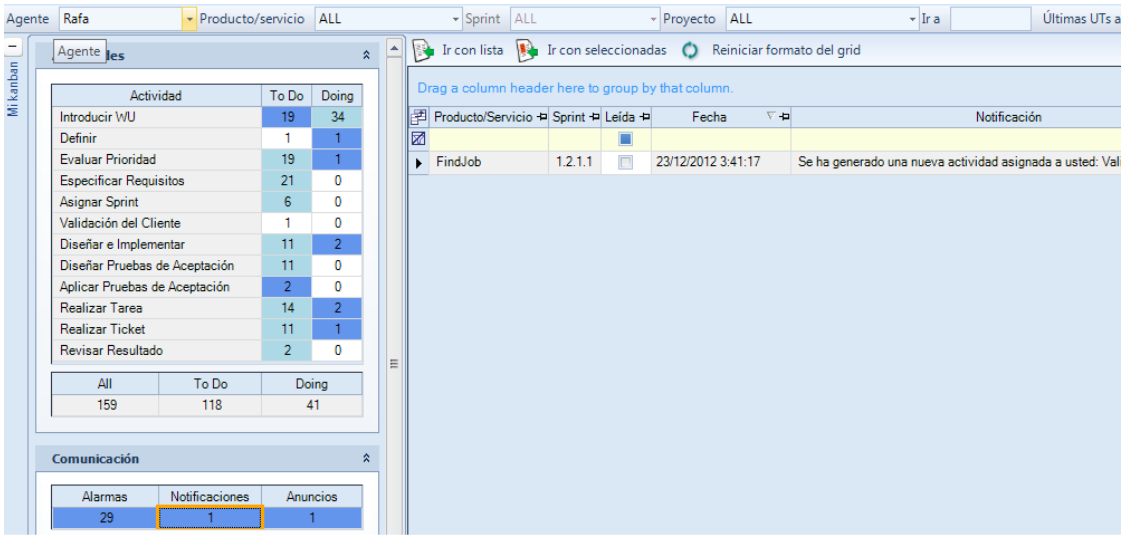
i. *"Sobrepasado el tiempo de postergación de una actividad en estado pendiente"*, solo se puede solucionar poniéndonos a trabajar en esta actividad

ii. *"Se sobrepasa el tiempo en proceso de una actividad"*, la única solución para esta alarma, es finalizar la actividad.

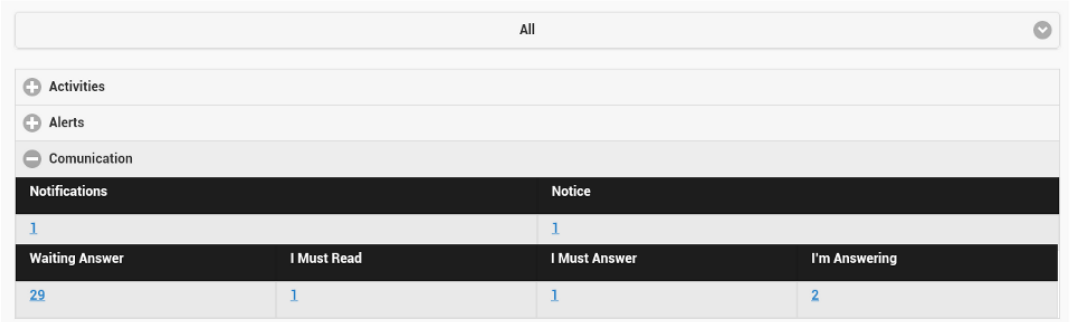
iii. *"El plazo de entrega de la UT se ha sobrepasado"*, igual que en los casos anteriores, hay que acabar la UT.

3. ***Notificaciones***, no tienen la relevancia de las alarmas, pero da información de cambios (Percepción), que ayuda al miembro del equipo a contextualizar las nuevas situaciones. Estos mensajes (figura 23) son generados automáticamente por el sistema, el usuario puede validar la notificación con una aceptación de lectura, de este modo desde el dispositivo móvil podemos atender las notificaciones. Podemos observar que desde el dispositivo móvil para acceder a la información, primero tenemos que visualizar el número de notificaciones desde la pantalla principal, apartado comunicaciones y en otra pantalla tendremos el resumen de las notificaciones. El check de leído de la aplicación escritorio se ha solucionado en la aplicación móvil con un link, con la misma funcionalidad en las dos aplicaciones. El número de campos es también sensiblemente inferior, pasando de 7 a 4 campos respectivamente.

Aplicación esccritorio



Aplicación móvil



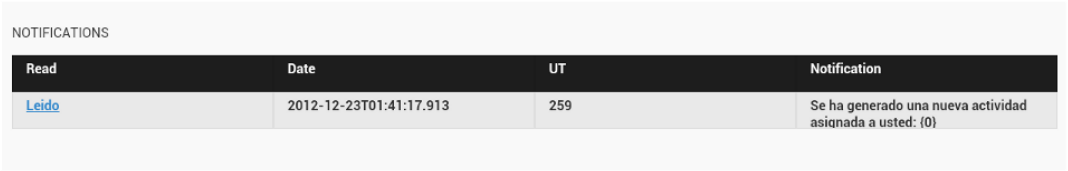


Figura 23. Notificaciones

4. ***Anuncio***, son mensajes masivos en los que no hay posibilidad de respuesta, de igual modo que sucede con las notificaciones se puede implementar desde el dispositivo móvil la confirmación de lectura, así como el proceso de enviar anuncios, para ello se seleccionarán los agentes que deben recibir el anuncio y una vez escrito el mensaje enviarlo. En la figura 24, vemos el resultado de consultar los anuncios, de igual modo que sucede con las notificaciones, la confirmación de lectura se realiza mediante un hiperenlace. No se ha implementado la funcionalidad del envió de anuncios, pero podemos ver en la figura 25, un boceto de esta funcionalidad. En la aplicación escritorio esta funcionalidad está formada por dos pantallas, el boceto en cambio se implementa todo en una sola pantalla, con el fin

de eliminar pasos de navegación. En este caso el diseño podría variar si consideramos que la lista de agentes es grande, y puede dificultar la selección de múltiples agentes, una posibilidad sería distribuir en dos grupos, por un lado los agentes seleccionados y por el otro los agentes no seleccionados.

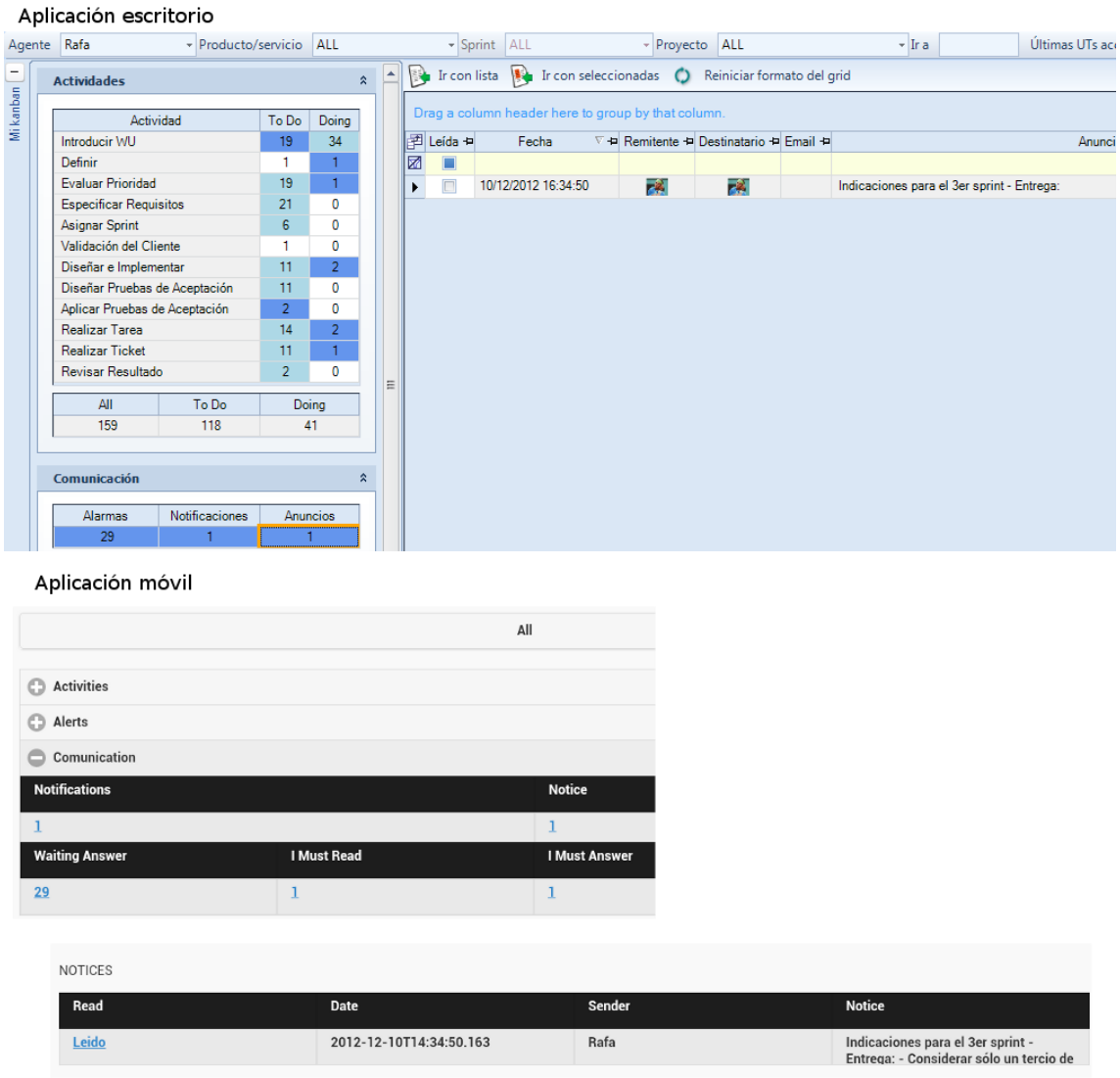
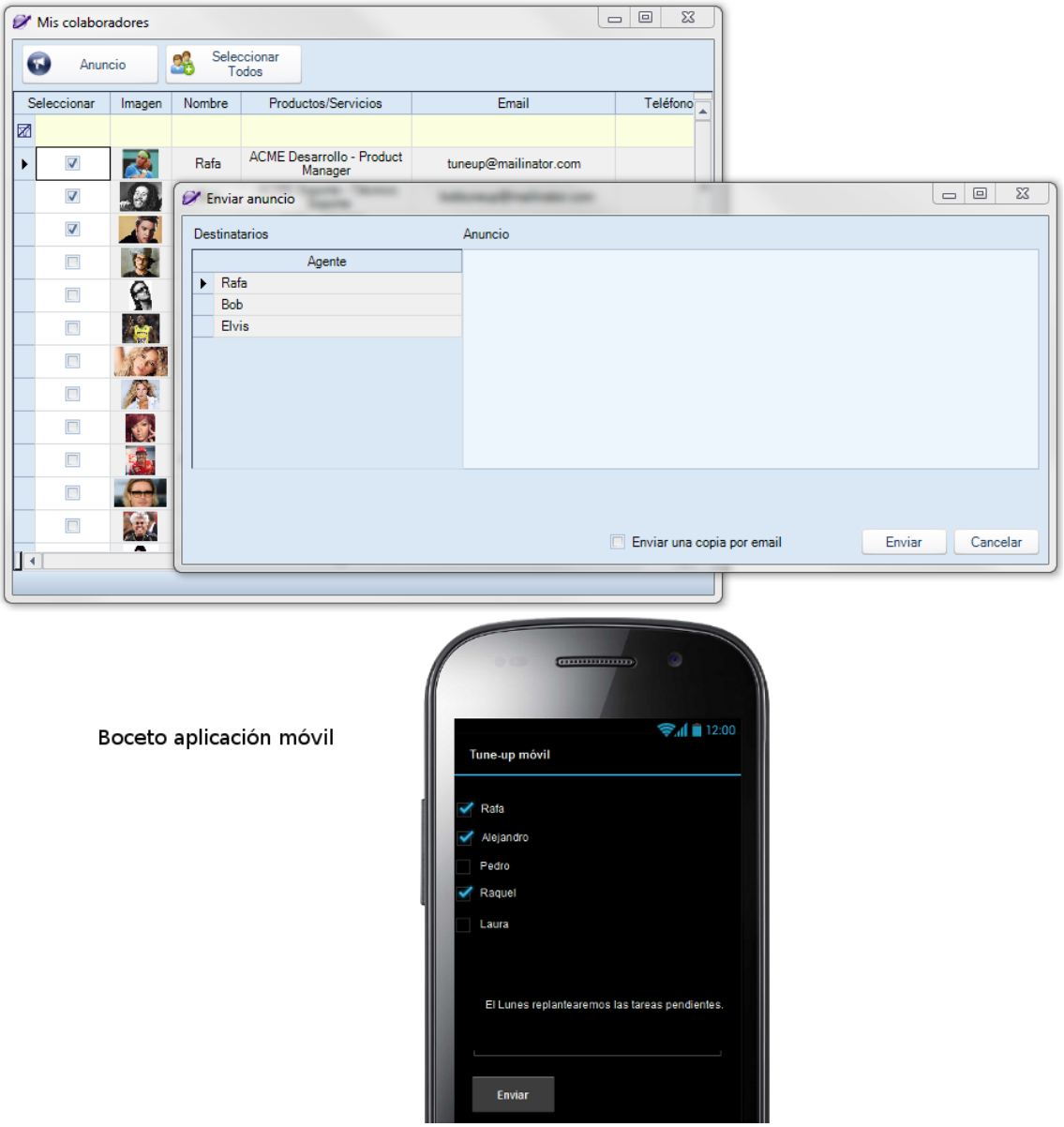


Figura 24. Anuncios

Aplicación escritorio



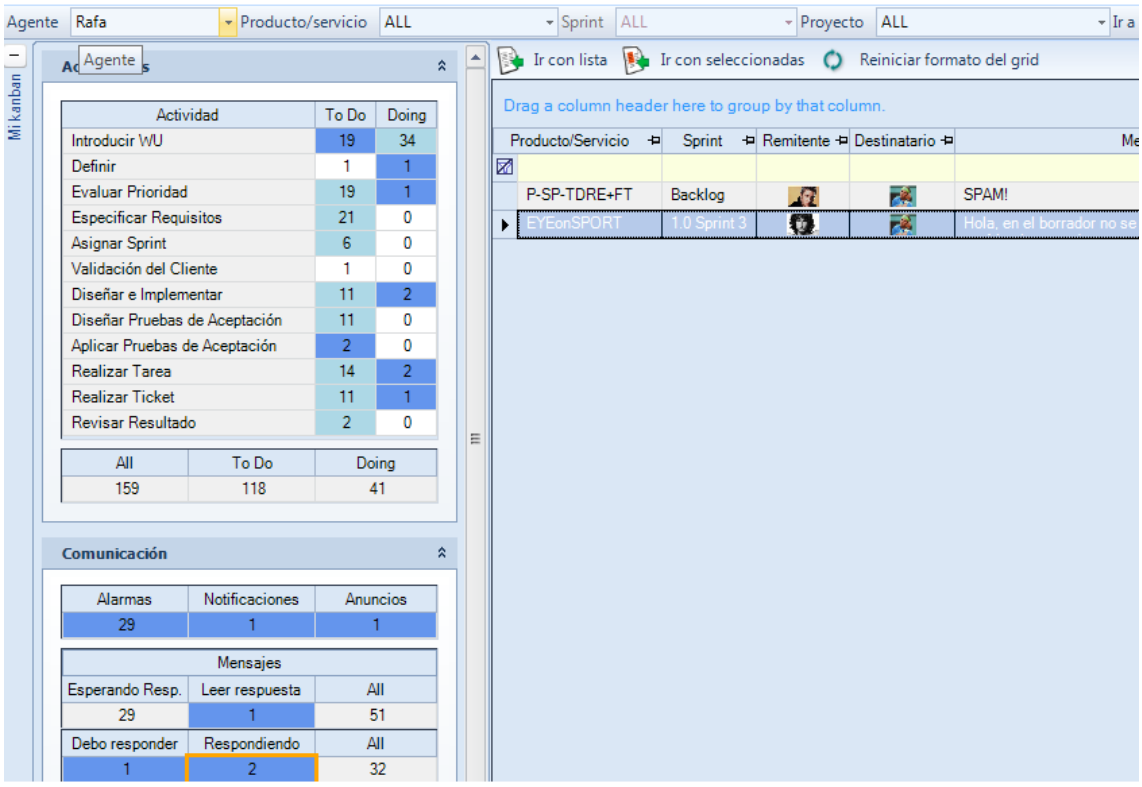
Boceto aplicación móvil

Figura 25. Enviar anuncio

5. ***Mensajes***, este punto es el más interesante en el contexto ***Situational Awareness***, ya que en los mensajes se produce una gran interacción con los demás miembros del equipo, y siempre asociada a una unidad de trabajo. Desde el dispositivo móvil se podrá atender a cuantas peticiones nos realicen. Todo ello debe desencadenar en una mayor flexibilidad y agilidad, ya que atender este tipo de mensajes muchas veces reducen tiempos muertos para unidades de trabajo que dependen su continuidad de la respuesta a un mensaje, y si ese agente se encuentra fuera de la oficina, el dispositivo móvil es su herramienta perfecta para estar comunicado y sincronizado con el resto de miembros del equipo. (Figura 26) De igual modo que

en la aplicación móvil veremos un resumen del número de mensajes pendientes de ser respondidos, mensajes respondidos, mensajes pendientes de responder y mensajes que se están respondiendo. Para explicar la mecánica, explicaremos como hemos implementado la funcionalidad de mensajes respondiendo, ya que el resto de tipos de mensajes son subconjunto de esta funcionalidad. En la aplicación de escritorio (figura 26), podemos ver simultáneamente el resumen de comunicación y el detalle del tipo de mensaje seleccionado, en cambio para la aplicación móvil, lo repartimos en dos pantallas, por un lado el resumen, y por otro el detalle. En la siguiente figura 27, es donde veremos más diferencias entre la aplicación de escritorio y la aplicación móvil, esto está motivado porque desde la aplicación de escritorio se accede a una pantalla que ofrece acceso a múltiples funcionalidades. Los mensajes están dentro del módulo de las WUs, e incluyen otras funcionalidades como definición y seguimiento, relaciones, el workflow, registro de tiempos, las pruebas de aceptación, etc... En cambio la aplicación móvil, descarta estas funcionalidades porque son tareas que se realizan en el lugar de trabajo, presentando una pantalla más simplificada para los mensajes, pero con la funcionalidad completa. Podemos dividir los tipos de mensaje en dos grupos, los que envía el agente, “Esperando respuesta” y “Leer respuesta”, donde podrá acceder en modo lectura al detalle del mensaje, y los mensajes que debe responder el agente, “Debo responder” y “Respondiendo”, donde hay un proceso de estado de los mensajes “*ACTIVO*”,”*PAUSADO*”,”*EN PROGRESO*” y “*FINALIZADO*” que esta implementado en su totalidad.

Aplicación escritorio



Aplicación móvil

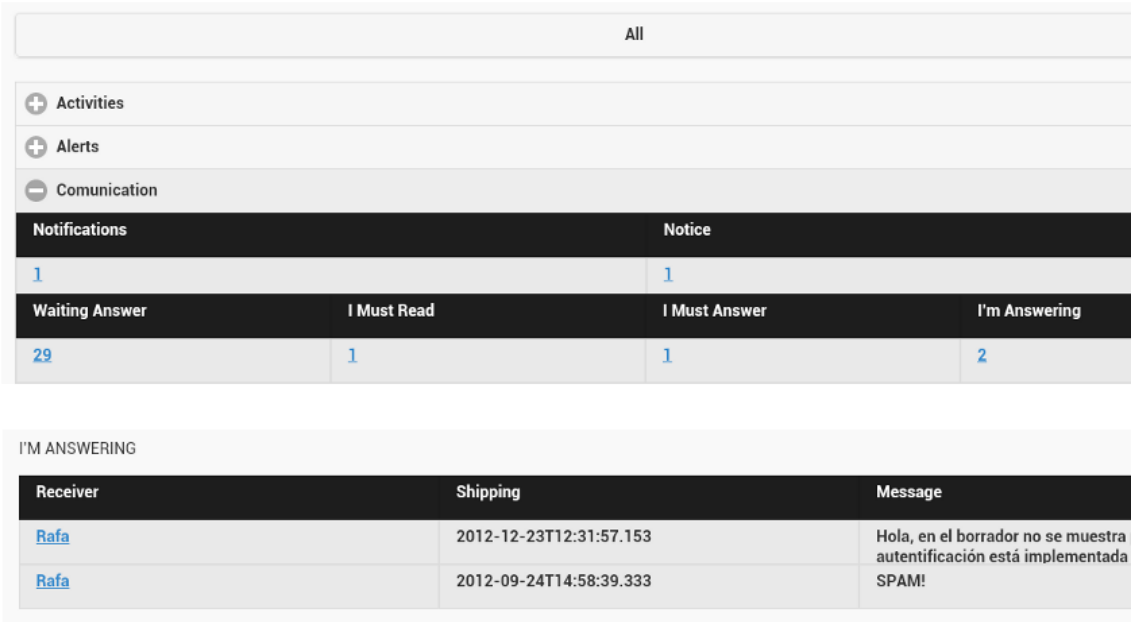
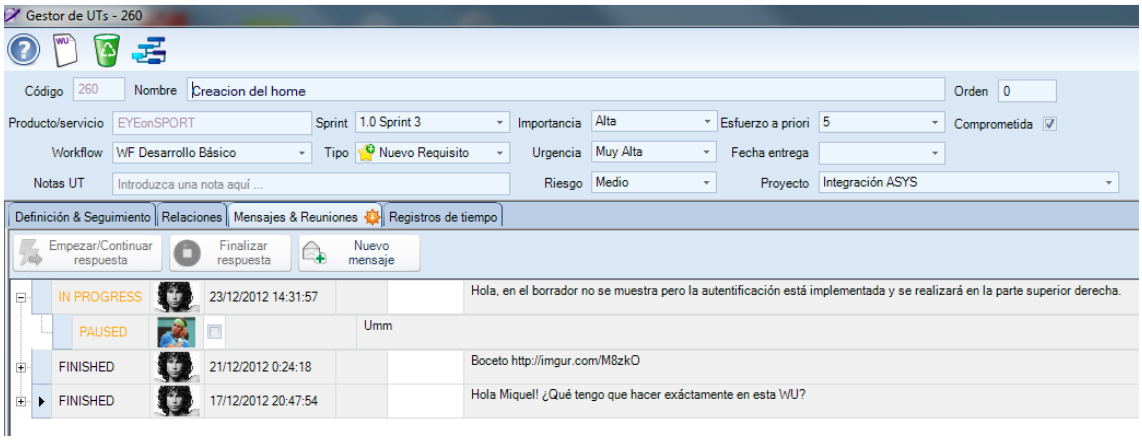


Figura 26. Mensajes

De este modo desde la aplicación móvil accedemos a una lista de mensajes, a partir del tipo de mensaje seleccionado (“Esperando respuesta”, “Leer respuesta”, “Debo

responder" y "Respondiendo"), de esta lista selecciona el agente el mensaje del que quiere conocer más información y dependiendo del tipo de mensaje, podrá responder y finalizar el proceso o será de consulta para saber la respuesta al mensaje enviado por el agente. Otra diferencia importante, es que los mensajes en la aplicación de escritorio están agrupados según la WU que la origino, y en la aplicación móvil no existe ese vínculo de visualización, es una funcionalidad a estudiar para reconocer el impacto de esta implementación como una posible mejora.

Aplicación escritorio



Aplicación móvil

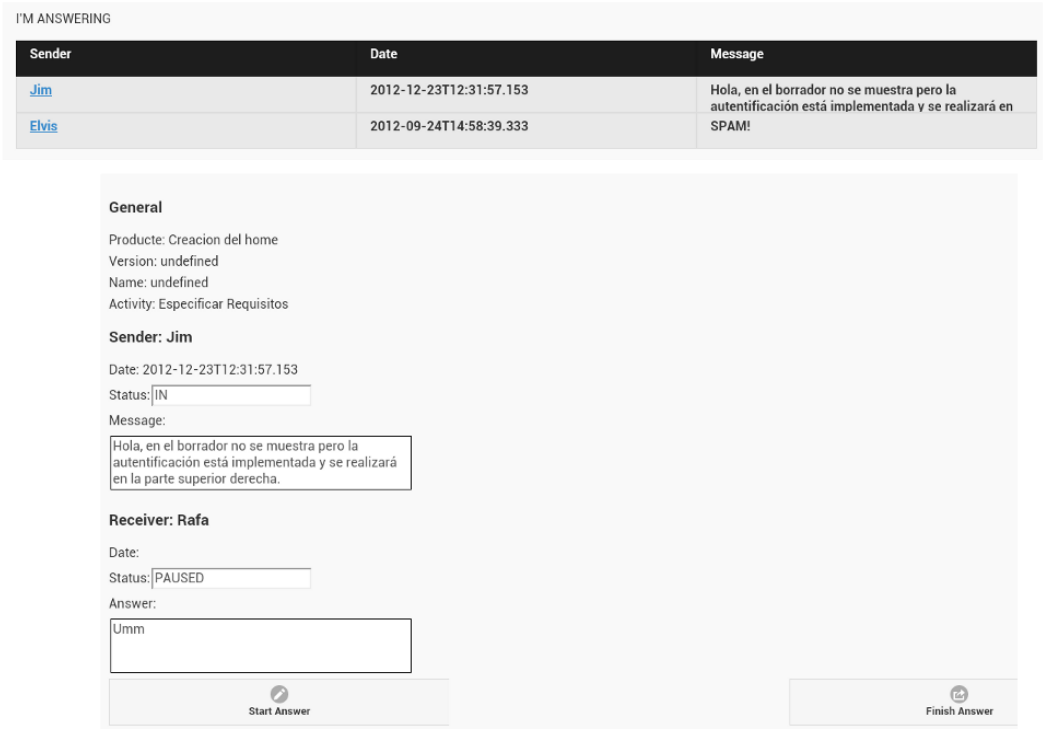
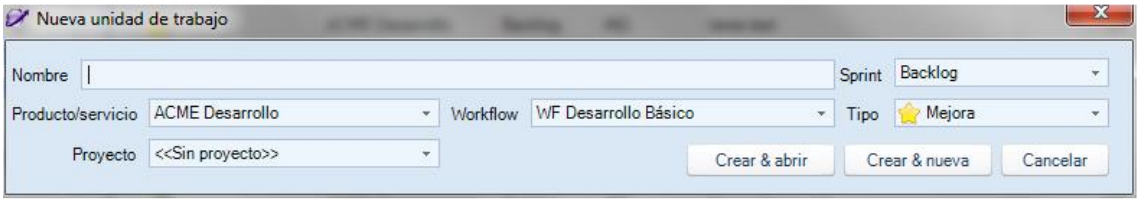


Figura 27. Gestión mensajes

6. ***Unidad de trabajo***, permitir crear nuevas unidades de trabajo, con la información básica. Nos permitirá crear unidades de trabajo desde el dispositivo móvil, para más tarde desde el puesto de trabajo completar la información restante, además de dar conocimiento al resto de miembros del equipo de estas nuevas unidades de trabajo (percepción). Esta funcionalidad no ha sido implementada por lo que la figura 28 muestra un boceto de la creación de una nueva unidad de trabajo.

Aplicación escritorio



Boceto aplicación móvil

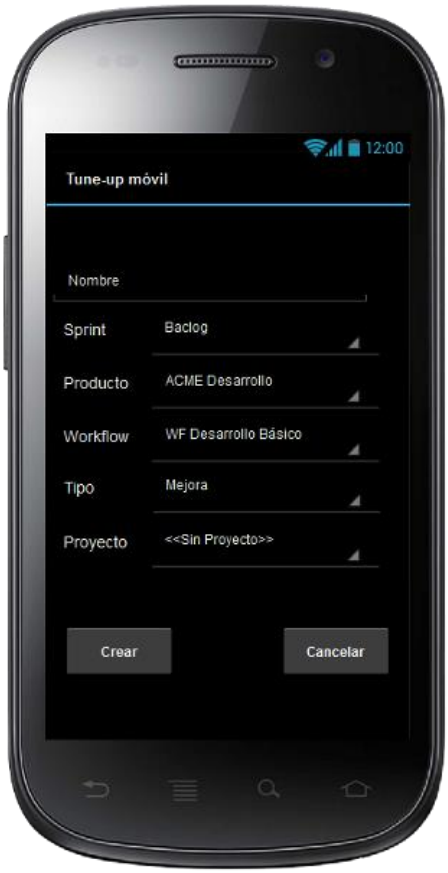


Figura 28. Nueva WU

Funcionalidades supervisión

Estas funcionalidades estan encaminadas a que el perfil supervisor pueda sacar conclusiones del estado del producto o de la versión , por lo que no sera necesario tener

una visión tan detallada de las actividades y WUs, pero si suficiente para poder acometer correcciones si asi fuera necesario. Otra diferencia de las funciones genericas, radica en que el perfil supervisor tiene que conocer el resultado hasta la fecha de la versión, sin preocuparse, en principio de los agentes implicados, siendo mas importante la evolución de las WUs. La posibilidad de gestionarlo desde el dispositivo móvil, repercute en tener la información actualizada en cualquier momento y lugar, y favorece la planificación y el seguimiento del producto [18]. No se han implementado ninguna de las funciones de supervisor, por lo que mostraremos los boceto correspondientes.

1. ***Flujo acumulado***, con esta grafica de la figura 29 podemos ver como se están moviendo las WUs por el workflow cada día, muy útil para detectar rápidamente posibles cuellos de botellas, y anomalías. Una secuencia lógica es que la WUs van pasando de una actividad a otra hasta finalizar la WU, un pico para una actividad no final, WUs que no cambian de actividad serán señales de aviso, que con alto nivel de ***Situational Awareness***, ayudaran a la corrección de las posibles anomalías.

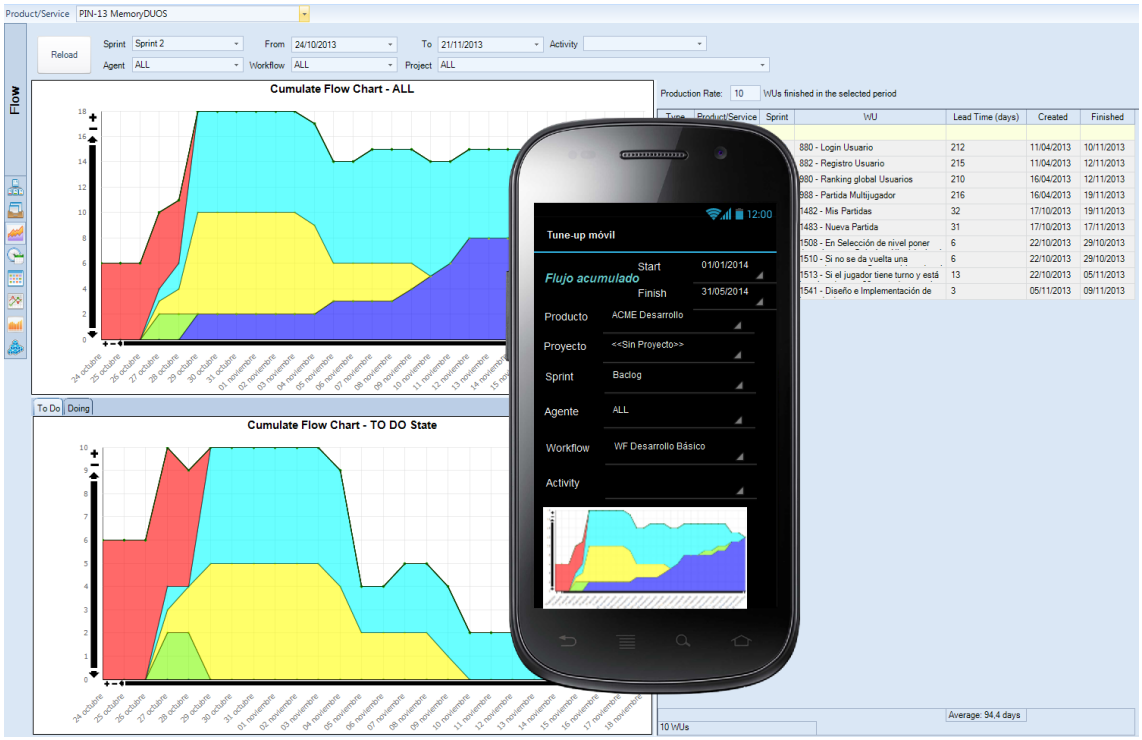


Figura 29. Boceto flujo acumulado

2. ***Dashboard del sprint***, este ítem corresponde al explicado con más detalle en el Capítulo 2. La información (figura 30) que podemos ver sobre el terminal móvil

será la gráfica de Burn Down, para conocer la evolución diaria del proyecto, el esfuerzo por tipo de actividad, nos ayudara a saber que WU han tenido mayor retroalimentación, la relación de WU finalizadas y no finalizadas, y por último el estado de las WUs a partir de las pruebas de aceptación.

Figura 30. Boceto Dashboard

3. ***Tendencias del sprint***, figura 31, nos permite contextualizar el producto en los diferentes sprints, mostrando la información asociada a la capacidad del equipo, cantidad y tipo de WU realizadas y duración de sprints. Estas medidas son importantes para futuros sprints, ya que permiten organizar las cargas de trabajo, siempre que estemos tratando sprints similares (mismo equipo, misma tecnología).

Figura 31. Boceto tendencias sprint

Para cualquiera de estas funcionalidades de supervisión, sería interensante que ademas de presentar la gráfica, podamos acceder a información mas detallada, nos situara con mayor precisión en el contexto y permitira tomar decisiones en el mismo momento de conocer la posible anomalia. En este sentido se plantea de manera generica, acceder desde las graficas a las WUs implicadas, y a su detalle si asi fuera necesario como sucede en la aplicación de escritorio.

Capítulo 7. PhoneGap

En este capítulo explicaremos los pasos para poder iniciarse en PhoneGap, desde cero hasta tener una pequeña aplicación que hace uso de las características nativas de los terminales. Finalmente explicaremos como desplegar la aplicación para poder generar el código para múltiples plataformas y los desafíos que nos hemos encontrado y las soluciones adoptadas.

7.1. Instalación de PhoneGap

Antes de iniciar este tutorial, hay que mencionar que el objetivo de este no es explicar el funcionamiento de eclipse ni de Android, aunque sería recomendable tener unos breves conocimientos en el entorno eclipse (http://www.eclipse.org/), para poder trabajar con mayor agilidad, y conocimientos básicos de Android y de su estructura, para ver en este caso concreto las diferencias al utilizar PhoneGap.

Desde la página de PhoneGap *www.PhoneGap*.com tenemos un menú llamado ***Developer → Get Started***, donde nos indica las distintas plataformas sobre las que podemos desarrollar. (Abril 2014), Amazon Fire Os, Android, Blackberry 10, iOS, Windows Phone, Windows 8 y Tizen.

A grandes rasgos, la metodología es la misma para todas las plataformas, con las peculiaridades de cada una, especialmente en el IDE a utilizar, XCode para iOS, Visual Studio para WindowsPhone, Eclipse para Android y BlackBerry.

El tutorial lo realizaremos sobre la plataforma de Android, para ello necesitaremos descargarnos en nuestro equipo el SDK, accederemos a la página http://developer.android.com/intl/es/sdk/index.html de Android SDK – Android Developers. En la documentación de Android SDK, veremos que para poder trabajar necesitaremos instalar:

1. Eclipse + ADT Plugin
2. Android SDK Tools
3. Android Platform-tools

Todo este software viene junto, tan solo pulsaremos sobre el botón “Download the SDK”, después de leer los términos y condiciones, aceptaremos y seleccionaremos la plataforma, 32 o 64 bits.

Por otro lado necesitaremos instalar Java, accederemos a la página www.oracle.com. Indicaremos dos formas de llegar al JRE, debido a que muchas están cambian de diseño e incluso la documentación sufre cambios en su alojamiento.

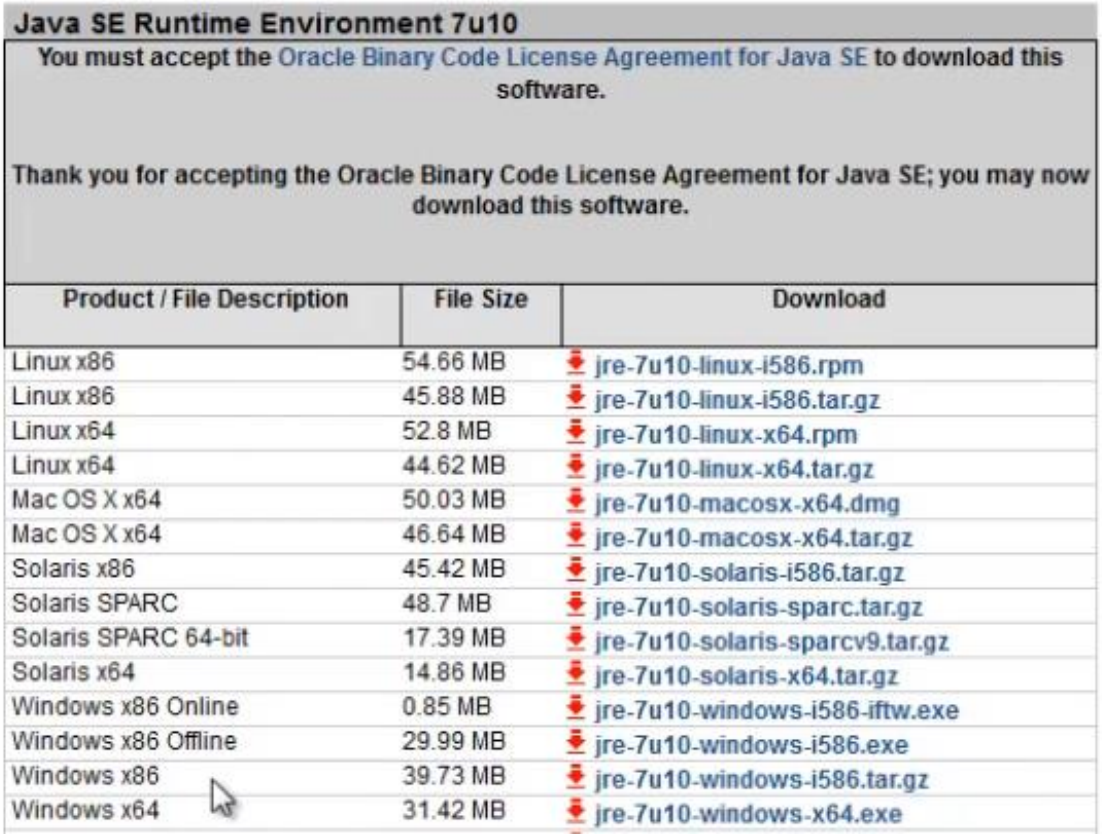
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Seleccionaremos ***Downloads → Java for Developers***

En ambos casos deberíamos poder llegar a la imagen de la figura 32

Figura 32. Descarga JRE

Pincharemos sobre JRE-Download, y una vez aceptamos los términos, elegimos el producto sobre el que vamos a trabajar, en nuestro caso será “Windows x64”, por tratarse de un sistema operativo de 64 bits con el que estamos trabajando. Ver figura 33.



Java SE Runtime Environment 7u10

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	54.66 MB	jre-7u10-linux-i586.rpm
Linux x86	45.88 MB	jre-7u10-linux-i586.tar.gz
Linux x64	52.8 MB	jre-7u10-linux-x64.rpm
Linux x64	44.62 MB	jre-7u10-linux-x64.tar.gz
Mac OS X x64	50.03 MB	jre-7u10-macosx-x64.dmg
Mac OS X x64	46.64 MB	jre-7u10-macosx-x64.tar.gz
Solaris x86	45.42 MB	jre-7u10-solaris-i586.tar.gz
Solaris SPARC	48.7 MB	jre-7u10-solaris-sparc.tar.gz
Solaris SPARC 64-bit	17.39 MB	jre-7u10-solaris-sparcv9.tar.gz
Solaris x64	14.86 MB	jre-7u10-solaris-x64.tar.gz
Windows x86 Online	0.85 MB	jre-7u10-windows-i586-iftw.exe
Windows x86 Offline	29.99 MB	jre-7u10-windows-i586.exe
Windows x86	39.73 MB	jre-7u10-windows-i586.tar.gz
Windows x64	31.42 MB	jre-7u10-windows-x64.exe

Figura 33. Listado JRE según plataformas

Una vez instalado el JRE, descomprimimos el fichero descargado del SDK de Android y lo copiamos en la carpeta C:\, entramos dentro de la carpeta eclipse y ejecutamos el fichero "**eclipse.exe**"

Ahora lo que necesitaremos será instalar unos elementos para trabajar con la SDK, y PhoneGap. La idea es poder simular el entorno de Android, para ello desde eclipse seleccionamos ***Help → Install New Software***, pulsaremos sobre el botón añadir y en la nueva ventana añadiremos la siguiente información para obtener la herramienta de desarrollo de Android. Ver figura 34.

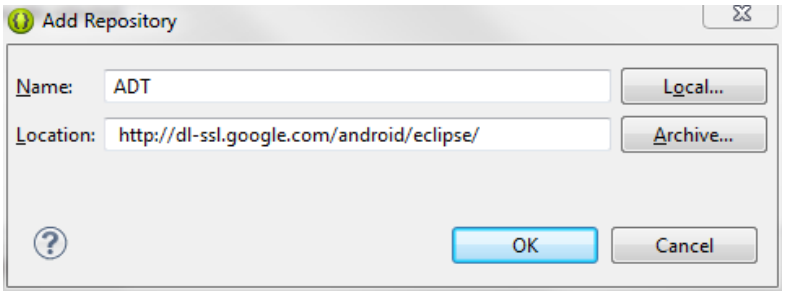


Figura 34. Repositorio Android

Aceptamos, y seleccionamos todo con las opciones por defecto. Durante el proceso aparece una ventana de advertencia sobre la autenticidad del software, aceptaremos y el proceso continuara hasta que nos pida reiniciar para que los cambios tengan efecto. (Comentar también que dependiendo del nivel de protección del firewall, nos aparece una ventana solicitando permiso para algunas de la características de Java ™, para poder acabar con éxito deberemos dar acceso).

Una vez se haya reiniciado eclipse tenemos que tener unos elementos propios de Android (*Android SDK manager* y *Android virtual device manager*), y con un aspecto similar a la figura 35.

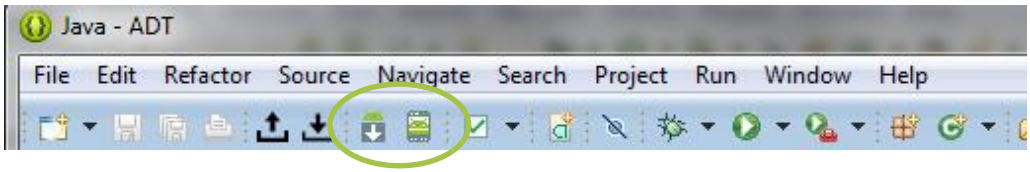


Figura 35. Menú de eclipse con las herramientas de Android

Para ejecutar el simulador, pulsaremos sobre *Android virtual device manager* y seguidamente sobre el botón *new*, y nos aparece un ventana donde podremos indicar el terminal sobre el que queremos simular. Figura 36.

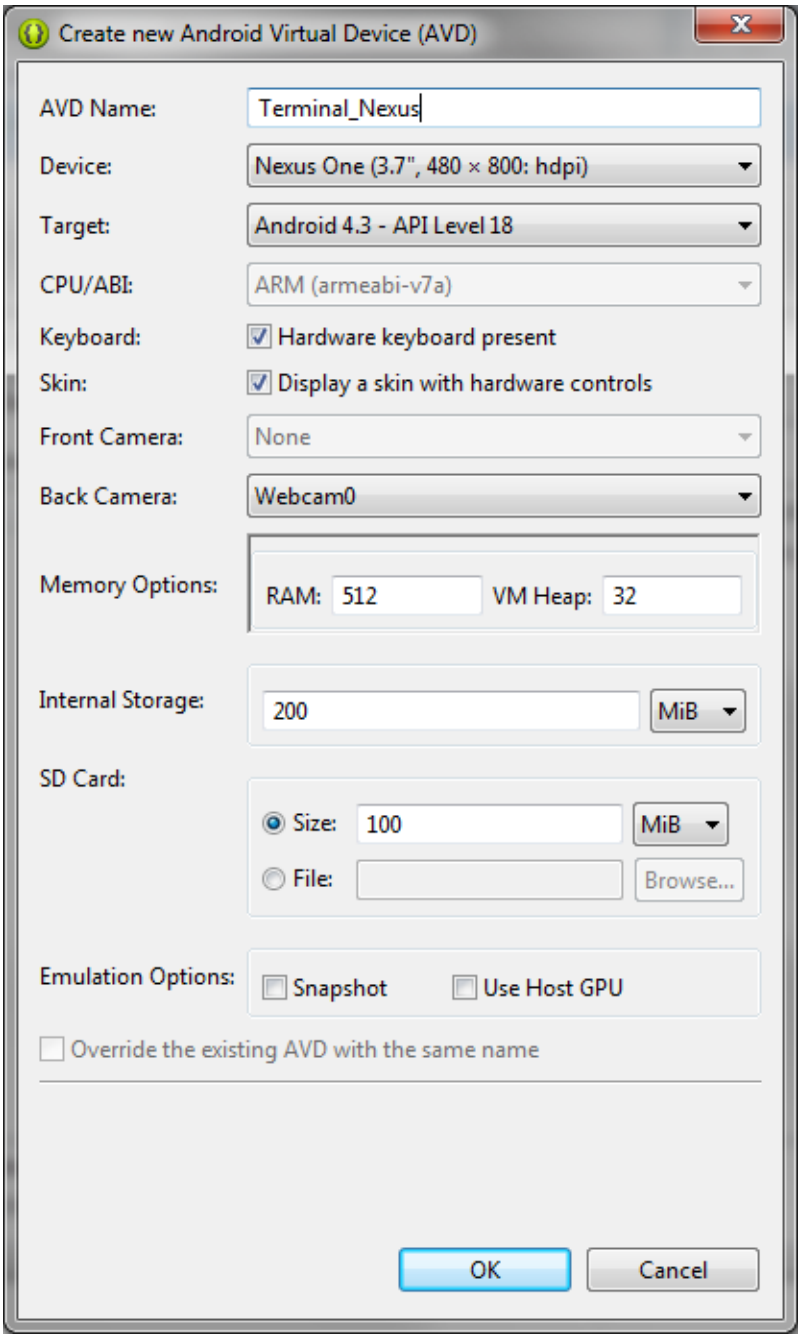


Figura 36. Configurar emulador Android

Indicaremos las características más importantes a tener en cuenta a la hora de parametrizar el simulador.

- *AVD Name*, Introduciremos un nombre para identificarlo en.
- *Device*, indicamos el dispositivo que queremos que se emule.
- *Target*, la versión de Android, dependiendo de la seleccionado, habrán características que no se podrán realizar.

- *Front camera*, esta opción no estará siempre habilitada, dependerá del modelo a emular.

- *Back camera*, podemos simular la cámara o bien seleccionar la webcam si nuestro equipo dispone de este dispositivo. Muy interesante si nuestra aplicación a desarrollar hace uso de la cámara.

- *SD card*, podemos introducir un tamaño, para que simule que tiene instalada una tarjeta SD.

Una vez introducidos todos los valores aceptaremos y nos aparecerá en nuestra lista de dispositivos, desde ese momento podemos lanzar el emulador pulsando sobre ***Start*** → ***Launch***. Este proceso puede ser lento, y dependerá de la máquina con la que estemos trabajando, del emulador que tengamos seleccionado, cuanto más nuevo sea el modelo a emular más recursos necesitara, ya que la emulación intenta ser tan fiel, que para estos las necesidades de recursos son mayores.

En este punto aún no tenemos instalado PhoneGap. Como la elección para la interfaz con el usuario fue jQuery mobile, propondremos una forma rápida y cómoda de instalar PhoneGap con ejemplos de las APIs. Accederemos al link http://www.mobiledevelopersolutions.com/home/start , veremos que nos detalla los pasos realizamos para instalar PhoneGap, y en el paso 5, copiaremos la dirección http://svn.codespot.com/a/eclipselabs.org/mobile-web-development-with-PhoneGap/tags/r1.2/download y realizaremos el mismo proceso que hemos hecho antes para instalar las características de SDK (***Help → Install new software***) Ver figura 37.

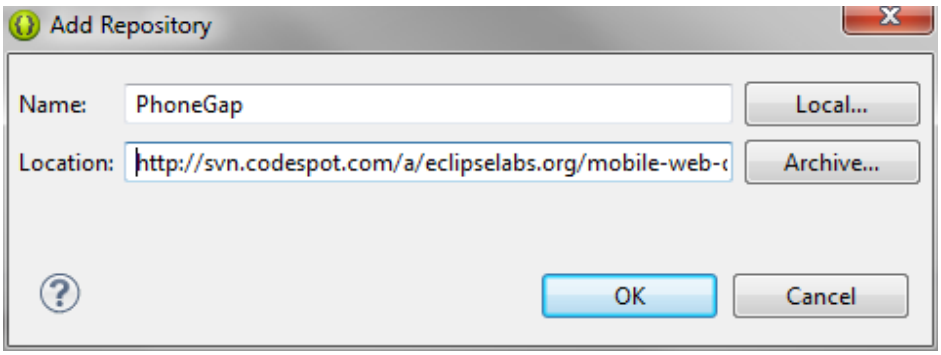


Figura 37. Instalación de ejemplos APIs con PhoneGap

Aparecerá el MDS-AppLaud, y de igual que en el anterior caso instalara los servicios necesarios, aceptaremos los términos de licencia y nos pedirá reiniciar el entorno de

eclipse. Al reiniciarse eclipse podremos observar que ha aparecido un nuevo icono. Ver figura 38.

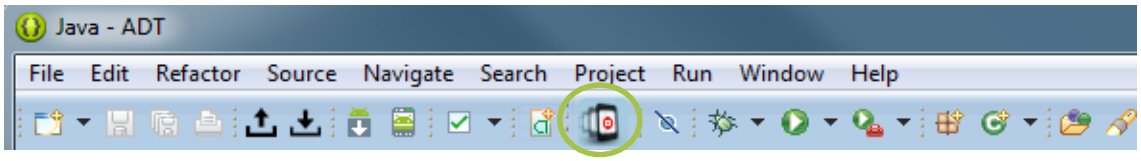


Figura 38. Instalación MDS-AppLaud

Con este paso finaliza la instalación de PhoneGap, ya estamos listos para empezar a trabajar.

Como ya hemos comentado anteriormente, podemos probar uno de los ejemplo que nos proporciona MDS-AppLaud, para conocer mejor como se trabaja con PhoneGap, además de tener siempre a mano la documentación de PhoneGap (***Developer*** → ***Docs*** desde http://PhoneGap.com). Seleccionamos el botón de PhoneGap y nos solicitara una serie de información.

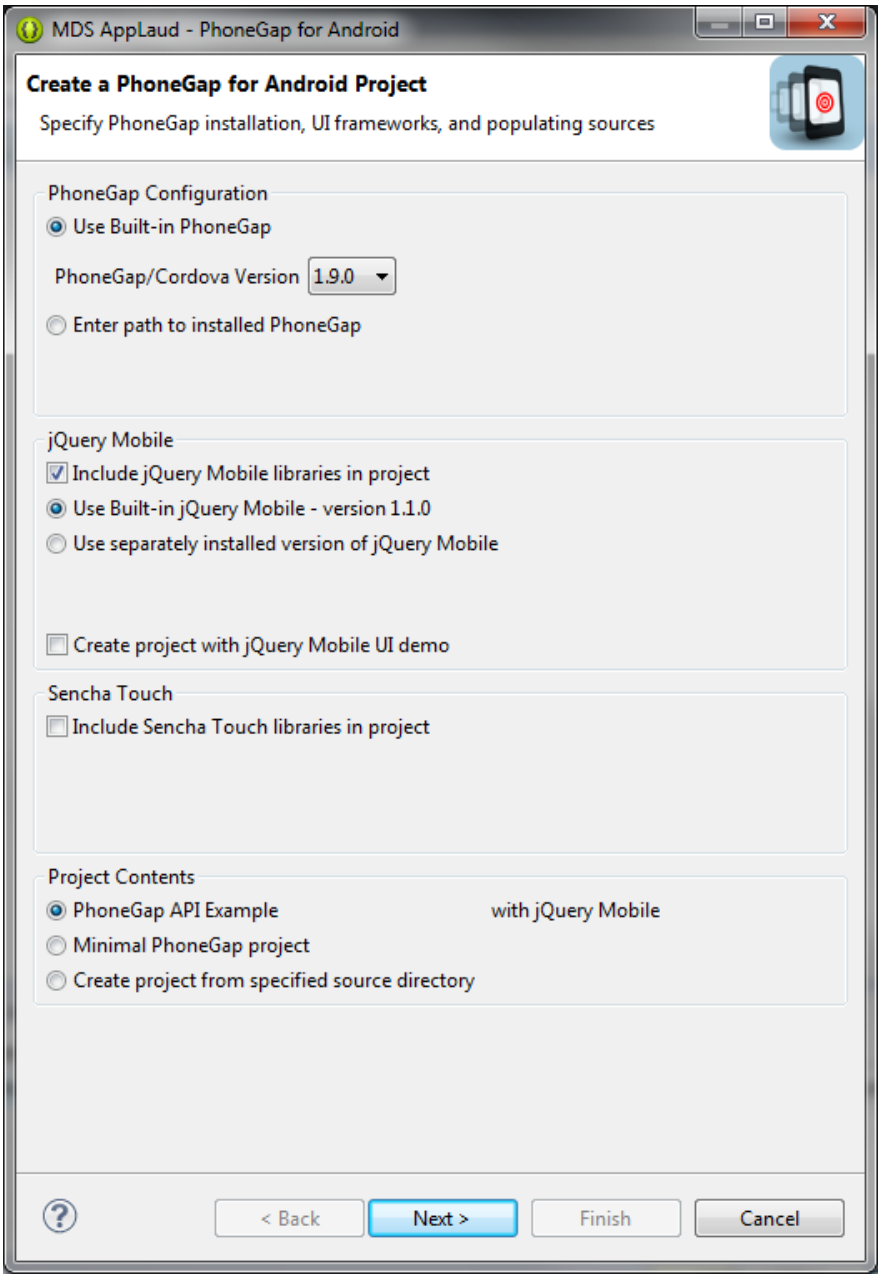


Figura 39. Crear nuevo proyecto PhoneGap

Para crear un nuevo proyecto PhoneGap, realizaremos los siguientes pasos que detallamos a continuación y que aparecen en la figura 39.

- Configuración PhoneGap, seleccionamos *Use Built-in PhoneGap*, y la versión de PhoneGap
- JQuery Mobile, incluimos JQuery mobile, para dar una interfaz de usuario más atractiva.
- Project Contents, dejamos la selección de *PhoneGap API example*

pulsaremos sobre next, y nos pedirá un nombre de proyecto, *next*, seleccionamos sobre que target trabajaremos, en nuestro caso es Android 4.3, pero puede ser superior ya que se están realizando continuos actualizaciones, pulsamos *next* e introduciremos el *package name*, pulsamos sobre *Finish* y tendremos un ejemplo con MDS-AppLaud.

Para introducir este ejemplo, y poder verificar el correcto funcionamiento de los múltiples paquetes, SDK Android, plugins ADT, librerías de PhoneGap, y ejemplos, explicaremos brevemente que nos encontraremos y como se distribuye la información.

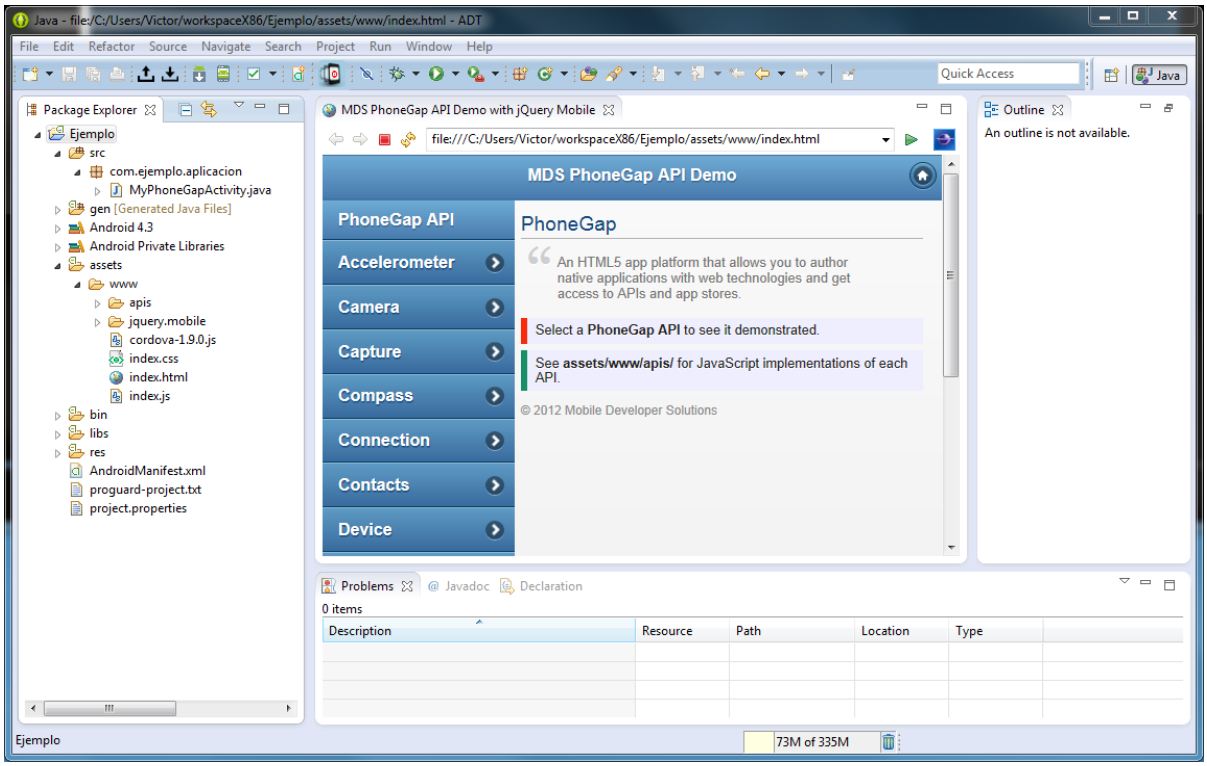


Figura 40. Ejemplo de aplicación con PhoneGap en IDE eclipse

El cuerpo del código se encuentra en la carpeta ***assets***, en esta carpeta podemos reconocer la carpeta ***www***, dentro de ella podemos ver una estructura que se asemeja al desarrollo de un entrono web, fácil de entender para un desarrollador de entornos webs. Tenemos el fichero ***index.html***, el fichero ***cordova-1.9.0.js*** es propiamente dicho PhoneGap, la carpeta ***apis***, donde estará el código para lanzar las funciones nativas del dispositivo móvil, véase cámara, contactos, acelerómetro, conexiones, geolocalización, notificaciones y almacenaje entre otras, este es el aporte de PhoneGap, la carpeta ***jquery.mobile***, donde están las librerías para mejorar el aspecto visual y de transición de pantallas (como ya hemos comentado jquery es opcional, pudiendo trabajar con otros frameworks , véase sencha). Ver figura 40.

Ahora bien, como este tutorial esta realizado bajo Android, para que este ejemplo pueda ejecutarse, debe lanzarse el fichero ***index.html***, esto se realiza desde el fichero *MyPhoneGapActivity.java* que se encuentra dentro del directorio *src*. Ver figura 41.

MDS PhoneGap API Demo with jQuery Mobile — MyPhoneGapActivity.java

```
package com.ejemplo.aplicacion;

import org.apache.cordova.DroidGap;

public class MyPhoneGapActivity extends DroidGap {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
    }
}
```

Figura 41. Llanada a index.html

Si ejecutamos la aplicación sobre el emulador podremos ver este comportamiento, y del mismo modo podemos lanzarlo sobre un terminal (móvil, tablet) de Android. Las pruebas se han realizado sobre una Archos 7 home Tablet con versión 1.5 y un móvil Sony Ericsson XPERIA con versión 2.1 de Android con resultado satisfactorio.

7.2. Desarrollo en PhoneGap

Para el desarrollo de la aplicación hay que decidir sobre que versión de framework vamos a desarrollar. En el caso de la versión de PhoneGap la decisión no será tan crítica, ya que por un lado cuando compilemos en PhoneGap Build podemos seleccionar sobre la versión definitiva y por otro que la aplicación TUNE-UP no va hacer un uso intensivo de las características de los terminales, se limita al uso de internet y notificaciones., en cambio para el interfaz de usuario, que en nuestro caso hemos decidimos realizarlo con *jQuery mobile v.1.4.2* sí que podemos encontrar algunas características que no están soportadas.

La aplicación móvil está basada en la aplicación de escritorio TUNE-UP Software Process, y atendiendo al estudio realizado del contexto situacional del Capítulo 3 y 6, reflejaremos cual ha sido el resultado final. En la figura 42 podemos ver la definición de lo que serán las pantallas y representada su navegación.

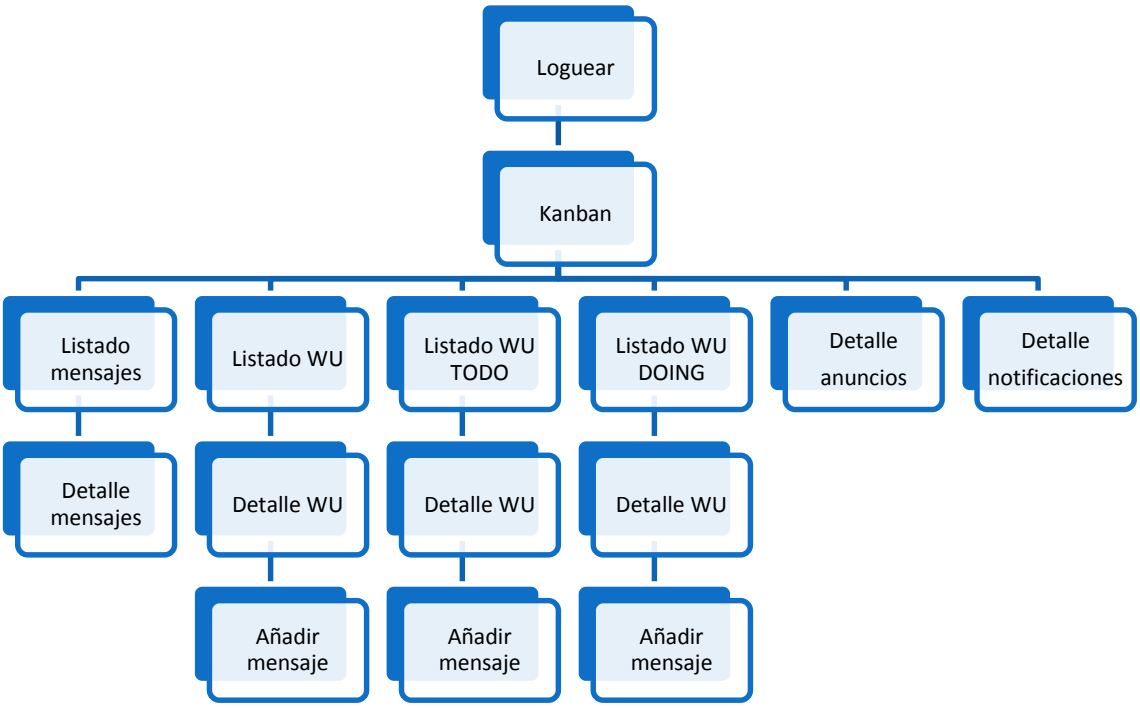


Figura 42. Diseño estructura pantallas

A continuación detallaremos las pantallas y las funcionalidades de cada una de ellas.

Todas las pantallas se dividen en cabecera, cuerpo y pie. La cabecera y el pie son comunes a todas las pantallas exceptuando la pantalla *Loguear*, en la cabecera aparece el nombre de la aplicación, en el pie los botones info (para conocer versión, desarrollador y demás datos genéricos de la aplicación), volver (permite volver a la pantalla anterior) y una descripción del usuario y el sitio conectado. El cuerpo será la parte más dinámica, en ella iremos mostrando la navegación de las pantallas.

- *Loguear*, en esta pantalla el usuario introducirá sus datos de validación a TUNE-UP, y dependiendo del tipo de usuario se le solicitara que seleccione el sitio sobre el que conectar si tiene varias opciones o entrar directamente en la aplicación si solo tiene registrado un sitio. Los servicios de los que se hacen uso son:
    - *getSitiosPorLogin*
    - *getAgentesPorLoginPassword*

- *Kanban*, es el núcleo de la aplicación, en ella se recoge la información principal para que el agente pueda tomar decisiones que faciliten la continuidad del desarrollo, gracias a la operatividad móvil. La información que se muestra son un resumen del número de WU en las actividades, el estado de los mensajes, alarmas, notificaciones y anuncios. Los servicios utilizados son:
    - *getMensajesEnviadosPorAgentesNoLeidos*
    - *getActividadesPorAgentes*
    - *getProductosActivosPorAgentesParticipantes*
    - *getNotificacionesTodas*
    - *getAnunciosTodos*
    - *getAlertasTodas*
- *Listado mensajes*, para los cuatro tipos de mensajes (esperando respuesta, mensaje leído, respondiendo, respondido) la lógicas de negocios será la misma, por lo que agruparemos su descripción. Se mostrara un grid con la información del mensaje, agente enviado, y fecha de envió. En esta pantalla no se hace uso de ningún servicio, ya que la información ya ha sido recuperada en la pantalla *Kanban* con el servicio *getMensajesEnviadosPorAgentesNoLeidos*.
- *Detalle mensajes*, presenta una pantalla con una descripción más detallada del mensaje, y su estado. La información es separada en tres bloques, información general del mensaje, información del agente que envía el mensaje e información del agente que recibe el mensaje. El agente puede realizar las siguientes acciones, iniciar, pausar o finalizar la respuesta, excepto para los mensajes respuesta. Los servicios de los que hace uso son:
    - *getMensajesRecibidos*
    - *updateMensajesRecibidos*
    - *finalizarMensajeRecibido*
    - *empezarMensajeRecibido*

- *pausarMensajerecibido*

- *Listado WU, Listado WU TODO, Listado WU DOING,* hemos agrupado las tres pantallas de *Listado WU* porque tienen la misma funcionalidad, la diferencia entre ellas es que la primera muestra todas las WUs, la segunda solo muestra las WUs que está en TODO, y la tercera muestra las WUs que están en DOING. Mostramos un grid con el número y nombre de la WU y su estado. Los servicios de los que hace uso son:
  - *getMisSeguimientosPendientes*
  - *getMisSeguimientosProgreso*
- *Detalle WU*, mostramos mayor detalle de la WU seleccionada, dando posibilidad al agente a enviar un mensaje asociado a esta WU. Los servicios asociados a la pantalla son:
  - *getUT*
  - *getWorkflow*
- *Añadir mensaje*, permitimos al usuario enviar un mensaje con los parámetros básicos de a que agente enviaremos la información y la actividad origen y destino. Los servicios utilizados son:
  - *getActividadesOrdenadasPosicion*
  - *getAgentesPorProducto*
- *Detalle anuncios*, podemos ver los anuncios publicados con su fecha de publicación, el agente que realizo el anuncio, y por supuesto el anuncio. Para validar ese anuncio, y que se registre como leído, seleccionaremos el ítem leído. Los servicios utilizados son:
  - *deleteAnuncio*
- *Detalle notificaciones*, podemos ver las notificaciones en un grid con la información básica de fecha, WU y descripción de la notificación, para validar la notificación, seleccionaremos sobre el registro correspondiente el ítem leído. Los servicios utilizados son:

- *deleteNotificacion*

Bajando el nivel de abstracción vamos a ver cómo hemos definido la arquitectura de la aplicación móvil TUNE-UP.

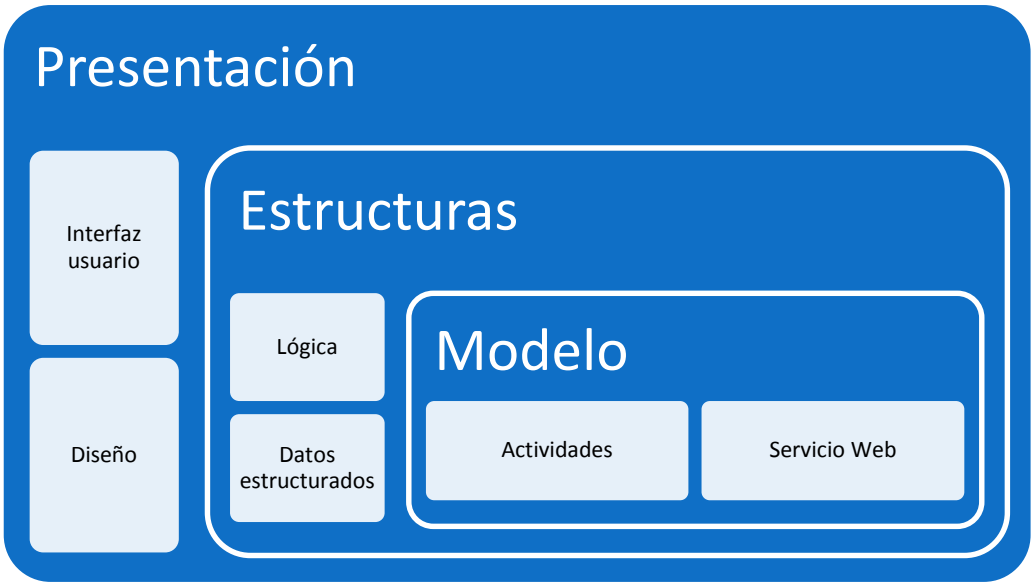


Figura 43. Arquitectura TUNE-UP móvil

Podemos ver en la figura 43 una vista de esta arquitectura, explicamos a continuación con más detalla las diferentes partes y los ficheros principales asociados a cada división.

- ***Presentación***, en esta capa queremos atender únicamente a la presentación de la información o datos, para ello tenemos por un lado la interfaz de usuario, que se plasma en el fichero index.html. Siguiendo el criterio que se aconseja en PhoneGap, para pequeñas aplicaciones se implementa en un único fichero todas las posibles pantallas, podemos mejorar el rendimiento y las pantallas siguen vivas en todo momento. Para darle un aspecto más atractivo utilizamos las librerías de jquery, así que según la declaración de clases en una etiqueta podemos darle aspecto de botón, lista, grid, etc.

- ***Estructuras***, esta capa va a recibir los eventos de la capa de presentación y recibir nuevos datos del modelo para presentarlos a la capa de presentación. La hemos dividido en dos apartados, la etiquetada como lógica, corresponde al fichero físico tuneup.js, y trata toda la lógica de los procesos, de forma que ante un evento en la capa de presentación, ejecuta el proceso correspondiente que contiene una serie de métodos que están en la capa de modelo, cuando esta capa finaliza su

proceso trata la información que viene en estructuras de datos que coinciden con la salida de la llamada a los servicio web. Estos datos son procesados y presentados en la capa de presentación. En este fichero también están los métodos auxiliares para filtrado y operaciones de apoyo, como pueden ser formatear fecha. El apartado de estructura de datos, son una serie de ficheros que contienen la estructura de datos que es devuelta por los servicios web, con los métodos get y set para acceso a los atributos de la estructura.

- ***Modelo***, esta capa contiene las actividades, que se corresponden con el fichero actividades.js, contiene para cada llamada a una funcionalidad del servicio web, el método que monta la estructura para realizar la llamada SOAP, y un método de recuperación del resultado. En el apartado servicio web [19], hacemos uso de una librería para clientes SOAP en javascript, el resultado es capturado por el método de recuperación de resultados, e invoca la estructura de datos con la información recibida y llama a un método de la capa intermedia para que trate la información.

En la figura 44, vemos sobre el propio eclipse la estructura de carpetas que cuelgan sobre la carpeta assets, que es donde se encuentra toda la información que necesita PhoneGap para generar la aplicación.

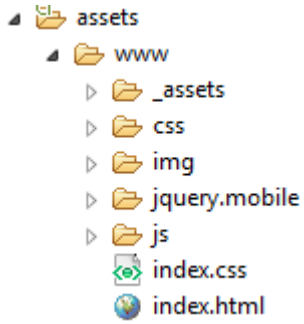


Figura 44. Estructura carpetas en eclipse

Tenemos por un lado el fichero index.html, que es el fichero que será invocado desde el entorno de Android para poder lanzar la aplicación, contiene el listado de todas las librerías necesarias, y el diseño de las pantallas. La carpeta js, contiene todos los ficheros javascrpits que hemos implementado. La carpeta jQuery-mobile contiene los fichero de jQuery para la interfaz de usuario. La carpeta img, tenemos las imágenes que añadimos a

la aplicación. La carpeta css, tiene los ficheros que proporciona el tema para la interfaz de usuario.

7.3. Despliegue

PhoneGap, es un framework, que se basa en javascript, CSS y HTML5. Desde PhoneGap vamos a poder compilar el código fuente, y decidir sobre qué plataforma queremos desplegarlo, Android, iOS, BlackBerry y Windows Phone.

La compilación se puede realizar sobre el propio equipo informático, una tarea más costosa, sobre todo si queremos desplegar sobre más de una plataforma, deberíamos tener un MAC, los IDEs eclipse y XCode, y sobre el proyecto tener las librerías para cada uno de los terminales. La opción más sencilla, es utilizar el servicio que hay disponible en la nube PhoneGap Build (figura 45), servicio disponible que nos permite construir la aplicación con el SDK más actualizado para las plataformas que indiquemos en el fichero *config.xml*.

Para poder compilar en la nube tendremos que validarnos con una cuenta de adobe o de Github. Por defecto, podemos tener una aplicación privada, el resto de aplicaciones serán públicas.

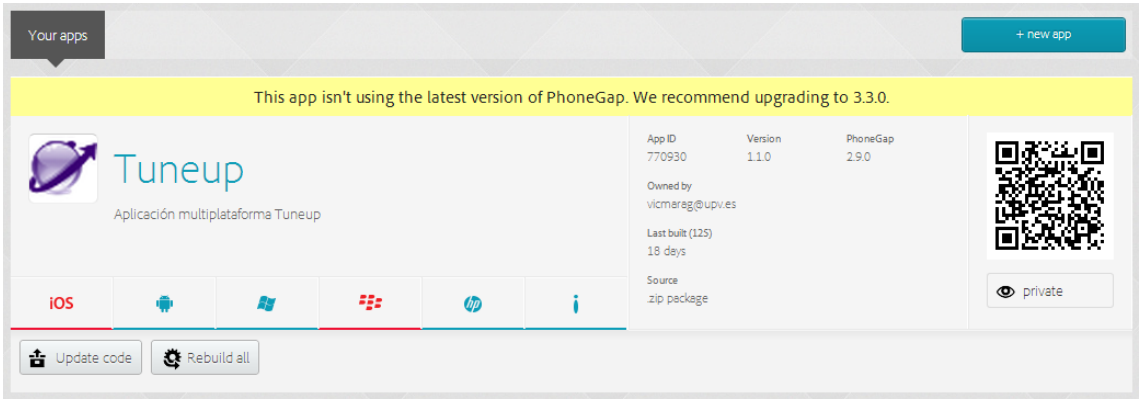


Figura 45. PhoneGap Build

Subiremos los ficheros empaquetados en formato zip, y procederemos a compilarlo. Generará un fichero específico para cada una de las plataformas, estos ficheros podemos descargarlos o bien podemos utilizar la imagen QR que se genera para que desde cualquier terminal se pueda descargar la aplicación.

Tanto para iOS como Windows Phone hay que registrarse y realizar el pago de una cuota para que la aplicación pueda ser descargada en algún terminal. Con el fin de probar la aplicación en las otras dos plataformas principales (iOS y WindowsPhone), realizamos una serie de pruebas. Para el caso de iOS, creamos una máquina virtual con el sistema operativo e instalamos la herramienta xCode, pero la falta de recursos hardware, un entorno nuevo y ha dificultado realizar las pruebas suficientes sobre el emulador. Con Windows Phone 8, existen varias posibilidades de poder lanzar sobre un terminal físico la aplicación. Una posibilidad es tener instalado el sistema operativo Windows 8, e instalar el SDK, y según la documentación podremos lanzar la aplicación sobre un número limitado de terminales, problema, la maquina sobre la que trabajaba es un Windows 7. Segunda posibilidad, utilizar el SDK para Windows 7, lanza un emulador, pero cuando nos logueamos la aplicación no avanza, como no se puede realizar trazas no podemos detectar el problema. La tercera posibilidad, fue probar sobre un servicio de Nokia (http://developer.nokia.com/resources/remote-device-access), que lanza nuestra aplicación sobre un terminal físico remoto, resultado insatisfactorio, a veces no cargaba el propio entorno del terminal, cuando subimos el paquete de la aplicación se queda en espera infinita. La cuarta, fue una herramienta de Phonegap, App Developper, que una vez instalada en tu terminal, y activado un servicio en el equipo de desarrollo, puede lanzar la aplicación que se encuentra en nuestro PC, el resultado fue que cargaba la aplicación, pero al loguearse lanzaba la llamada al servicio web y de ahí no pasaba. Esto nos hace pensar que tanto para Windows Phone como iOS tenemos un problema con el servicio web, y la única forma de solucionarlo es con sus respectivos IDEs.

En todos los casos existe la posibilidad de lanzar el emulador con el IDE adecuado a la plataforma a probar.

Volviendo a PhoneGap Build, el servicio compilara un fichero zip, la estructura de este fichero zip estará formada por los ficheros de la figura 44, eliminando carpetas y ficheros que finalmente no se utilicen (por ejemplo, dentro de la carpeta assets hay ejemplos que no tienen relación con el proyecto). De la carpeta js, eliminaremos la librería de PhoneGap, ya que posteriormente desde el fichero config.xml ya indicaremos que versión de librería queremos que se compile y será desde PhoneGap Build el que utilizara la librería adecuada. Del fichero index.html, dejaremos el nombre de la librería de PhoneGap sin ninguna versión si la hubiere y en la ruta raíz

<script src="PhoneGap.js"></script>

Añadiremos las imágenes que consideremos necesarias, pude ser que tengamos tantas como para cada uno de los tipos de terminales y tamaños. Finalmente tendremos que añadir un fichero llamado config.xml como mostramos en la figura 47, y que estará estructurado como detallamos a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<widget xmlns       = "http://www.w3.org/ns/widgets"
        xmlns:gap   = "http://phonegap.com/ns/1.0"
        id          = "com.upv.tuneup"
        version     = "1.1.0">

    <name>Tuneup</name>
    <description>Aplicación multiplataforma Tuneup</description>

    <preference name="orientation" value="default" />
    <preference name="target-device" value="universal" />
    <preference name="fullscreen" value="false" />
    <!--
    Solo IOS
    -->
    <preference name="webviewbounce" value="false" />
    <icon src="logo.png"/>

    <gap:splash src="logogrande.png" />
    <!-- imagen de presentacion-->
    <preference name="phonegap-version" value="2.9.0" />
    <feature name="http://api.phonegap.com/1.0/network"/>
    <feature name="http://api.phonegap.com/1.0/file"/>
</widget>
```

Figura 46. Fichero config.xml

Elementos admitidos por todas las plataformas soportadas:

- <***widget***>, elemento id, es el identificador de reserva de dominio de la aplicación y numero de versión completa
- <***name***>, nombre de la aplicación, aparece en la pantalla principal y en las tiendas
- <***description***>**,** <***autor***>, metadatos e información de contacto
- <***preference***>, establece opciones con los pares name y value. Dependiendo de la preference name, no estará disponible para todas las plataformas. En nuestro caso son globales fullscreen y orientation. La preference webviewbounce que solo está

para iOS, será obviada en el resto de plataformas. Una muy interesante es PhoneGap-version, que la utilizaremos cuando no queramos compilar en una versión distinta a la que por defecto ofrece PhoneGap Build.

- ***<feature>, <plugin>***, lo utilizamos para activar las características de APIs externas que podamos utilizar y las características de APIs nativas, como son cámara, internet, contactos, etc.

- ***<icon>***, identificamos la ruta del fichero que utilizaremos de icono de la aplicación. Además permite crear con la etiqueta *platform* un icono distinto por cada plataforma que indiquemos, y para cada plataforma un tamaño diferente por características dispositivo.

7.4. Desafíos

Durante el desarrollo nos hemos encontrado con diversos inconvenientes, que relataremos a continuación con las soluciones adoptadas, cuando ha sido posible.

Desarrollar con un lenguaje nuevo, javascript, siempre supone un reto para sacarle el máximo provecho. Lo más difícil ha sido que pasamos de tener clases a que todo son objetos, incluso las funciones. La solución es practicar, practicar, estudiar otros códigos y practicar.

Si bien el trabajar con un servicio web debería a ver sido fácil, encontramos más problemas de los imaginados. Fue difícil encontrar una librería en javascript que satisficiera los requisitos, y aun así hubo que realizar algunas modificaciones, este imprevisto supuso un retraso en los tiempos programados. El problema fue que la librería funcionaba bien para otros servicios web, pero al probarlo con el servicio de TUNE-UP el resultado era insatisfactorio, así que tuvimos que profundizar en el protocolo de los servicios web, especialmente en SOAP y modificar la librería en su cabecera SOAP. Buscamos servicios online que dieran soporte SOAP y estudiamos su estructura para saber dónde fallaba las llamadas a la librería y poder atacar el problema. Los links que nos ayudaron a conocer mejor la estructura fueron http://www.plumvoice.com/soaptester/, http://wsdlbrowser.com/soapclient?wsdl_url y la aplicación de escritorio SOAPSonar Personal.

El framework elegido para la interfaz de usuario, JQuery mobile, facilita mucho la presentación, permite crear temas personalizados, pero ya con el proyecto avanzado detectamos que la aplicación parecía lenta al lanzar sus eventos, investigando señalaban que la versión 1.4.2 era más lenta que su antecesora, este punto no se ha podido corroborar, porque tampoco es el objetivo comparar distintas versiones de este framework. También comprobamos que dependiendo del terminal, estos retrasos eran imperceptibles, cuanto más nuevo el terminal mejor funcionaba, su versión de Android también es más reciente. No modificamos la versión, y para posteriores revisiones sería bueno plantearse continuar con JQuery o probar con las demás opciones, Sencha, Kendo, Bootstrap, etc.

Para las pruebas de desarrollo utilizamos un Tablet, Archos 7 home Tablet con una versión de Android 1.5, y luego el despliegue lo probábamos en un Sony Ericcson con una versión de Android 2.1 (figura 48). El problema vino con terminales más nuevos, donde la aplicación se iniciaba, pero no tenía el comportamiento esperado. El problema fue debido a un error en el código html, teníamos unas etiquetas que no estaban cerradas, y con los terminales viejos obviaba este error de sintaxis y hasta que no utilizamos un terminal con una versión superior (Tablet modelo Idea Tab S6000-F con versión de Android 4.2.2) no pudimos simular el error y corregirlo.

Además tanto para el despliegue sobre iOS y Windows Phone 8 necesita de una firma, de la que no disponíamos ya que había que registrarse y tenía un coste. Con iOS la única posibilidad era sobre un MAC y el IDE XCode, pero solamente podríamos lanzar el emulador. En cambio para Windows Phone 8, teníamos más posibilidades, instalar el SDK, para lo que nuestro Sistema Operativo tenía que ser un Windows 8, de este modo podremos vincular un número limitado de terminales con el fin de testear la aplicación, o utilizar un servicio de Nokia para lanzar sobre terminales físicos remotos (http://developer.nokia.com/resources/remote-device-access).

Finalmente Phonegap saco la aplicación App Developper, que permitia lanzar la aplicación sobre cualquier terminal, esto nos dio una guía de donde podría estar el problema. La única solución posible a día de hoy es instalar los IDEs correspondientes para depurar los posibles errores.

Capítulo 8. Conclusiones y trabajos futuros

Hemos conseguido crear una aplicación móvil a partir de los requerimientos de movilidad y de shared context awareness en el marco de equipos de desarrollo de software. Se han identificado las funcionalidades que son interesantes en este ámbito y algunas de ellas se han implementado, como ha sido el kanban, donde veremos que tareas esta pendientes de iniciar y cuales estamos en desarrollo, enviar mensajes a otro agente desde una unidad de trabajo, conocer que alertas tenemos y pueden retrasar nuestro proyecto, las notificaciones y anuncios, que permiten al agente estar informado y confirmar la lectura de los avisos y finalmente toda la dinámica de los mensajes que permite la retroalimentación con los distintos agentes del grupo.

Se ha construido un prototipo totalmente funcional integrado en TUNE-UP, y hemos utilizado PhoneGap como tecnología multiplataforma para su desarrollo.

Importante reto personal al trabajar con tecnologías multiplataforma, concretamente con PhoneGap, ya que se trataba de una tecnología de la que carecía experiencia, y que ha supuesto un continuo reto.

Como trabajos futuros sería interesante estudiar la usabilidad para este tipo de aplicaciones móviles, ya que no corresponden al patrón de aplicaciones de escritorio ni de aplicaciones web. Habría también que estudiar otros aspectos de calidad en uso, en cuanto a eficacia, productividad, satisfacción y seguridad (atributos sugeridos por ISO 9126). Además habría que refinar aspectos tales como cuanta información debe contener una pantalla (teniendo en cuenta los diferentes formatos existentes, número máximo de botones, la profundidad, etc...), en definitiva crear una interfaz más atractiva.

Queda pendiente implementar las funcionalidades

- Dashboard para permitir a los agentes conocer la evolución diaria del proyecto
- Flujo acumulado que mostrara gráficamente como está evolucionando las unidades de trabajo en el workflow
- Tendencias del sprint que ayuda a la distribución de las cargas de trabajo sobre sprints de características similares
- Anuncios, que permite enviar avisos masivos sin posibilidad de respuesta

- Crear WUs, con una plantilla simplificada de datos

Queda pendiente la realización de las pruebas por parte de los usuarios, ello nos reportará una valiosa información que permitirá la mejora de la aplicación.

Finalmente surgen nuevas ideas de cómo utilizar el nuevo soporte móvil integrándolo a la aplicación, como puede ser en el caso de mensajes, el mensaje podría ser texto o una llamada telefónica registrando la llamada de igual modo que hacemos con el texto, o permitir el uso de la cámara para adjuntar documentos, en definitiva explorar los nuevos recursos que proporcionan los dispositivos móviles para mejorar la funcionalidad de la aplicación móvil.

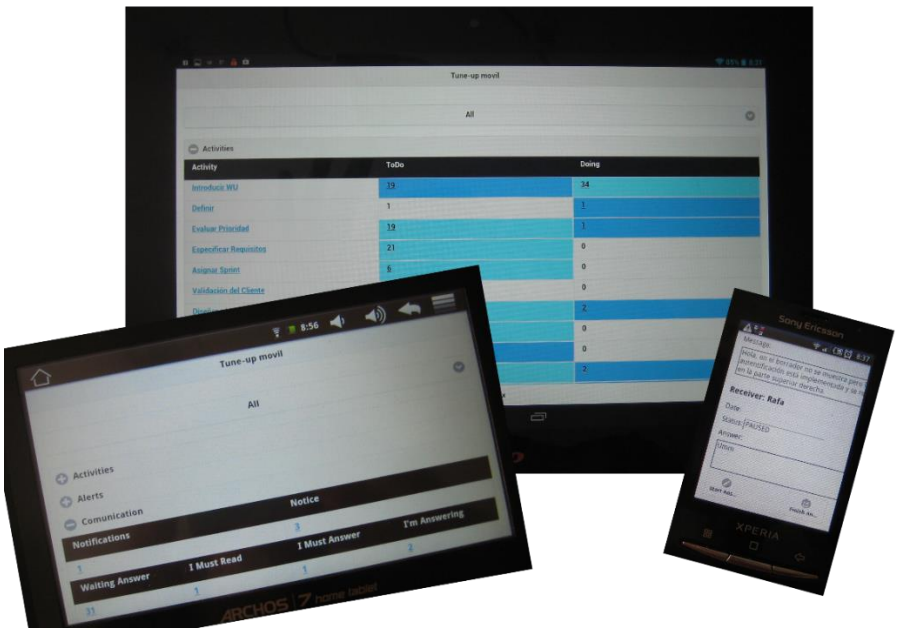


Figura 47. Terminales

Anexo I: Cuestionario tecnologías multiplataforma

Cuando realizamos el estudio de las tecnologías multiplataformas se realizó un cuestionario a las distintas tecnologías candidatas (PhoneGap, Titanium, Genexus, RhoMobile, Adienc, Velneo, Telerik, Xamarin, Appery.io) para ayudarnos a la seleccionar las tecnologías más adecuadas para realizar en ultimo termino la comparativa. Solamente dos respondieron a nuestra solicitud, por lo que no pudimos tomar como referencia este cuestionario, de todos modos agradecemos la colaboración de Genexus y Velneo y mostramos a continuación el cuestionario integro de Genexus y Velneo respectivamente.

Genexus

1. ***¿Cómo surge la idea de Genexus?***

La idea de GeneXus surge a mediados de los ochentas, donde los directores de la empresa se enfrentaron a un proyecto en el cual debía desarrollar un gran sistema con una base de datos corporativa. Es esos momentos las base de datos no tenían la popularidad que tienen hoy, los sistemas eran basados en archivos de datos fundamentalmente. El punto que este se trato de un proyecto en el cual la base de datos a implementar era de dimensiones importantes, pero el problema que las herramientas que se contaban en ese momento (algebra relacional, modelos entidades y relaciones, diagramas de Bachman, etc) no eran suficientes para diseñar y poder visualizar tamaña estructura.

Por otra para una vez diseñada era necesario generar todos los scripts para la creación de estructuras, índices, relaciones, etc.

Lo cierto que en esos años se comenzaba a hablar de lenguajes de inteligencia artificial, de forma que con los conocimiento que se tenían sobre bases de datos relacionales y estos lenguajes de inteligencia artificial, se crea la primer versión que permitía crear las estructuras de datos, con sus índices y relaciones, en forma totalmente automática.

El punto que una vez que se tenía ese conocimiento, resultaba sencillo poder realizar en esas estructuras altas, bajas, modificaciones y consultas, en forma totalmente automática, de esa manera se fueron agregando funcionalidades, las cuales con el pasar de los años nos traen al GeneXus que tenemos hoy.

2. ***¿Cuál es el objetivo final de Genexus?***

Si definimos GeneXus, es un programa que hace programas. Por lo cual el objetivo de GeneXus, es automatizar todo lo automatizable.

Si miramos las distintas industrias, todas ellas han tenido evoluciones en los procesos de elaboración, automatizando los mismos. En cambio el proceso de elaboración de software sigue siendo el mismo, en forma totalmente artesanal. Podemos decir que hoy existen librerías que ayudan, o componentes pre desarrollados sin lugar a dudas. Pero el proceso final de ensamble y desarrollo de software sigue siendo manual. En cambio con GeneXus, se cambia el paradigma ya que se describe en lugar de programar, y es en este caso la herramienta que desarrolla en lugar de la persona. Esto nos trae mejoras en el código generado y aumento de eficiencia en forma sustancial.

3. ***¿Considera que Genexus es una alternativa a las aplicaciones con código nativo?***

Si, sin dudas. Cada vez que se escribe una línea de código nativo, es por llamarlo de alguna manera una línea de código muerta. ¿Por qué lo digo de esta manera? Porque esa línea de código está asociada a un lenguaje y este irremediablemente tiene un tiempo de vida. Los lenguajes de programación evolucionan conforme pasa el tiempo y las líneas de código escritas en lenguajes antiguos no se pueden actualizar. En cambio cada linea de código que se declara en GeneXus, puede ser generada para cualquier lenguaje que la herramienta soporte y uno de los propósitos fundamentales de la empresa es estar en la última tecnología del momento. Así lo prueban más de 20 años de historia y todos los lenguajes que hemos soportado, soportamos y soportaremos, donde la lista es bastante grande.

4. ***¿Cuál es su cliente objetivo?***

Cualquier persona o empresa que tenga la necesidad de desarrollar alguna pieza de software. Es una herramienta ideal para casas de software, corporativos o entidades de gobierno.

5. ***¿Qué ventajas considera que tiene Genexus respecto a otras alternativas similares?***

Desde nuestro punto de vista hoy existen dos forma de desarrollar, una es la tradicional que todos conocemos y la otra es en forma descriptiva. Describir en lugar de programar, en esta segunda alternativa si bien hay algunas herramientas no existe una que realice tal totalidad del proceso en cuanto al desarrollo y mantenimiento de las aplicaciones. Algunas facilitan el desarrollo pero no realizan el mantenimiento, lo mismo sucede con la parte de la base de datos donde no todas las herramientas trabajan en la misma manera. En ese punto GeneXus es la única herramienta que desarrolla y mantiene el 100% de la aplicación, entendiendo como aplicación la totalidad de sus programas y estructuras de datos con las cuales estos interactúan.

Por tanto las ventajas con muchas con respecto a las alternativas, desde productividad, velocidad en el desarrollo, mantenimiento automático, flexibilidad en cuanto a la elección de plataformas, adaptabilidad a los cambios.

6. ***¿Porque eligen como herramienta de desarrollo html5,CSS3 y javascript?***

Por seguir los estándares que marca el mercado.

7. ***¿Creen que las metodologías ágiles (Scrum, Kanban) son un buen compañero de viaje para esta herramienta?***

Si bien GeneXus tiene una metodología propia de desarrollo, que es el desarrollo incremental basado en modelos. Cualquiera de las metodologías se pueden adaptar al desarrollo con GeneXus y tomar lo mejor de ambas cosas, una herramienta de alta productividad con metodologías avaladas por el mercado también.

8. ***Opinión de las aplicaciones móviles como complemento a herramientas de gestión ágil de proyectos***

Los niveles de adopción de las tecnologías móviles hacen que cualquier clase de aplicación móvil, mejor el proceso de gestión en los proyectos sin lugar a dudas. Esto permite al usuario tener una gran flexibilidad y velocidad en tiempos de respuesta, lo que hace que los tiempos 'muertos' pasen a ser minimizados en lo referente a gestión de los proyectos.

9. ***Aplicación móvil versus aplicación web (Ventajas e inconvenientes)***

Ni una ni otra, ambas! Cada una de ellas tiene un sentido. Sin dudas las aplicaciones Web nos permiten estar frente a una pantalla más grande y tener mayor detalle y expresividad en cuanto a la información. Por su lado las aplicaciones móviles nos brindan la posibilidad de estar conectados en cualquier momento y lugar.

Ahora si la discusión es aplicaciones web móvil versus nativas, la respuesta sería otra.

10. ***Aportación de las aplicaciones móviles en empresas ERP, CRM, BI.***

Todos los sistemas tienen módulos los cuales pueden ser utilizados en dispositivos móviles. Sin dudas pensar que un ERP, CRM, BI, HRM, etc, pueda ser 100% móvil entraríamos en un error. Esta movilidad que vivimos el día de hoy nos brinda una nueva plataforma y es de esa manera que debemos de analizarla. Por lo cual debemos de ver que cosas se pueden o se necesitan tener al alcance de la mano en un dispositivo móvil.

Velneo

1. ***¿Cómo surge la idea de Velneo?***

Las herramientas que manejaba Juan Muñoz-Cobos a principios de los 80 le obligaban a repetir tareas tediosas una y otra vez:

- *No se podía reaprovechar nada hecho*

- *Había que dominar varias tecnologías e idiomas (si con suerte había documentación, normalmente estaba en inglés)*
- *A principios de los 80 no había bases de datos, y mucho menos relacionales. Solo había ficheros (y como mucho indexados)*
- *La programación no era asistida, estaba basada en líneas y líneas de código, lo que provocaba muchos errores durante el desarrollo*
- *Las modificaciones en producción solían ser auténticas películas de suspense y terror*
- *Las plataformas eran muy generalistas y poco productivas para sus necesidades*

En 1.986 harto de todos estos inconvenientes, empezó a trabajar en un asistente de programación para uso propio. Un sistema de indexados y tablas relacionadas. La nochebuena de ese mismo año, antes de acudir a la típica cena familiar pudo probar su creación. Ese fue el auténtico nacimiento de Velázquez Visual.

En 1.992, aprovechando una mayor estabilidad de C, tomó la decisión de hacer algo más elaborado, algo que evitara a desarrolladores como él las mismas frustraciones que había sufrido los años anteriores. Reescribió Velázquez Visual y se lanzó a ofrecerlo a otros desarrolladores, dejando así de hacer productos finales.

Puedes profundizar en la historia en el siguiente post.

2. ***¿Cuál es el objetivo final de Velneo?***

El objetivo de Velneo es desarrollar aplicaciones para empresas de manera sencilla y rápida.

Velneo V7 cuenta con todos los componentes y recursos necesarios para analizar, documentar, desarrollar, mantener e implantar soluciones empresariales tanto en local como en la nube.

Velneo V7 está orientada al desarrollo de aplicaciones empresariales y cuenta con características avanzadas que te permiten obtener la máxima rentabilidad en tus desarrollos.

3. ***¿Considera que Velneo es una alternativa a las aplicaciones con código nativo?***

Velneo no es una alternativa, es una solución a problemas concretos de la industria de desarrollo de software para empresas. El desarrollo de software está avanzando rápidamente, los programadores quieren soluciones a sus problemas concretos. Da igual que uses o no código nativo, lo importante es que soluciones problemas y Velneo facilita la vida de muchos programadores.

4. ***¿Cuál es su cliente objetivo?***

Este es el perfil de cliente que más nos compra hoy día:

Un programador de 34 a 44 años que desarrolla para vender programas con una plataforma sin evolución (VB, Fox Pro, Cobol, Clipper,…) o que su plataforma es complicada y cara de mantener (.NET o Java) que vive en (España, México, Chile, Colombia, Uruguay o Argentina) y que tiene su domicilio en las ciudades de (Madrid, Bogota, Santiago de Chile, Mexico City, Barcelona y Buenos Aires). Este programador tiene que cubrir las necesidades de desarrollar software para empresas.

5. ***¿Que ventajas considera que tiene Velneo respecto a otras alternativas similares?***

Estas son las características más nombradas por los clientes actuales en la encuesta de satisfacción y por tanto consideramos que son nuestras ventajas:

Open Apps, producto en Español, robustez, simplicidad, reusabilidad del código, plataforma de desarrollo integral, Javascript, personal/equipo, refactorización, comunidad, multiplataforma, fácil aprendizaje, cloud, blog y foro, facilidad para desarrollar, desarrollo ágil, cercanía, Base de Datos, rapidez al programar, el hecho de no tener que escribir código, evolución, fiabilidad, soporte/asistencia técnica y herencia.

6. ***¿Creen que las metodologías ágiles (Scrum, Kanban) son un buen compañero de viaje para esta herramienta?***

Cualquier metodología de desarrollo bien implantada en un equipo es un buen compañero para la plataforma. Nuestros clientes nos consideran una herramienta ágil y rápida de desarrollo por tanto, un scrum o kanban bien implantados puede traer grandes ventajas al desarrollo con Velneo.

7. ***Opinión de las aplicaciones móviles como complemento a herramientas de gestión ágil de proyectos***

El mundo móvil y escritorio, están confluyendo, ¿Qué es móvil? ¿Qué es escritorio? ¿Qué es un portátil? Las ventajas son las mismas, simplemente son dispositivos que interactúan con la información. Cualquier soporte que ayude a la gestión de un proyecto será bienvenida, sea en un PC, TV, portátil o móvil.

8. ***Aplicación móvil versus aplicación web (Ventajas e inconvenientes)***

Cada uno tiene su aplicación, hay que saber muy bien cuando usar que, es el debate entre creación y consumo de contenidos. Si quieres crear contenido e interactuar con él, sin duda la mejor opción son las aplicaciones, si quieres consumir contenido lo más probable que lo mejor sea la web.

En nuestra plataforma puedes crear aplicaciones móviles y aplicaciones web.

Estas son las ventajas de crear aplicaciones:

- *Experiencia de usuario, si quieres experimentar una gran experiencia, es necesario que hagas una aplicación, te permitirá crear controles e interactuar con el dispositivo de otra manera. Facebook, Twitter, Gmail, tienen todas web, pero al final lo que más se usa en los dispositivos móviles son las apps ya que la experiencia es superior.*
- *Usar capacidades locales del dispositivo, si quieres usar todas las capacidades locales del dispositivo donde está ejecutándose tu código lo mejor es usar apps.*
- *Si lo que quieren conseguir es aplicaciones orientadas acciones, la app es la solución.*
- *Es menos costoso de desarrollar y mantener una app que una web.*
- *Personalizar el entorno para un usuario es muy complicado con una web, si necesitas que tus usuarios disfruten de personalización la app es la solución más práctica.*
- *Cálculos complejos, informes y gráficos.*

Ventajas de crear web:

- *Indexación en buscadores.*
- *Fácil de linkar y navegar por la web.*
- *No necesita instalación ni actualización.*

Artículo Aplicaciones vs Webs

9. Aportación de las aplicaciones móviles en empresas ERP, CRM, BI

Como he comentado antes, los móviles son nuevos dispositivos, de distintos tamaños y funciones donde podrás interactuar con la información de tu empresa. Antes sólo podías ver los pedidos desde el ordenador de la oficina y hoy lo puedes ver desde tu móvil o tablet. Al tener más diversidad aumentan las posibilidades y la productividad de las empresas.

Referencias

1. http://es.wikipedia.org/wiki/Consciencia_situacional
2. Ana León Palacio. Shared Situational Awareness aplicado a equipos de trabajo de desarrollo software (Proyecto fin de carrera, julio 2013)
3. Theoretical underpinnings of situation awareness: a critical review. Mica R Endsley and Garland D.J (2000)
4. Toward a theory of situation awareness in dynamic systems. Mica R Endsley (1995).
5. The Importance of Awareness for Team Cognition in Distributed Collaboration. Carl Gutwin and Saul Greenberg (2004)
6. Joan Ribas Lequerica. Desarrollo de aplicaciones para Android (2013)
7. Web oficial de PhoneGap. http://www.PhoneGap.com
8. Web oficial de Titanium http://www.appcelerator.com/titanium/
9. Web oficial de Genexus http://www.genexus.com/
10. La Génesis de Genexus. Breogán Gonda y Nicolás Jodal (2010)
11. Mobile and Smart Devices Development Solution. Genexus @ (2014)
12. RhMobile Suite http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite
13. Definición MVC. http://es.wikipedia.org/wiki/Modelo_Vista_Controlador
14. Pattern-Oriented Software Architecture. A system of patterns. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. (1996)
15. http://www.TUNE-UPprocess.com/ (TUNE-UP Overview)
16. Seguimiento ágil de proyectos de desarrollo de software utilizando Gráficas Burn Down. Marisa Isabel Marante, María Company y Patricio Letelier
17. TUNE-UP: Un enfoque pragmático para la planificación y seguimiento de proyectos de desarrollo y mantenimiento de software. Maria Isabel Marante, Patricio Letelier y Francisco Suárez.
18. http://help-pep.tuneupprocess.com/, http://help-globaldashboard.tuneupprocess.com/ (TUNE-UP Software Process)
19. http://www.codeproject.com/Articles/12816/JavaScript-SOAP-Client (Matteo Casati - Librería cliente JavaScript SOAP)

20. JIRA, documentación. https://confluence.atlassian.com/display/AOD/Using+JIRA+on+a+Mobile+Device
21. Rally software, http://www.rallydev.com/about/rally-software-acquires-mobile-application-iphone
22. TargetProcess, descarga iTunes https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewSoftware?id=593808012
23. VersionOne, http://ios-apps.findthebest.com/l/804604/V1-Mobile-VersionOne-for-mobile-devices
24. Material asignatura HMI (Herramientas CASE y Métodos Semiformales en Ingeniería del Software) impartida por el profesor Patricio Letelier, DSIC-UPV, 20123