UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

TÍTULO DEL INFORME

"Diseño de un sistema Web para una empresa Internacional"

MODALIDAD DE TITULACIÓN:

"Experiencia Profesional"

Para obtener el título de:

Ingeniero en Computación

P R E S E N T A:

José Ulises Vargas García

ASESORA:

M. I. NORMA ELVA CHÁVEZ RODRÍGUEZ

CIUDAD UNIVERSITARIA MÉXICO D. F. 2012

Índice

INTRODUCCIÓN

A lo largo de la vida profesional se van adquiriendo diferentes conocimientos, muchos de ellos técnicos que complementan de una manera significativa los adquiridos en la etapa escolar, y son importantísimos para nuestro desarrollo, sin embargo hay otros de forma de trabajo, de manejo de personas y hasta cómo controlar nuestro carácter en determinadas situaciones, todos estos conocimientos son importantes.

En este documento si bien es primordial el aspecto técnico destaca el aspecto organizacional y forma de trabajo de una empresa (el cliente al cual le trabajo). Es una ideología que debería copiarse en muchas de las empresas de nuestro país.

La formación que nos proporciona la Facultad de Ingeniería y por la cual estoy muy agradecido es integral, si bien el aspecto técnico es fundamental, todo el complemento humanístico es indispensable para desarrollarse, el aspecto cultural es súmamente importante para relacionarse y no digamos el idioma inglés que es una herramienta muy valiosa para poder conseguir mejor calidad de vida al salir de los estudios.

He tenido la fortuna de desarrollarme en diversos ambientes y puedo decir con agrado que he avanzado significativamente desde que comencé mi vida profesional. Y puedo decir que gracias a la Facultad de Ingeniería y a la Universidad Nacional Autónoma de México soy un buen profesionista y una persona mejor de lo que pude haber sido sin ellas.

Objetivo

Llevar a cabo un análisis de la metodología, las prácticas de trabajo y el desarrollo para diseñar e implementar un sistema Web para una empresa Internacional..

Definición del problema

Diseñar e implementar un sistema Web para una empresa Internacional, el desarrollo es en C# y .NET, con metodología de desarrollo de software Scrum y las prácticas de desarrollo son algunas de XP (e**X**treme **P**rogramming) Programación extrema.

Método

La metodología de desarrollo de Software utilizada fue Scrum, las técnicas y herramientas Pair Programming, Unit Testing, Knockout**,** Jasmin**,** las prácticas de desarrollo son algunas de XP (eXtreme Programming)o Programación extrema.

Metodología de Desarrollo

Método:

Con el uso de un marco de trabajo genérico lo podemos aplicar al desarrollo del proyecto en cuestión, el cual consta de varias tareas que incluyen:

Comunicación. Esta actividad del marco de trabajo implica una intensa colaboración y comunicación con los clientes; además, abarca la investigación de requisitos y otras actividades relacionadas.

Planeación. Esta actividad establece un plan para el trabajo de la ingeniería del software. Describe las tareas técnicas que deben realizarse, los riesgos probables, los recursos que serán requeridos, los productos del trabajo que han de producirse y un programa de trabajo.

Modelado. Esta actividad abarca la creación de modelos que permiten al desarrollador y al cliente entender mejor los requisitos del software y el diseño que logrará satisfacerlos.

Construcción. Esta actividad combina la generación del código (ya sea manual o automatizado) y la realización de pruebas necesarias para descubrir errores en el código.

Despliegue. El software (como una entidad completa o un incremento completado de manera parcial) se entrega al cliente, quien evalúa el producto recibido y proporciona información basada en su evaluación.

La actividad de comunicación

La ingeniería de requisitos proporciona un mecanismo para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación, y administrar los requisitos conforme estos se transforman en un sistema operacional. El proceso de la ingeniería de requisitos se lleva a cabo a través de siete distintas funciones: inicio, obtención, elaboración, negociación, especificación, validación y gestión.

Resulta importante declarar que algunas de estas funciones de la ingeniería de requisitos ocurren en paralelo y que todas deben adaptarse a las necesidades del proyecto. Todas están dirigidas a definir lo que el cliente quiere, y todas sirven para establecer una base sólida respecto del diseño y la construcción de lo que obtendrá el cliente.

La actividad de la planeación

La actividad de comunicación ayuda al equipo de software a definir sus metas y objetivos generales (por supuesto, sujeto al cambio conforme pasa el tiempo). Sin embargo, entender esas metas y objetivos no es lo mismo que definir un plan para llegar a ellos. La actividad de planeación abarca un conjunto de prácticas técnicas y de gestión que permiten al equipo de software definir un mapa de camino mientras se viaja a través de su meta estratégica y objetivos tácticos.

Actividad de modelado

Los modelos se crean para obtener un mejor entendimiento de la entidad real que se construirá. El modelo debe ser capaz de representar la información que el software transforma, la arquitectura y las funciones que permiten que ocurran las transformaciones, las características que desean los usuarios, y el comportamiento del sistema como se realiza la transformación. Los modelos deben cumplir estos

objetivos en diferentes grados de abstracción (primero al presentar al software desde el punto de vista del cliente y después al presentar el software en un nivel más técnico).

En el trabajo de la ingeniería de software se crean dos software se crean dos clases de modelos: modelos de análisis y modelos de diseño. Los modelos de análisis representan los requisitos del cliente al presentar el software en tres dominios diferentes: el dominio de la información, el dominio funcional y el dominio del comportamiento. Los modelos de diseño representan características del software que ayudan a los profesionales a construirlo de manera efectiva: la arquitectura, la interfaz de usuario y el detalle a nivel de componentes.

La actividad de construcción

La actividad de construcción abarca una serie de tareas de codificación y realización de pruebas que conducen al software operativo que está listo para entregarlo al cliente o usuario final. En el trabajo de software moderno la codificación puede ser: 1) la creación directa de código fuente de un lenguaje de programación; 2) la generación de código fuente al utilizar una representación intermedia de diseño de componente que será construido; 3) la generación automática de código mediante un lenguaje de programación.

La actividad de despliegue

La actividad de despliegue abarca tres acciones: entrega, soporte y retroalimentación. Como el software moderno es evolutivo por naturaleza, el despliegue no se presenta una sola vez, sino varias veces conforme el software avanza hacia su terminación. Cada ciclo de entrega le proporciona al cliente y a los usuarios finales un incremento de software operativo que provee funciones y características útiles.

Cada ciclo de soporte proporciona documentación y asistencia humana para todas las funciones y características introducidas durante todos los ciclos de despliegue que se presenten. Cada ciclo de retroalimentación ofrece al equipo de software una guía importante que conduce a modificaciones en las funciones y el enfoque que se toma para el siguiente incremento.

El presente documento es una demostración en papel de un proyecto realizado en una empresa, si bien explica código no es ni pretende ser un tutorial, aunque puede ser una referencia de una experiencia profesional detallada, al no ser un tutorial no se hace un análisis más profundo y detallado del código.

También doy una semblanza de mi paso por la vida laboral y las características de las dos empresas para las que trabajo.

He aquí un breve resumen de lo que este documento presenta:

Capítulo 1: Proyectos anteriores

Es un breve resumen de la experiencia adquirida en anteriores trabajos en los que he tenido la oportunidad de colaborar.

Capítulo 2: Quarksoft S.A. de C.V. y Daktronics.

Son las características de las empresas para las que me contrato y es interesante ver también las diferencias entre ellas.

Capítulo 3: Descripción del puesto de trabajo.

Aquí podemos ver lo diferente que son las características que tengo con respecto a las de la empresa que me contrata, sin embargo el proyecto o cliente requería alguien con mis características.

Capítulo 4: WorkTicket

Las herramientas que fueron necesarias para el llevar a cabo el proyecto y la descripción del desarrollo del sistema llamado "*WorkTicket"*.

CAPÍTULO 1: Proyectos anteriores

Algunos de las empresas y proyectos en los que he participado comenzando por el más reciente antes del actual y del que trata este documento.

1.1 LiveOps

En este proyecto participé ya estando en esta empresa con la que estoy contratado QuarkSoft y fue mi primer proyecto.

LiveOps es una empresa en San José California USA, uno de los centros de llamadas (call center), más grandes de aquel país, entre sus clientes se encuentran empresas de televentas, eventos masivos de televisión como american Idol, pizzerias, etc, recibiendo miles de llamadas y transacciones por hora. Una de las características importantes de esta empresa es que su personal de call center trabaja remotamente desde su domicilio, eligiendo los proyectos y las horas en las cuales quieren trabajar.

El trabajo inicialmente fue el mantenimiento de sus sistemas. El código de estos está escrito en Perl, orientado a web, montado sobre Apache. Como complemento usan JavaScript y Mason entre otros. Con una base de datos MySql.

Este proyecto sumamente interesante donde por primera vez utilicé en equipo una metodología de desarrollo en forma real, Scrum. El trabajo al igual que en algunos trabajos anteriores era tener el ambiente de desarrollo en la máquina local. Con una base de datos local.

El *"bug tracking"* que utilicé es bugzilla y el control de versiones CVS.

Esta empresa tiene una gran organización la cual se ve reflejada en su método de trabajo, el cual consiste en que después de haber planeado el trabajo por medio de Scrum, se procedía a resolver las historias, que consistían en historias o *"bugs"*, después de esto el contenido se "subía" al servidor de control de versiones con el fin de que el revisor lo pudiera ver y hacerle un *"code review"* indispensable que lo hacía algún compañero desarrollador. Tarea que muchas veces nos tocaba a nosotros hacer para algún otro compañero.

Después de realizar los cambios propuestos por el compañero que revisó el código, pasa este *"Patch"* o parche al departamento de QA ("Quality Assurance") o Aseguramiento de la Calidad, que revisa que el parche propuesto no afecte nada en el sistema, esto es probado en un servidor que emula completamente el real, se le aplican de ser necesario pruebas de regresión para lograr que todo esté correcto.

Al terminar estas pruebas, pasa a ser programado para su "deployment" o liberación en el servidor de producción. La cual se hacía de manera periódica de dos semanas, y se hacía para todos los parches o mejoras hechas, probadas y aprobadas en ese lapso. En la figura 1.1: se observa el diagrama de bloques general

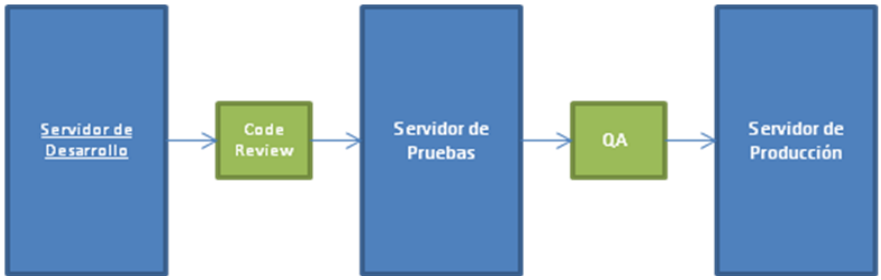


Figura 1.1 Diagrama de Bloques general

Las verificaciones de tarjetas de crédito cuando el cliente realiza los pagos (recordemos que es un call center, que admite pagos mediante tarjetas de crédito) fueron las primeras tareas con las que me relacioné, muy interesante la forma de verificarlas, no lo hacíamos directamente la empresa, sino que el servicio lo proporcionan varias empresas la información se manda por distintos métodos, todos ellos seguros y encriptados, por medio de SOAP, o algún otro protocolo y poder llevar a cabo esa transacción por medio del lenguaje. Hubo proyectos muy interesantes con esta empresa.

1.2 DWC

Este trabajo lo conseguí vía remota a través de una página de empleos por internet. La empresa se encuentra en Minesota, USA.

Distributed Website Corporation. La empresa creó un gran sistema web que vendía a varios consumidores, su principal mercado son escuelas.

En este sistema llevaban control de personal, alumnos, micrositios de clases, tareas, usuarios, incluso exámenes en línea.

Tenían algunos bancos como usuarios en los bancos únicamente se llevaba la página y un *"frontend"* de acceso al sistema del banco.

El sistema está construido en Perl, aunque cuando entré estaban migrando a un CMS llamado drupal (php).

Entré para hacer el mantenimiento y realizar nuevas implementaciones al sistema, uno de los proyectos más interesantes fue hacer una conexión del sistema a una base de datos que ya teníen ellos, y pudieran accessar con su usuario y contraseña que ya poseían y no tener que tener múltiples usuarios y contraseñas, así que tuve que hacer que la autenticación se hiciera a través de nuestro

sistema por medio de LDAP autentificando usuario y contraseña hacia su sistema y regresar si fue exitoso y así tener su accesso al sitio.

En general había muchas peticiones de las escuelas que obviamente las cobraban y me las pasaban a mí o a otros miembros del equipo para llevarlas a cabo.

No teníamos ninguna metodología de trabajo, únicamente la asignación de tickets por medio de Bugzilla, el jefe directo los asignaba y teníamos cierto tiempo para entregarlo.

Teníamos contactos para poder enterarnos bien del problema pero en realidad se sentía el ambiente poco amable.

A pesar de ser una compañía americana el ambiente no era particularmente amigable como generalmente se siente, esto debido a que había muchos trabajadores también filipinos en particular el jefe directo, quien tenía el perfil clásico de un jefe de tercer mundo.

1.3 ASE

Agencia de Seguridad Estatal del Estado de México, fui contratado para construir un sistema que se encargara de llevar el reporte de los recursos asignados tanto por el gobierno federal como por el estatal para los distintos ejes y programas que tenía esta agencia para el combate a la inseguridad.

El sistema fue para web, hecho en lenguaje perl con una base de datos mysql sobre un servidor apache, en un sistema operativo Windows.

El proyecto llevó tiempo pues nadie era especialista en la operación, me tomó mucho tiempo llevar a cabo toda la recopilación de la lógica del funcionamiento de este.

No había organización en realidad en esta dependencia de gobierno estatal, el ambiente es demasiado pesado, y según mi experiencia adquirida allí, puedo decir que uno de los factores determinantes para estar en un trabajo es el ambiente laboral.

1.4 UPS

United Parcel Service de México. Es la mayor empresa de entrega de paquetes y mensajería express del mundo y el líder en servicios de transporte, logística, capital y comercio electrónico.

Sus oficinas se encuentran al sur de la ciudad sin embargo el centro operativo está ene l aeropuerto donde laboré, estaba contratado para soporte de segundo nivel.

En esta empresa no programé, estuve haciendo Soporte a máquinas, servidores y sistemas, asistido por el proveedor de estos, soporte a redes, telefonía, etc.

Es una empresa que a pesar de ser de un tremendo tamaño a nivel mundial, no tiene gran organización, sin embargo llevábamos a cabo el trabajo, el equipo no era muy grande. Y se dividía en los que asistían a sitios con los clientes y los que operábamos dentro de la empresa, entre algunos proyectos fue la renovación de cableado área de callcenter. Y actualizaciones del software, asistidos por el proveedor en USA.

CAPÍTULO 2: Quarksoft S.A. de C.V. y Daktronics.

2.1 Quarksoft

Sinopsis de la empresa:

QuarkSoft se constituye el 7 de junio del 2001, su objeto social radica principalmente en el asesoramiento técnico, económico financiero y consultoría; Prestan servicios a sociedades cuyo objeto social sea cualquier actividad de carácter informático, de telecomunicaciones o de desarrollo de nuevas tecnologías incluidas la consultoría, así mismo para las áreas de software o programación y también hardware o equipos de cómputo y accesorios.

QuarkSoft basa sus operaciones en el aseguramiento de la calidad y el establecimiento de procesos. Por ello, utiliza las tecnologías más recientes en procesos, aseguramiento de la calidad y en tecnologías de la información. QuarkSoft es una de las contadas empresas mexicanas que está utilizando el modelo Capability Maturity Model® Integration (CMMI) y las metodologías Personal Software Process ^{SM} (PSP ^{SM}) y Team Software Process ^{SM} (TSP ^{SM}) del Software Engineering Institute (SEI) de manera exitosa para lograr sus objetivos.

QuarkSoft cuenta con 4 instructores certificados en Personal Software Process ^{SM} (PSP ^{SM}), dos coach certificados en Team Software Process ^{SM} (TSP ^{SM}) y el apoyo de un SCAMPI Lead Appraiser.

Actualmente QuarkSoft cuenta con un poco más de 200 ingenieros de Software repartidos en las oficinas de San Jose California, Madrid España, Ciudad de México y Centros de Desarrollo en Zacatecas, León y Aguascalientes.

Para QuarkSoft la calidad de un producto depende de la calidad de los procesos que se utilizaron para su elaboración. Están convencidos que nada da más satisfacción, pública y privada que realizar trabajo con calidad, porque da seguridad, confianza y en corto o largo plazo, será reconocido como un gran trabajo. El secreto de su éxito es la pasión, emoción y creer en ellos mismos.

Misión.

Proveer la mejor solución para nuestros clientes, basada en tecnología y software de alta calidad, que lo satisfaga permanentemente y sea motivo de orgullo para nosotros, usando innovación continua en tecnologías de la información, procesos, calidad y talento humano, y la colaboración armónica y creativa con nuestro entorno, generando valor a la compañía y seguridad laboral para los que trabajamos en ella.

Visión.

Ser los líderes en el mercado de investigación, formación y desarrollo de software, marcando la pauta en el uso de tecnologías de ingeniería de software y un modelo a seguir donde los colaboradores y clientes satisfagan sus necesidades a la vez que crea una organización que fomenta el desarrollo profesional, personal, familiar y social.

Filosofía de trabajo.

QuarkSoft basa su cultura organizacional en el mejoramiento continuo y en el aseguramiento de la calidad. Se considera primordial guardar un balance entre procesos, recurso humano y tecnología. Para ello, se trabaja bajo la filosofía y mejoramiento continuo de procesos. Por otro lado, se fomenta el uso de metodologías, tecnologías recientes, controles estadísticos y métricas.

Calidad.

- 0.36 defectos por cada 1000 líneas de código (lo que implica menos del 4% de mantenimiento)
- +- 15% desviación del plan (Schedule deviaton)
- Ahorros entre un 30% y hasta un 70% en mantenimientos
- Expertos en la implementación de PSP SM y TSP SM

Reconocimientos.

- QuarkSoft es ganador del Premio Nacional a la Innovación 2008 Dell – American Express – Endeavor.

- QuarkSoft es ganador del segundo lugar a nivel internacional del premio Dell Small Business Excellence Award 2008.

- QuarkSoft es una empresa Endeavor.

- Quarksoft recibió el Premio Nacional IMDA 2008 por parte del Estado de Zacatecas por el desarrollo de “Kioskos de Servicios Electrónicos” un proyecto innovador para la modernización y desarrollo administrativo en la Administración Pública.

- QuarkSoft recibió el Premio Nacional de la Secretaría de Economía TechBA (Technology Business Acelerator).

- QuarkSoft recibió el premio Emprendedores 2009 por parte de la revista Expansión.

- Quarksoft recibió el premio Entrepreneur of the year 2010.

El organigrama general de la empresa se muestra en la figura 2.1

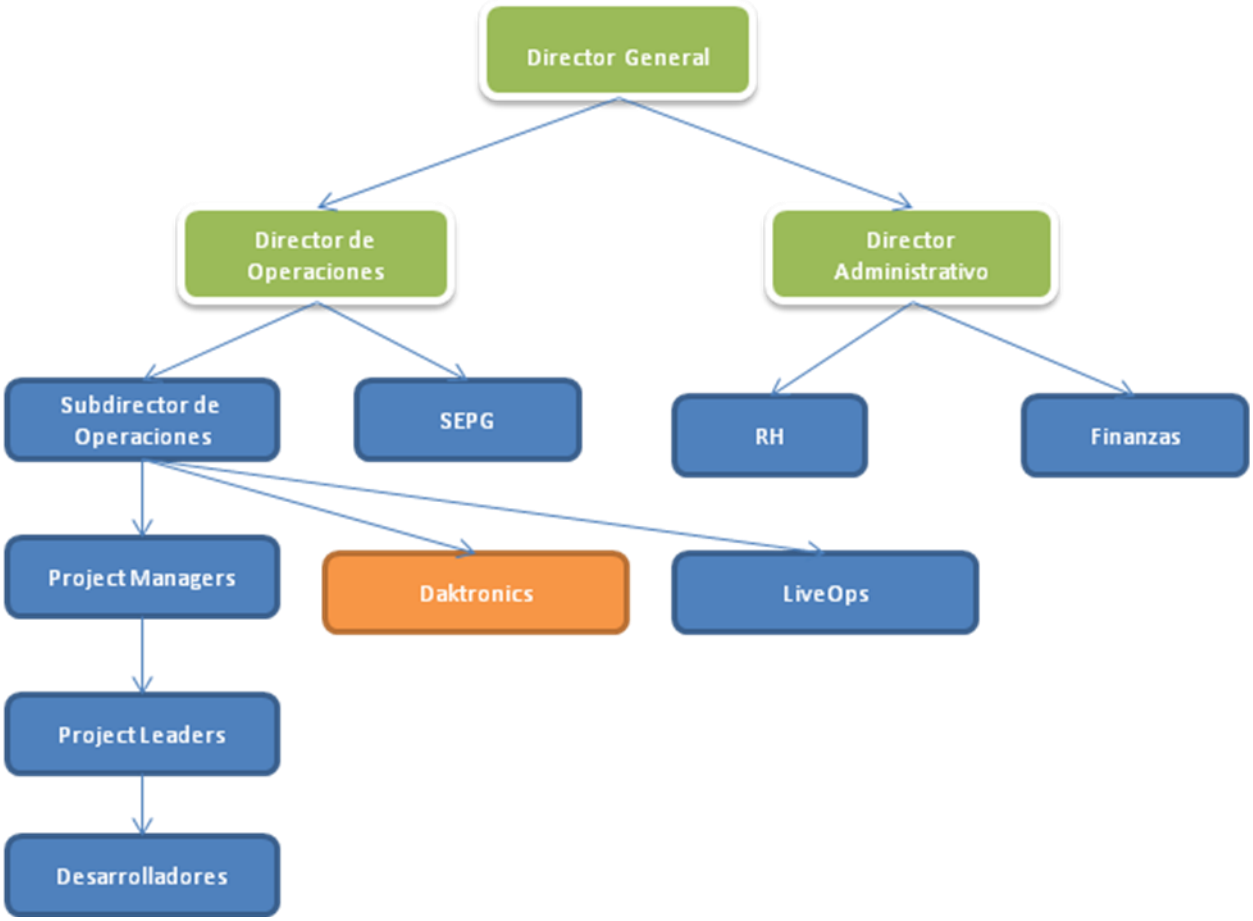


Figura 2.1. Organigrama general de la empresa

2.2 Daktronics

Daktronics® es una empresa dedicada a la construcción de tableros de leds y displays industriales, deportivos etc. Es una empresa de alcance global, la recién inaugurada Arena de la Ciudad de México tiene tableros o displays de esta empresa, la pantalla externa es la más grande construida por esta empresa. El nuevo estadio Corona del equipo de futbol Santos de Torreón tiene también pantallas de esta empresa y Times Square en Nueva York, Estados Unidos tiene el 80% de sus pantallas luminosas de la marca Daktronics®.

Daktronics es una empresa reconocida a nivel internacional por ser una de las más grandes del orbe en la fabricación y comercialización de displays de mediano y gran tamaño, para intemperie o para zonas cerradas.

Esta empresa se ubica en Brookings, Dakota del Sur, en Estados Unidos, Tiene un extenso departamento de Tecnologías de la Información (IT) y cuenta con excelente plataforma y muy buen presupuesto para mantenimiento de sistemas y realización de nuevos proyectos. Su plataforma está cimentada en la gama de productos Microsoft® así como lenguajes de esta misma empresa.

Su lenguaje principal es C# y su framework es .NET, su IDE es Visual Studio 2010. Tanto el control de versiones como el seguimiento de proyectos se hace por medio de TFS® (Team Fundation Services ®). Su servidor de bases de datos es SQL Server 2008 y su webserver IIS® (Internet Information Services®)

Su metodología de trabajo es Scrum. Scrum es un marco de trabajo para la gestión y desarrollo de software, el cual ya lo había utilizado yo en el proyecto anterior y conocido en trabajos anteriores. De hecho el tener experiencia en esta metodología fue una de las razones por las cuales me aceptaron en el nuevo proyecto.

Vale la pena hablar ampliamente de esta metodología, que desde mi personal punto de vista es la más efectiva, lo haremos en el tema de herramientas utilizadas en el proyecto.

Tienen diversos sistemas para el manejo de interno tanto de ventas como de cotizaciones, alguno de control de personal, diseño, transporte, consulta, todos ellos para usuarios internos. No se tiene ningún software para clientes externos, excepto el que programa los displays.

La responsabilidad para mi en esta compañía fue, inicialmente, trabajar en el mantenimiento de sus sistemas que después de varios meses derivó en trabajar en nuevos componentes para sus proyectos existentes y nuevos proyectos. Su personal de software son aproximadamente 20 personas, entre desarrolladores, QA's business analists y DBA´s .Existe mucho más personal para soporte técnico, redes, etc. Sin embargo con ellos no tenemos mucha relación, únicamente cuando se nos presenta un problema de este tipo

La forma de trabajo remota es mediante una VPN Cisco y con eso tenemos acceso a las intranet y poder conectarnos a los servidores y bases de datos.

El hardware es proporcionado por el cliente, cabe destacar que son muy potentes las estaciones de trabajo que nos dieron para trabajar, y sin duda nos ha servido para poder desarrollar nuestro trabajo

correctamente. Y tienen una política de uso de estas estaciones muy estricta, con la cual ya tuve un par de incidentes por conectar un disco duro externo.

Trabajamos con un código local en nuestras laptops, proporcionado por nuestro Control de Versiones de TFS. Sin embargo la base de datos de desarrollo es única para todos los desarrolladores, situación que en ocasiones nos ha complicado un poco el trabajo al momento de depurar (debugg) los cambios. Aunque no ha sido de forma muy importante.

Se cuenta también con una base de datos de pruebas, y una de producción. Las cuales ninguna de las dos tenemos acceso los desarrolladores, es decir, para manipularlas, nuestros permisos son de sólo lectura.

Hasta el momento he trabajado en 6 diferentes proyectos entre mantenimiento y nuevas mejoras a los que ya existen y un nuevo proyecto.

Capítulo 3: Descripción del puesto de trabajo.

QuarkSoft tiene básicamente dos divisiones, México y Estados Unidos, la división americana es completamente distinta a la mexicana, el dueño de la empresa en EUA decidió contratar mano de obra mexicana para empresas americanas y necesitaba un socio aquí, pues él al ser extranjero no puede establecer una empresa en territorio mexicano, así que buscó un socio y QuarkSoft le ofreció lo que él buscaba. Así pues, los perfiles contratados para los clientes americanos, conseguidos por QuarkSoft-LLC (QuarkSoft - Limited Liability Company), no son exactamente los mismos que los especificados en QuarkSoft México.

MI perfil a diferencia de los especificados era en Lenguaje Perl y metodología SCRUM.
En QuarkSoft manejan CMMI.

Otra diferencia es que no fabricamos software a la medida como lo hace QuarkSoft México, nosotros trabajamos prácticamente para el Cliente. Cualquier cosa que sea relacionada al trabajo la tratamos con el cliente, no tenemos un Líder de Proyecto, sí contamos con un Project Manager que se encarga de ser nuestro enlace administrativo entre el cliente y el Director de QuarkSoft-LLC.

Por lo escrito arriba, a pesar de pertenecer a QuarkSoft mi perfil no es exactamente el que se describe para mi puesto en la empresa, así que describiré con más detalle el perfil que manejo.

El lenguaje que manejaba hasta antes de comenzar el proyecto “Daktronics” era Perl, con el cual llevo trabajando cerca de 8 años, había trabajado en distintos proyectos personales y después empecé a trabajar de manera independiente en empresas estadounidenses, gracias a un muy buen amigo que conocí en un trabajo anterior, por la peculiaridad del lenguaje, (Perl es software libre y está licenciado bajo la Licencia Artística y la GNU General Public License) Se lleva de maravilla con MySQL que es el manejador de bases de datos que use hasta antes de Daktronics, y al trabajar con empresas americanas, quienes tienen mayormente difundida la metodología Scrum, aprendí a utilizarla, mi formación digamos estaba más enfocada fuera de los estándares que utiliza empresa para quien trabajo.

Sin embargo el primer proyecto en el que tuve la oportunidad de trabajar en QuarkSoft fue LiveOps y precisamente estaba construido bajo este software y llevaban esta metodología de desarrollo y como ya he comentado me querían como integrante del equipo que ellos manejaban, por lo tanto la descripción de mi puesto está más ligado al cliente que a QuarkSoft.

En el actual proyecto (Daktronics) Estoy bajo el mismo esquema que LiveOps, por lo tanto el perfil es más parecido a LiveOps que a QuarkSoft, prácticamente estoy como un empleado más del cliente, la ‘única diferencia es que estoy vía remota, le reporto directamente al cliente, las vacaciones las trato con el cliente.

Y referente al perfil, al cliente le interesaba que tuviera conocimientos sólidos en Scrum, que tuviera experiencia en ambientes empresariales de trabajo, y experiencia en mantenimiento de sistemas, la limitante era el lenguaje de programación, el cual es un gran limitante, sin embargo el cliente confió en que no tendría demasiados problemas en utilizar una plataforma totalmente diferente, honestamente sí los he tenido pero el soporte del gran equipo que tienen en USA me ha apoyado muchísimo y he logrado aprender lo necesario y seguir avanzando en el dominio de este.

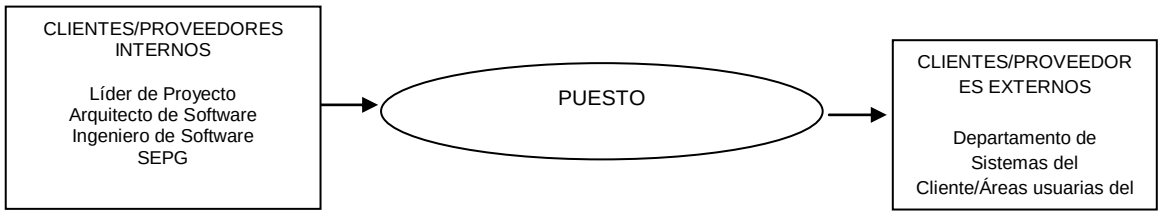
DESCRIPCIÓN UNIVERSAL DE PUESTO

Título del puesto: Ingeniero de Software Chief
Función/departamento: OPERACIONES
Centro de Costos: -
Título del puesto al que reporta: Líder de Proyecto

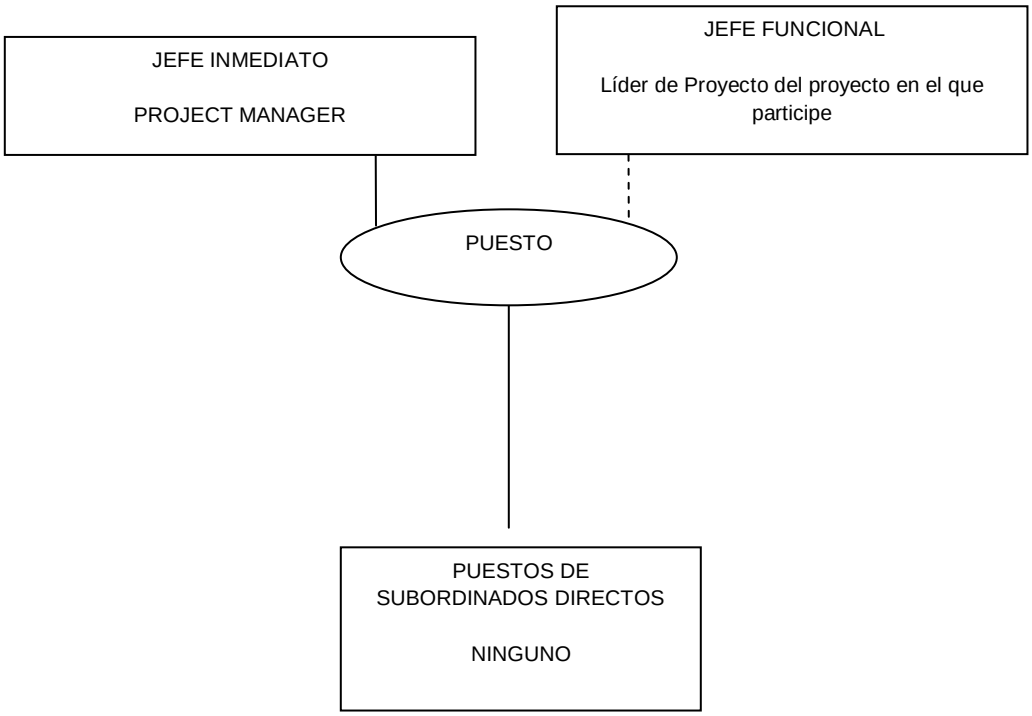
MISIÓN DEL PUESTO	
Analizar, diseñar, desarrollar, probar y liberar proyectos de desarrollo de software a la medida, mediante el seguimiento de los procesos y estándares establecidos en el QSSDP, el uso de herramientas de apoyo y lenguajes de programación, con la finalidad de terminar el proyecto dentro de la planeación programada y con la calidad establecida.	
ALCANCE	
Nivel: Número de Reportes Directos: Ninguno Número de Reportes Indirectos: Ninguno	
RESPONSABILIDADES PRINCIPALES	**INDICADORES DE MEDICION**
A. Seguir y cumplir el plan de trabajo individual dentro del proyecto, mediante el seguimiento de los procesos definidos en el QSSDP (Procesos de desarrollo internos), con la finalidad de concluir los proyectos en el tiempo planeado.	• **De Forma:** Valor ganado • **De Resultado**: Desviación del calendario
B. Cumplir con los objetivos de calidad definidos por el equipo para los componentes de software y documentos desarrollados en los proyectos, a través de revisiones e inspecciones, para disminuir el costo de mantenimiento de los sistemas.	• **De forma**: Plan de calidad **De Resultado**: Densidad Defectos Final
C. Ejecutar los roles establecidos para los proyectos de acuerdo al TSP (Team Software Process), mediante el seguimiento de los procesos definidos en el QSSDP (Procesos de desarrollo internos), para garantizar la coordinación del proyecto en todas sus fases.	• No. Menciones en auditorias

D. Recolectar información histórica y vaciar en repositorio de datos organizacional, mediante el registro de tiempos, tamaños y defectos de los productos o componentes de los proyectos, para monitoreo de proyectos en curso y para obtener mejores estimaciones para proyectos futuros.	• No. Menciones en auditorias
E. Ejecutar los procesos de estimación de tamaños (no. de líneas de código) de los proyectos, mediante la utilización de datos históricos y el seguimiento de los procesos definidos en el QSSDP (Procesos de desarrollo internos), a fin de generar la cotización del proyecto correspondiente.	• Desviación del min y max vs comportamiento histórico QS • Error de estimación individual vs final de equipo

RELACIONES – RED DE INTERACCION



ORGANIZACIÓN



PERFIL REQUERIDO

- Educación: Pasante de Ingeniería en Sistemas, Licenciatura en Sistemas, Licenciatura en Informática o carrera afín

Ingeniero de software Chief.:

- Experiencia laboral: de 3 a 5 años como desarrollador en desarrollo de software.
- Idiomas: Inglés (Lectura 100%, Escritura 80%, Hablado: 80%)
- Conocimientos Específicos (incluye habilidades específicas, paquetería, sistemas o herramientas especializadas):

Conocimientos sólidos de programación orientada a objetos en JAVA o .NET

Conocimientos de diversas Bases de Datos, My SQL, SQL, Oracle, Informix.

Conocimiento de Hibernate. Frameworks como STRUTS, SPRING, JSF Tecnologías de reporteo como Jasper o Cristal Reports o Microsoft reporting services. Diseño y modelado, UML, RUP, Rational Rose.

Conocimientos del negocio: Capacitación PSP, TSP, QSSDP

Agrega valor con la programación que realiza

Conoce de Patrones de Diseño

Sabe programar en 3 capas de manera estructurada (Flex, Ajax, Struts, JB, Hibernate, etc.)

Conocimientos en SPL

Capítulo 4. WorkTicket

4.1 Antecedentes

Como relaté en el capítulo anterior, entré a esta empresa (QuarkSoft) para trabajar con un cliente ubicado en Santa Clara, California, USA llamada LiveOps, el segudo callcenter más grande de ese país para desarrollar en Perl, lenguaje en el cual tengo amplia experiencia, estuve en ese proyecto durante 1 año, tiempo en el cual se cumplieron ciertos objetivos.

El proyecto terminó y el equipo se vio de pronto en el problema de no tener más trabajo.

El dueño de la empresa QuarkSoft en USA había conseguido un posible cliente, sin embargo el lenguaje de programación que necesitaban así como las herramientas eran completamente distintas a lo que yo había manejado hasta entonces, Stuart (El dueño de QuarkSoft LLC, nos proyectó como equipo con Daktronics y nos dejó la responsabilidad de, si lográbamos convencer al cliente de nuestra capacidad no importando el hecho de no haber usado antes los lenguajes, firmaría, de lo contrario, no sólo no firmaría sino que nos quedaríamos sin proyecto y por ende, sin empleo. Así pusimos manos a la obra, Stuart ya había conseguido entrevistas con la jefa del área de Tecnologías de la Información de Daktronics, y el equipo completo negoció con el cliente dar resultados y que las herramientas y el lenguaje usado no serían problema.

Después de evaluar la experiencia, los conocimientos, la habilidad para comunicarse en inglés, el interés, el entusiasmo y el compromiso individual y colectivo, el cliente firmó el contrato y comencé a estudiar todas las herramientas descritas antes. Cambiar de Linux a Windows fue el primer paso, el cliente nos regaló una suscripción para cursos en línea de PluralSigth (www.pluralsight-training.net/) y nos dio un par de semanas para aprender lo que necesitábamos aprender.

Obviamente no fue suficiente, pero era mucho mejor que nada, el cliente me trató y me ha tratado hasta hoy como empleado suyo. Así me llevó junto con mis dos compañeros de equipo a Dakota del Sur a conocer la empresa, conocer a nuestros colegas y nuestro trabajo. Puedo decir que es una empresa impresionante. Pasé a conocer todas las áreas y toda la planta. Se fabrican desde los soportes tubulares de los tableros hasta las líneas de leds que los componen. Los laboratorios de calidad son extraordinarios, se hacen pruebas de humedad, de calor, de paso de tiempo, de resistencia, con cámaras especiales para cada prueba unas incluso del tamaño de un pequeño auditorio. Un proceso de operaciones impresionante, esto claro está fuera del alcance de este trabajo, pero valía la pena mencionarlo.

4.2 Metodología de Desarrollo

Una de las características más importantes es la planeación que tiene esta empresa y sus prácticas, la jefa de TI nos ha reiterado y nos lo dijo antes de entrar que su prioridad es tener elementos que tengan la actitud (aparte de la aptitud) para trabajar en equipo. Y esta integración se ve y se refuerza tanto en

planeación como en las prácticas de desarrollo. El método que llevamos para el desarrollo es Scrum y las prácticas de desarrollo son algunas de XP (eXtreme Programming)o Programación extrema.

4.2.1 Scrum

"Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el ScrumMaster, que mantiene los procesos y trabaja de forma similar al director de proyecto, el ProductOwner, que representa a los stakeholders (interesados externos o internos), y el Team que incluye a los desarrolladores."

El Scrum es una serie de prácticas para llevar a cabo el proceso de planeación y seguimiento del desarrollo de software. Es una metodología ágil, y la llevamos a cabo en la empresa al pie de la letra.

"Durante cada sprint, un periodo entre una y cuatro semanas (la magnitud es definida por el equipo), el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint.2 Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint."

Nuestros sprints son generalmente de 2 semanas y se planea una serie de historias, tomadas de nuestro backlog, este backlog puede llenarse con historias tanto del equipo como del product owner, cuando vemos algún detalle que se puede o debe cambiar para mejorar, y que está fuera del alcance de la itearción actual, creamos una historia en el backlog que posteriormente el equipo completo punteará y el product owner decidirá cuando es conveniente llevarla a cabo.

Scrum permite la creación de equipos autoorganizados impulsando la co-localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.

Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan, y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por lo tanto, Scrum adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes.

“Existen varias implementaciones de sistemas para gestionar el proceso de Scrum, que van desde notas amarillas "post-it" y pizarras hasta paquetes de software. Una de las mayores ventajas de Scrum es que es muy fácil de aprender, y requiere muy poco esfuerzo para comenzarse a utilizar.”

Roles Principales

Product Owner

“El Product Owner representa la voz del cliente. Se asegura de que el equipo Scrum trabaja de forma adecuada desde la perspectiva del negocio. El Product Owner escribe historias de usuario, las prioriza, y las coloca en el Product Backlog”

En nuestro caso nuestros product owners son business analists que están a cargo de diferentes proyectos y están entre el business y el área de IT. Priorizan las historias y escriben historias recibidas de las peticiones recabadas por el personal de negocios que son quienes son nuestros clientes dentro de Daktronics.

ScrumMaster (o Facilitador)

“El Scrum es facilitado por un ScrumMaster, cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El ScrumMaster no es el líder del equipo (porque ellos se auto-organizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El ScrumMaster se asegura de que el proceso Scrum se utiliza como es debido. El ScrumMaster es el que hace que las reglas se cumplan.”

En el caso exclusivo de Daktronics (que no es el de LiveOps por ejemplo) el Scrum Master es nuestro Product Owner y si tenemos algún bloqueo tiene la responsabilidad de auxiliarnos para resolverlo, es quien agiliza nuestras peticiones y conaliza las dudas con la persona indicada para resolverla, organiza las juntas, da los mensajes, concilia reparte si es necesario.

Equipo de desarrollo

El equipo tiene la responsabilidad de entregar el producto. Un pequeño equipo de 3 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc).

En el caso de particular de Daktronics, los equipos varian dependiendo de la demanda del proyecto en ese momento determinado, ya que existen muchos proyectos y cuando alguno tiene prioridad se le asignan más ingenieros, así pues nos encontramos ingenieros, un estudiante, un QA, y un business analist por cada proyecto.

Roles Auxiliares

Los roles auxiliares en los "equipos Scrum" son aquellos que no tienen un rol formal y no se involucran frecuentemente en el "proceso Scrum", sin embargo deben ser tomados en cuenta. Un aspecto importante de una aproximación ágil es la práctica de involucrar en el proceso a los usuarios, expertos del negocio y otros interesados (stakeholders). Es importante que esa gente participe y entregue retroalimentación con respecto a la salida del proceso a fin de revisar y planear cada sprint.

Stakeholders (Clientes, Proveedores, Vendedores, etc)

Se refiere a la gente que hace posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su producción. Sólo participan directamente durante las revisiones del sprint.

Administradores (Managers)

Es la gente que establece el ambiente para el desarrollo del producto.

Reuniones en Scrum

Daily Scrum

"Cada día de un sprint, se realiza la reunión sobre el estado de un proyecto. Esto se llama "daily standup". El scrum tiene unas guías específicas:

La reunión comienza puntualmente a su hora. A menudo hay castigos -acordados por el equipo- para quien llegue tarde (por ejemplo: dinero, flexiones, llevar colgando una gallina de plástico del cuello, etc)

Todos son bienvenidos, pero sólo los responsables pueden hablar.

La reunión tiene una duración fija de 15 minutos, de forma independiente del tamaño del equipo.

Todos los asistentes deben mantenerse de pie (esto ayuda a mantener la reunión corta)

La reunión debe ocurrir en la misma ubicación y a la misma hora todos los días.

Durante la reunión, cada miembro del equipo contesta a tres preguntas:3

¿Qué has hecho desde ayer?

¿Qué es lo que estás planeando hacer hoy?

¿Has tenido algún problema que te haya impedido alcanzar tu objetivo? (Es el papel del ScrumMaster recordar estos impedimentos)."

En Daktronics se tienen al inicio del día un standup general de todos los proyectos donde sólo uno de los integrantes de cada proyecto dice cómo va el equipo.

Por separado y a una hora que el equipo acuerda en conjunto, se realiza el standup del proyecto, este proceso en realidad tarda más de lo que debería tardar porque a veces se discuten algunas cosas de trabajo, sin embargo no debería suceder.

Retrospectiva del Sprint (Sprint Retrospective)

"Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua del proceso"

El tiempo para la retrospectiva no es necesariamente fijo, se revisa qué sucedió en el sprint que terminó, dividimos un pizarrón en tres columnas escribimos en la primera *"Stop Doing"* (dejar de hacer) en la segunda *"Start doing"*(comenzar a hacer) y en la última *"Keep doing"* (seguir haciendo) y se refiere a las prácticas que hemos estado llevando a cabo durante el sprint. Por ejemplo se me ocurre mencionar en la columan de stop doing " hacer dos juntas diarias".

Cada integrante puede poner tantas como se le ocurran y al final se agrupan en temas y se analizan.

Otra forma de hacer estas juntas es poner en las columnas, "Happy" (feliz), "Sad"(triste) y "Mad" (molesto), y se refieren a lo que sucedió en el sprint que te hizo sentir feliz, triste y molesto, en general es similar aunque tiene diferencias muy puntuales uno y otro método.

Planeación del Sprint

En las juntas de planeación se reune el equipo y de acuerdo a un criterio definido por la empresa se hace la puntuación del esfuerzo necesario a cada historia. En algunas empresas el criterio es por ejemplo, un punto es igual a un día de trabajo por ingeniero.

En otros casos es, un punto es una pagina html básica, 8 puntos es un script muy complejo. Es una medida muy subjetiva.

Para hacer el punteo utilizamos un juego muy popular en el Scrum llamado planning poker que son cartas marcadas cada una con un número que es el puntaje que le da cada integrante se da la historia y cada integrante "echa" su carta boca abajo, cuando todos han tirado su carta se voltean y si coinciden esta es el puntaje de esta tarea, cuando no se coincide con la puntuación el equipo discute y analiza un poco más la historia y vuelve a votar.

En el caso particular de nuestro equipo ya que somos personal remoto, utilizamos la herramienta mediante el software de un tercero que es gratuita y se encuentra disponible en internet llamada www.planningpoker.com

Después de puntear, se deciden cuantas historias caben el sprint, se analizan las historias y se van aceptando, el equipo no sólo acepta, sino se compromete a cumplir esas tareas, por eso es que debe uno ser cuidadoso de las historias a las que se compromete.

Enseguida se asignan tareas a cada historia, que es el trabajo detallado de lo que se realizará, a estas se les asignan horas.

Al final de esto, se tiene un total de horas en el sprint y el control del trabajo se visualiza mediante una gráfica "burn down". La figura 4.1 muestra esta gráfica

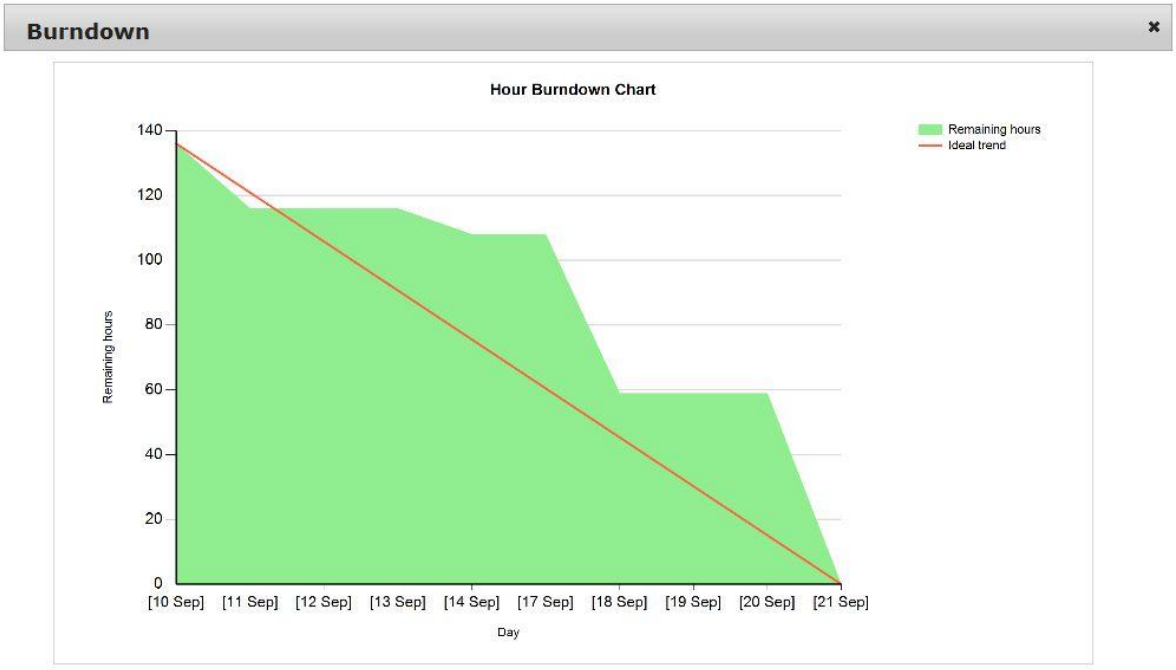


Figura 4.1. Gráfica “burn down”.

4.2.2 Prácticas de Desarrollo

Se llevan en la empresa ciertas prácticas de desarrollo, algunas de ellas llamadas XP(eXtreme Programming), estas prácticas no las he visto en ninguna otra empresa incluida en la que estoy contratado, estas prácticas que llevamos con ellos son:

Pair Programming

Este como su nombre lo indica es programación en pares, esta práctica consiste en que la programación se hace entre dos ingenieros, mientras uno escribe y tiene la idea general de lo que se está programando el otro integrante revisa lo que se está escribiendo para verificar errores de concepto o sintaxis, y al mismo tiempo tiene una visión más amplia porque no está concentrado en lo que se está escribiendo en ese momento, así pues puede hacer sugerencias, dictar el código, sugerir un método mejor. Y así cambian el rol del que escribe y del que sugiere.

Es una práctica en la que parecíera que se pierde un elemento del equipo para hacer otras tareas, que fue lo que pensé yo al inicio, sin embargo es una práctica donde se ganan varias cosas, entre ellas intercambio de conocimientos, agilidad para escribir, enfrentar los problemas desde dos perspectivas diferentes, realizar más rápido el código y con menos errores.

Unit Testing

Es un metodo por el cual se aplican pruebas a unidades individuales de código, se escriben pequeñas pruebas, a veces no por pequeñas quiero decir sencillas, a metodos publicos del código con datos enviados desde nuestras unidades de test.

Contacto estrecho con el área de business

Tenemos contacto directo con el área de business para dudas que tengamos o para que nos apoyen con sugerencias de pruebas, a veces incluso informalmente nos pueden apoyar para hacer pequeñas pruebas a lo realizado.

Book readings

Recientemente empezamos a leer un libro y hacer revisiones el equipo completo de IT acerca de un libro llamado *"Agile Testing"* lo cual aparte de integrar el equipo, nos ayuda a tener mejores prácticas y sugerencias en nuestro trabajo.

4.3 Herramientas, lenguajes y frameworks

Para este proyecto WorkTicket se han utilizado una serie de herramientas y lenguajes que anteriormente no había utilizado y varios que ya conocía con anterioridad.

4.3.1 Visual Studio

Esta herramienta es in IDE (Integrated Development Environment) o entorno de desarrollo integrado, para windows que soporta varios lenguajes entre ellos C# que es el utilizado en su mayoría en este proyecto, contiene un sinfín de plantillas y una de als mejores características que he visto en un IDE el intelisense, excelente hace mucho más efectiva la programación, no sólo completa palabras y namespaces, sino métodos nuevos, objetos, es muho muy efectivo.

4.3.2 TFS

Este producto también de Microsoft ofrece para el proyecto el control de código (Source control), data collection, reportes o informes, y administración de proyectos. Este está integrado al Microsoft Visual Studio y a pesar de no ser tan intuitivo es una poderosa herramienta.En lo particular nosotros utilizamos el TFS enfocado a Scrum que tiene una amplia gama de herramientas para la planeación, agregar historias, sprints, releases, backlog, tareas, estatus de cada tarea e historia, etc, diagrama Burndown.

Ofrece también una interfaz web muy cómoda para la planeación y seguimiento del sprint que para mí es más cómoda y muy intuitiva.

En la figura 4.2 se muestra la interfaz del TFS, las historias y las tareas que se han hecho y las que quedan por hacer.

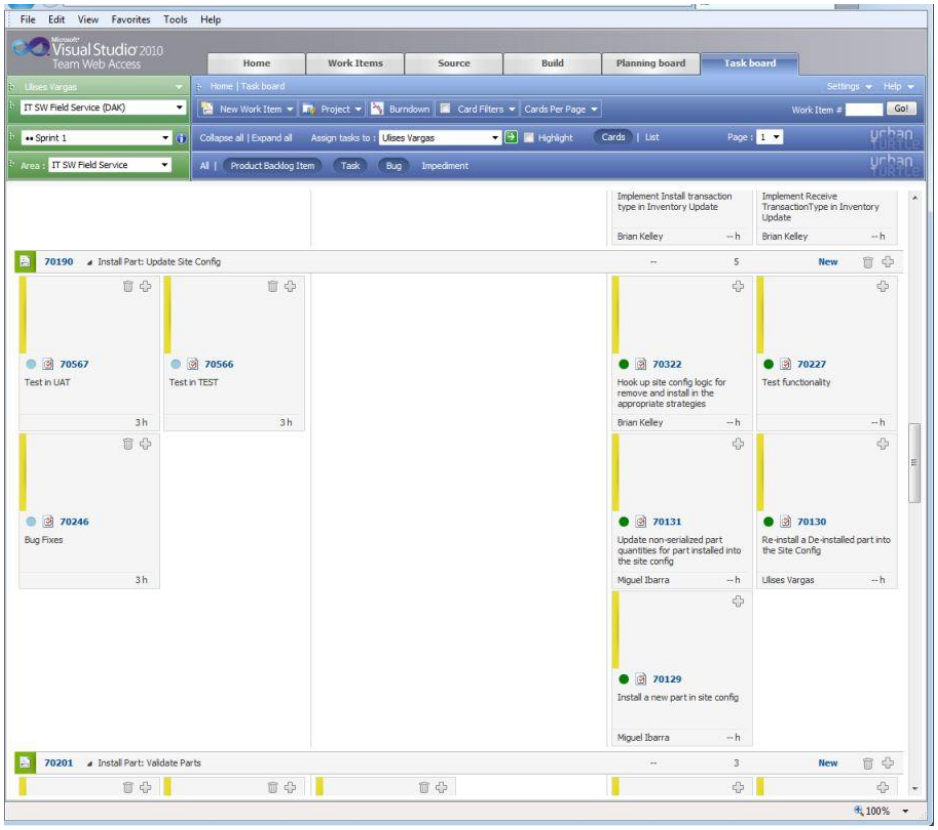


Figura 4.2 TFS Scrum

4.3.3 Knockout

Es una biblioteca diseñada para aplicar cómodamente el patrón MVVM en aplicaciones javascript.

Knockout nos ofrece una manera muy sencilla de enlazar View y ViewModel a través de data binding. Este data binding se hace de forma declarativa usando atributos data-bind.

Las posibilidades que ofrece Knockout para realizar el *data binding* son de lo más variado.

Para que funcione esto del data binding es fundamental poder detectar los cambios en los valores de los controles para propagarlos al ViewModel, pero también es necesario poder detectar los cambios en el ViewModel para propagarlos a la vista. En este aspecto Knockout cuenta con su propia implementación del patrón observer, a través de funciones como ko.observable.

En la figura 4.3 Observamos un pedazo de código extraído de su página de internet.

Figura 4.3 Knockout

4.3.4 Jasmine

Las pruebas unitarias son una de las técnicas de ingeniería del software que mayor aceptación están logrando entre los desarrolladores en los últimos tiempos. Escribir buenas pruebas unitarias es labor casi imposible en la realidad sin apoyarse en un framework que permita aislar nuestras pruebas de dependencias externas.

Jasmine es un framework de pruebas unitarias para JavaScript.

Jasmine es un marco de trabajo de BDD para probar código Javascript

- No depende de ningún otro marco de trabajo
- No requiere un DOM
- Nos permite realizar pruebas de manera limpia y obvia

Las Suites describen tus pruebas

Una suite de pruebas comienza por la función describe con dos parámetros, una cadena (string) y una función.

La cadena es el nombre de la suite, y la función es el código que la implementa.

Specs

Las especificaciones o "Specs" se definen con la función it, muy parecida a describe utiliza una cadena y la función del spec (o prueba). Un spec puede tener una o mas expectativas que prueban el estado del código dentro de la prueba.

Una expectativa en Jasmine es una aseveración de que algo es verdadero o falso. Un spec con todas sus expectativas verdaderas significan que el spec es satisfactorio. Un spec con una o mas expectativas que evalúen a falso, es un spec que fallo.

Ejemplo:

```
describe("A suite", function() {
        it("contains spec with an expectation", function() {
          expect(true).toBe(true);
        }
}
```

4.3.5 JS

En el proyecto se utiliza muchísimo, toda la interfaz tiene muchísimos elementos y validaciones en JavaScript.

Toda la interfaz tiene elementos de JavaScript, ya que se utilia Knockout, no podría ser de otra forma, su uso en este proyecto es indispensable.

4.3.6 C#

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Todo el backend del proyecto está escrito en C#, que es el lenguaje que utilizamos. Este lenguaje es extremadamente completo, ha sido una grata experiencia conocerlo y utilizarlo.

4.3.7 SQL Server Management Studio

Es una herramienta indispensable en nuestro entorno de desarrollo, utilizamos SQL Server y para poder crear Stored Procedures, visualizarlos, modificarlos, probar, manipular nuestras tablas y revisarlas, lo necesitamos, no podríamos hacerlo de manera menos visual y sencilla si no fuera gracias al SQL Server Management Studio.

Este tiene integrada una herramienta para hacer traces de la base de datos llamado SQL Server Profiler, que nos ha sido muy útil en este proyecto. Y que nos ha permitido revisar paso a paso lo que hace la base de datos al procesar el antiguo sistema la información.

4.3.8 JQuery

Se utiliza en varios puntos del código y básicamente para los popups que utilizamos en el proyecto, y para algunas confirmaciones. Es extensivamente utilizado en este medio, es una librería ya indispensable para cualquier programador.

4.4 WorkTicket

El WorkTicket es nuestro Proyecto por el cual dejamos de hacer mantenimiento, si bien hemos estado involucrados en varios proyectos y con varios compañeros, sin embargo el proyecto de WorkTicket fue uno nuevo al que fuimos asignados de principio los tres compañeros de México y de allí hemos participado en varios proyectos pero siempre regresamos al WorkTicket en determinado momento.

4.4.1 Antecedentes

A grandes razgos cuando se coloca un Display en una ubicación nueva (como una empresa, un estadio, una universidad, un gimnasio), en la base de datos se guarda toda la información relacionada con el usuario, con el sitio, con el técnico que lo colocó, etc. Pero también se guarda toda la información de las piezas que lleva el display.

Para ello se necesita cierta información y se organiza de cierta manera.

Todo el paquete se coloca en una “bandeja” que generalmente es el nombre del sitio donde se colocó el display, dentro de esta bandeja se colocan las piezas y hay piezas que tienen piezas hijos, y así sucesivamente. Todas estas piezas quedan con el status de “installed” cuando están en el display con el cliente.

Cuando el display falla, se levanta un ticket de trabajo (WorkTicket), esta se asigna a un técnico y se le da fecha para ir a hacer el trabajo.

El técnico cuenta en su vehículo una serie de piezas de repuesto (inventario del Técnico). Y cuando termina su trabajo reporta lo que hizo, este será el trabajo del WorkTicket, procesar todo lo realizado por el técnico.

Anteriormente se hacía mediante un sistema en el que cada pieza cambiada, regresada, intercambiada, utilizada por cada técnico se procesaba independientemente en el sistema se abre el sitio, después despliega la pila de piezas instaladas, se verifica la información tomada telefónicamente y guardada en una hoja de pdf de parte del técnico y cada pieza se comienza a desinstalar, reinstalar, cambiar, regresar, consumir del inventario, etc, dependiendo de lo que se les hayan hecho.

El proyecto principal comienza por una pequeña forma para recibir de parte del técnico lo que se le ha hecho a cada pieza, honestamente nadie tenía idea de lo grande de este proyecto, hasta que se comenzó, pues empezamos 3 ingenieros y un business analist, hasta que se llegaron a tener 6 ingenieros y 4 personas de business en el desarrollo.

4.4.2 Desarrollo

Lo primero fue definir al arquitectura que utilizaríamos, decidimos que sería MVC 3 (Model View Controller) o Modelo Vista Controlador, debido al tipo de sistema que ibamos a implementar nos quedaba bien separar el backendo del frontend por medio de este patrón.

Así que pusimos manos a la obra y la primer historia en nuestro backlog fue mostrar el ticket abierto que tenía le pertenecía a cada ingeniero de soporte, para eso, el área de business no tenía muy bien la idea de cómo es que quería el formato, así que le pedimos al área de business que elabarorara un mockup para eso.

Nos proporcionaron uno y con base en él trabajamos, necesitabamos algo para identificar el usuario y que casos tenía pendientes, un caso qué atender contiene varios subcasos, así que el subcaso fue, digámoslo así, nuestro dato de batalla. Junto con el id del usuario.

Para la base de datos comenzamos a trabajarla con la ayuda de un DBA, el comenzó a realizarla conforme a lo que business le daba, y entre las tres áreas comenzamos a ponernos de acuerdo cómo iba a funcionar todo.

Las tablas fueron construidas dentro de la base de datos que ya existía para este sistema, los datos los sacamos también de tablas ya existentes en diferentes bases de datos.

La primer pantalla como podemos verla en la figura siguiente, este es el subcase summary (Figura 4.4)

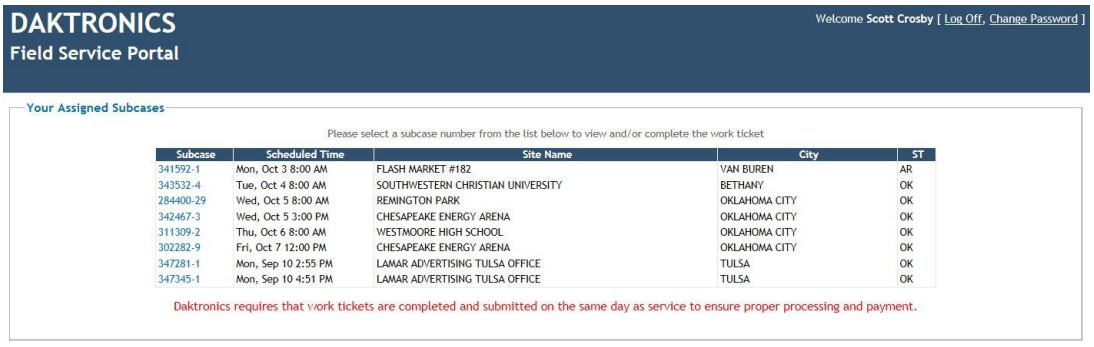


Figura 4.4 Pantalla de casos asignados

Observamos que tiene muy pocos elementos, aquí el código de la vista

```
@model Daktronics.IT.FieldService.SubcaseSummaryModel

@using Daktronics.IT.FieldService.BusinessLogic;

@{
   ViewBag.Title = "Choose a Work Ticket";
   Layout = "~/Views/Shared/_Layout.cshtml";
}

<fieldset>
   <legend>Your Assigned Subcases</legend>

   @if (Model.Subcases.Count != 0)
   {
      <br />
      <div style="margin: 0 auto; width: 50%; text-
align: center">Please select a subcase number from the list below to view and/or complete the work ticket</div>

   }
   <table class="standardTable" style="width: 75%; margin: 10px auto;">
      <tr>
         <th style="text-align: center; width: 10%">Subcase</th>
         <th style="text-align: center; width: 18%">Scheduled Time</th>
         <th style="text-align: center; width: 44%">Site Name</th>
         <th style="text-align: center; width: 23%">City</th>
         <th style="text-align: center; width:  5%">ST</th>
      </tr>

   @if (Model.Subcases.Count == 0)
   {
      <tr><td colspan="6" style="text-align:center">No Dispatches to Show</td></tr>
   }

   @foreach (SubcaseSummary subcase in Model.SubcasesSortedByScheduledOnSite)
   {
      <tr style="line-height: 1.5em;">
         <td class="display-
field"><a href="@Url.Action("Index", "ViewSubcase", new { SubCaseObjId = subcase.SubCaseObjId })" target="_bla
nk">@subcase.SubcaseId</a></td>
         <td class="display-field">@subcase.ScheduledDateTimeShort</td>
         <td class="display-field">@subcase.SiteName</td>
         <td class="display-field">@subcase.City</td>
         <td class="display-field">@subcase.State</td>
      </tr>
   }
   </table>
   <div class="ImportantMessage">Daktronics requires that work tickets are completed and submitted on the same
 day as service to ensure proper processing and payment.</div>

</fieldset>
```

Observamos que no tiene elementos de knockout, ni de JS, vemos los elementos de Razor y html que nos dan la estructura de la página, la primera línea nos da el modelo que se usara en esta vista de donde procederán los datos:

```
@model Daktronics.IT.FieldService.SubcaseSummaryModel
```

Buscamos el modelo y tenemos esto:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Daktronics.IT.FieldService.BusinessLogic;

namespace Daktronics.IT.FieldService
{
  public class SubcaseSummaryModel
  {
    public SubcaseSummaryModel()
    {

    }

    public List<SubcaseSummary> Subcases { get; set; }

    public IEnumerable<SubcaseSummary> SubcasesSortedByScheduledOnSite
    {
      get
      {
        this.Subcases.Sort(SubcaseSummary.CompareByScheduledOnSite);
        return Subcases;
      }
    }
  }
}
```

Lo único que hace nuestro modelo es ordenar los datos por fecha. Y los pasa a la vista.

Veamos el controlador ahora

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Daktronics.IT.FieldService.BusinessLogic;
using System.Web.SessionState;
using Daktronics.IT.Common.Tools;

namespace Daktronics.IT.FieldService.Controllers
{
    [SessionState(SessionStateBehavior.ReadOnly)]
    public class SubcaseSummaryController : Controller
```

```
    {
        public ActionResult Index()
        {

     if (SafeConvert.ToBool(HttpContext.Session["LoggedInWithDefaultPassword"]) == true)
                                    {
                                                return RedirectToAction("ChangePassword", "Account");
                                    }

                                    var model = new SubcaseSummaryModel();

     model.Subcases = SubcaseSummaryRepository.GetSubcaseSummary(Session.CurrentUser().EmployeeObjId);

                                    Response.Cache.SetCacheability(HttpCacheability.NoCache);
                                    return View(model);
        }
    }
}
```

Esta es una versión actual, en la cual ya tenemos validación de usuario, aquí vemos que le asigna al modelo.Subcases lo que regresa la clase

```
SubcaseSummaryRepository.GetSubcaseSummary(Session.CurrentUser().EmployeeObjId);
```

Y luego le pasa el resultado a la vista

Veamos que hace este método GetSubcaseSummary.

```
namespace Daktronics.IT.FieldService.BusinessLogic
{
  public class SubcaseSummaryRepository
  {
    public static List<SubcaseSummary> GetSubcaseSummary(int EmployeeObjId)
    {
      var provider = DependencyResolver.Current.GetService<SubcaseSummaryDataProvider>();
      return SubcaseSummaryBuilder.Build(provider.GetSubcaseSummary(EmployeeObjId));
    }
  }
}
```

Este se va a un dependency resolver que dependiendo que data provider sea nos envía el método correcto, ya sea un mock data o datos reales.

En este caso veremos a donde vamos por los datos que se necesitan

```
namespace Daktronics.IT.FieldService.BusinessLogic
{
 public class WebServiceSubcaseSummaryDataProvider : SubcaseSummaryDataProvider
 {
  public string GetSubcaseSummary(int EmployeeObjId)
  {
   return WebService.CallGet(WebService.Method.WorkTicketSubcaseSummary, EmployeeObjId.ToString());
  }
 }
}
```

Este envía nuestro número de empleado a una llamada GET de un webservice.

Esta clase WebService es muy grande pues maneja todas las llamadas a nuestro proyecto de Webservices que está en otro servidor y este le asigna la dirección a la que debe ir.

Cabe destacar que utilizamos REST para el diseño de este servidor de Webservices

Vamos a poner la clase completa solo para ilustrar y ver este y las demás direcciones de los webservices:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Configuration;
using System.Net;
using System.Web.Mvc;

namespace Daktronics.IT.FieldService.BusinessLogic
{
 public static class WebService
 {

  public static ContentResult Get(Method method, params string[] args)
  {
   var jsonData = CallGet(method, args);
   return new ContentResult()
   {
    Content = jsonData,
    ContentType = "application/json"
   };
  }

  public static ContentResult Post(Method method, string data, params string[] urlParameters)
  {
   string jsonData = CallPost(method, data, urlParameters);
   return new ContentResult()
   {
    Content = jsonData,
    ContentType = "application/json"
```

```
  };
}

public static ContentResult Delete(Method method, string data)
{
  string jsonData = CallDelete(method, data);
  return new ContentResult()
  {
    Content = jsonData,
    ContentType = "application/json"
  };

  throw new NotImplementedException();
}

public static string CallGet(Method method, params string[] args)
{
  try
  {
    return RequestHelper.SendGETRequest(BuildURL(method, args));
  }
  catch (WebException)
  {
    return null;
  }

}

public static string CallPost(Method method, string data, params string[] urlParameters)
{
  try
  {
    return RequestHelper.SendPostRequest(BuildURL(method, urlParameters), data);
  }
  catch (WebException e)
  {
    using (System.IO.StreamReader reader = new System.IO.StreamReader(e.Response.GetResponseStream()))
    {
      string thisIsWhatWeWant = reader.ReadToEnd();
    }
    return null;
  }
}

public static ResponseType CallPost<ResponseType>(Method method, object data)
{
  string[] emptyArray = new string[0];
  string jsonData = JavascriptSerializer.Serialize<object>(data);
  string resultjsonString = WebService.CallPost(WebService.Method.WorkTicketUpdatePassword, jsonData, emptyArray);

  return JavascriptSerializer.Deserialize<ResponseType>(resultjsonString);
}

public static string CallDelete(Method method, string data)
{
  try
  {
    return RequestHelper.SendDeleteRequest(BuildURL(method), data);
  }
  catch (WebException)
  {
    return null;
  }
}

private static string BuildURL(Method method)
{
  return BuildURL(method, new string[0]);
}

private static string BuildURL(Method method, string[] args)
{
  string baseURL;

  switch(method)
  {
    case Method.WorkTicketSubCase: baseURL = ServiceUrlDefines.WorkTicketSubCase; break;
```

```
      case Method.WorkTicketPartRequest: baseURL = ServiceUrlDefines.WorkTicketPartRequest; break;
      case Method.WorkTicketDispatch: baseURL = ServiceUrlDefines.WorkTicketDispatch; break;
      case Method.WorkTicketSubcaseSummary: baseURL = ServiceUrlDefines.WorkTicketSubcaseSummary; break;
      case Method.WorkTicketAuthenticateUser: baseURL = ServiceUrlDefines.WorkTicketAuthenticateUser; break;
      case Method.WorkTicketPartUses: baseURL = ServiceUrlDefines.WorkTicketPartUses; break;
      case Method.WorkTicketPartsSearch: baseURL = ServiceUrlDefines.WorkTicketPartsSearch; break;
      case Method.WorkTicketTimestamps: baseURL = ServiceUrlDefines.WorkTicketTimestamps; break;
      case Method.WorkTicketTimestampsValidation: baseURL = ServiceUrlDefines.WorkTicketTimestampsValidation; break;
      case Method.WorkTicketPart: baseURL = ServiceUrlDefines.WorkTicketPart; break;
      case Method.WorkTicketOtherParts: baseURL = ServiceUrlDefines.WorkTicketOtherParts; break;
      case Method.WorkTicketTrunkInventory: baseURL = ServiceUrlDefines.WorkTicketTrunkInventory; break;
      case Method.WorkTicketOtherPart: baseURL = ServiceUrlDefines.WorkTicketOtherPart; break;
      case Method.WorkTicketReturningParts: baseURL = ServiceUrlDefines.WorkTicketReturningParts; break;
      case Method.WorkTicketReturningPart: baseURL = ServiceUrlDefines.WorkTicketReturningPart; break;
      case Method.WorkTicketSummary: baseURL = ServiceUrlDefines.WorkTicketSummary; break;
      case Method.WorkTicketSubmit: baseURL = ServiceUrlDefines.WorkTicketSubmit; break;
      case Method.WorkTicketNotes: baseURL = ServiceUrlDefines.WorkTicketNotes; break;
      case Method.WorkTicketStatus: baseURL = ServiceUrlDefines.WorkTicketStatus; break;
      case Method.WorkTicketUpdatePassword: baseURL = ServiceUrlDefines.WorkTicketUpdatePassword; break;
      case Method.SendEmailMessage: baseURL = ServiceUrlDefines.SendEmailMessage; break;

case Method.WorkTicketShouldWarnUnsoundReturningPart: baseURL = ServiceUrlDefines.WorkTicketShouldWarnUnsoundReturningPart;
break;



  default: throw new ArgumentOutOfRangeException("method", "Unknown Web Service Method");
    }


    string argsURL = String.Join("/", args);

    if (argsURL != String.Empty)
    {
      return baseURL + "/" + argsURL;
    }

    return baseURL;
  }

  public static class ServiceUrlDefines
  {
    // XXX TODO check if SOASericeBaseUrl has or not a trailing '/'
    static string baseUrl = ConfigurationManager.AppSettings["SOAServiceBaseUrl"];

    public static string WorkTicketSubCase             = baseUrl + "WorkTicket/SubCase";
    public static string WorkTicketDispatch            = baseUrl + "WorkTicket/Dispatch";
    public static string WorkTicketSubcaseSummary      = baseUrl + "WorkTicket/SubcaseSummary";
    public static string WorkTicketAuthenticateUser    = baseUrl + "WorkTicket/AuthenticateUser";
    public static string WorkTicketPartUses            = baseUrl + "WorkTicket/PartUses";
    public static string WorkTicketTimestamps          = baseUrl + "WorkTicket/Timestamps";
    public static string WorkTicketPart                = baseUrl + "WorkTicket/Part";
    public static string WorkTicketOtherParts          = baseUrl + "WorkTicket/OtherParts";
    public static string WorkTicketOtherPart           = baseUrl + "WorkTicket/OtherPart";
    public static string WorkTicketReturningParts      = baseUrl + "WorkTicket/ReturningParts";
    public static string WorkTicketReturningPart       = baseUrl + "WorkTicket/ReturningPart";
    public static string WorkTicketPartsSearch         = baseUrl + "Parts/Description";
    public static string WorkTicketPartRequest         = baseUrl + "Parts/Subcase";
    public static string WorkTicketTrunkInventory      = baseUrl + "Parts/TrunkInventory";
    public static string WorkTicketTimestampsValidation = baseUrl + "WorkTicket/Timestamps/Validation";
    public static string WorkTicketSummary       = baseUrl + "WorkTicket/Summary";
    public static string WorkTicketSubmit        = baseUrl + "WorkTicket/Submit";
    public static string WorkTicketNotes         = baseUrl + "WorkTicket/Notes";
    public static string WorkTicketStatus              = baseUrl + "WorkTicket/Status";
    public static string WorkTicketUpdatePassword      = baseUrl + "WorkTicket/UpdatePassword";
    public static string SendEmailMessage        = baseUrl + "Email/Send";
    public static string WorkTicketShouldWarnUnsoundReturningPart = baseUrl + "WorkTicket/ShouldWarnUnsoundReturningPart";
  }

  public enum Method
  {
    WorkTicketSubCase,
    WorkTicketPartRequest,
    WorkTicketDispatch,
    WorkTicketSubcaseSummary,
    WorkTicketPartById,
    WorkTicketAuthenticateUser,
```

```
    WorkTicketPartUses,
    WorkTicketPartsSearch,
    WorkTicketTimestamps,
    WorkTicketPart,
    WorkTicketOtherParts,
    WorkTicketTrunkInventory,
    WorkTicketOtherPart,
    WorkTicketReturningParts,
    WorkTicketTimestampsValidation,
    WorkTicketReturningPart,
    WorkTicketSummary,
    WorkTicketSubmit,
    WorkTicketNotes,
    WorkTicketStatus,
    WorkTicketUpdatePassword,
    SendEmailMessage,
    WorkTicketShouldWarnUnsoundReturningPart,
  }
 }
}
```

Básicamente lo que hace esta clase es decirle a qué dirección debe ir para nuestro caso vamos a ir a:

```
public static string WorkTicketSubcaseSummary      = baseUrl + "WorkTicket/SubcaseSummary";
```

Veamos ahora en nuestro servidor lo que tuvimos que construir para traer los datos:

Tenemos nuestros webservices, ya no pondré toda la clase porque es mucho código, más que la clase anterior, así que solo pondré la sección de código que nos incumbe para este servicio:

```
[WebGet(UriTemplate = "SubcaseSummary/{EmployeeObjId}")]
  [MultiRoledPrincipalPermission(SecurityAction.Demand, MethodName = "GetSubcaseSummary")]
  public List<SubcaseSummary> GetSubcaseSummary(string EmployeeObjId)
  {
   try
   {
    return WorkTicketRepository.GetSubcaseSummary(EmployeeObjId);
   }
   catch (ArgumentOutOfRangeException)
   {
    WebOperationContext.Current.OutgoingResponse.SetStatusAsNotFound();
    return null;
   }
  }
```

Aquí observamos que es un webGet y estamos creando la dirección SubcaseSummary y el valor {EmployeeObjId} este valor lo mandaremos a:

```
WorkTicketRepository.GetSubcaseSummary(EmployeeObjId);
```

Que también está en un repositorio que nos enviará datos de prueba o datos reales, el cual no lo vamos a revisar, nos iremos directamente a los datos reales, antes comentaremos que lo que nos est'a

regresando es lo que viene directamente de esta función que es una Lista de SubcaseSummary este DTO contiene:

```
namespace Daktronics.IT.SOA.ServiceLogic
{
 public class SubcaseSummary
 {
  public int     SubCaseObjId;
  public string  SubcaseId;
  public int     DispatchObjId;
  public string  SiteName;
  public string  City;
  public string  State;
  public DateTime ScheduledOnSite;
 }
}
```

Bien entonces vamos al método que nos regresa esta Lista de SubcaseSummary

```
namespace Daktronics.IT.SOA.ServiceLogic
{
 public class SubcaseSummaryDataProvider : ISubcaseSummaryDataProvider
 {
  public List<SubcaseSummary> GetSubcaseSummary(string EmployeeObjIdAsString)
  {
   int EmployeeObjId = SafeConvert.ToInt32(EmployeeObjIdAsString);
   Database database = DatabaseFactory.CreateDatabase(DatabaseDefines.AmdocsDatabase);

   using (DbCommand command = database.GetStoredProcCommand(DatabaseDefines.GetSubcaseSummary))
   {
    database.AddInParameter(command, "@EmployeeObjId", DbType.String, EmployeeObjId);
    DataTable data = database.ExecuteDataSet(command).Tables[0];

    return ConvertDataTableToSubcaseSummaryList(data);
   }
  }

  private List<SubcaseSummary> ConvertDataTableToSubcaseSummaryList(DataTable data)
  {
   var result = new List<SubcaseSummary>();

   foreach (DataRow dr in data.Rows)
   {
    result.Add(new SubcaseSummary() {
     #region Filling All Fields & Convertions
     SubCaseObjId    = SafeConvert.ToInt32(dr["SubcaseObjId"]),
     SubcaseId       = SafeConvert.ToString(dr["SubcaseId"]),
     DispatchObjId   = SafeConvert.ToInt32(dr["DispatchObjId"]),
     SiteName        = SafeConvert.ToString(dr["SiteName"]),
     City            = SafeConvert.ToString(dr["city"]),
     State           = SafeConvert.ToString(dr["state"]),
     ScheduledOnSite  = SafeConvert.ToDateTime(dr["x_gsa_arrive_onsite"])
     #endregion
    });
   }

   return result;
  }
 }
}
```

Aquí podemos observar cómo recibe el numero de empleado y lo envía a un Stored Procedure que lo mostraremos, este regresa un DataTable del cual extraemos cada datarow y cada uno lo asignamos a un SubcaseSummary y así obtenemos la lista de SubcaseSummary y la enviamos de regreso.

Este es el Stored Procedure que utilizamos para acceder a la base de datos y traer las columnas que necesitamos, podemos ver que es un Stored Procedure muy pequeño y sin ninguna complejidad. Sin embargo tenemos Stored Procedures muy complejos en la aplicación.

```
ALTER PROCEDURE [AppServices].[usp_GetOpenDispatches_Sel]
 @Username VARCHAR(20) = NULL,
 @EmployeeObjId INT = NULL
AS
BEGIN

 SET NOCOUNT ON;

 SELECT sc.objid SubCaseObjId, sc.id_number SubcaseId, d.objid DispatchObjId, d.x_gsa_arrive_onsite, s.name SiteName, d.city,
d.state
 FROM table_subcase as sc
  INNER JOIN table_disptchfe as d
   ON sc.objid = d.disptchfe2subcase
  INNER JOIN table_employee as e
   ON d.disptchfe2employee = e.objid
  INNER JOIN table_user as u
   ON e.employee2user = u.objid
  INNER JOIN table_case as c
   ON sc.subcase2case = c.objid
  INNER JOIN table_site as s
   ON c.case_reporter2site = s.objid
  INNER JOIN table_address as a
   ON s.cust_primaddr2address = a.objid
  INNER JOIN table_condition as co
   ON sc.subc_state2condition = co.objid
 WHERE (u.login_name = @Username OR @Username IS NULL) AND
   (e.objid = @EmployeeObjId OR @EmployeeObjId IS NULL) AND
   (co.title = 'Open' OR co.title = 'Open-Dispatch')
 RETURN 0
END
```

Este Stored Procedure recine dos datos de entrada opcionales que son

```
@Username VARCHAR(20) = NULL,
@EmployeeObjId INT = NULL
```

Y con cualquiera de estos busca los datos a trav'es de los joins de varias tablas.

Este es una pequeña muestra del código que hemos llevado a cabo todo un grupo de trabajo, para un sistema muy grande y con varios meses de trabajo, pondré algunas pantallas más con ejemplos de Knockout y algo de Jasmine, únicamente para mostrar al código realizado, y como es el sistema actualmente.

Tenemos alrededor de 40 Stored Procedures, cientos de archivos no sé si ya más de mil métodos pero supongo que sí.

Las figuras 4.5 y 4.6 muestro la pantalla principal en la cual el ingeniero de servicio llena la forma con los datos que se necesitan para procesar las piezas, esta pantalla ha tenido varias modificaciones a lo largo del proyecto, esta es la versión final que tiene un gigantesco backend.

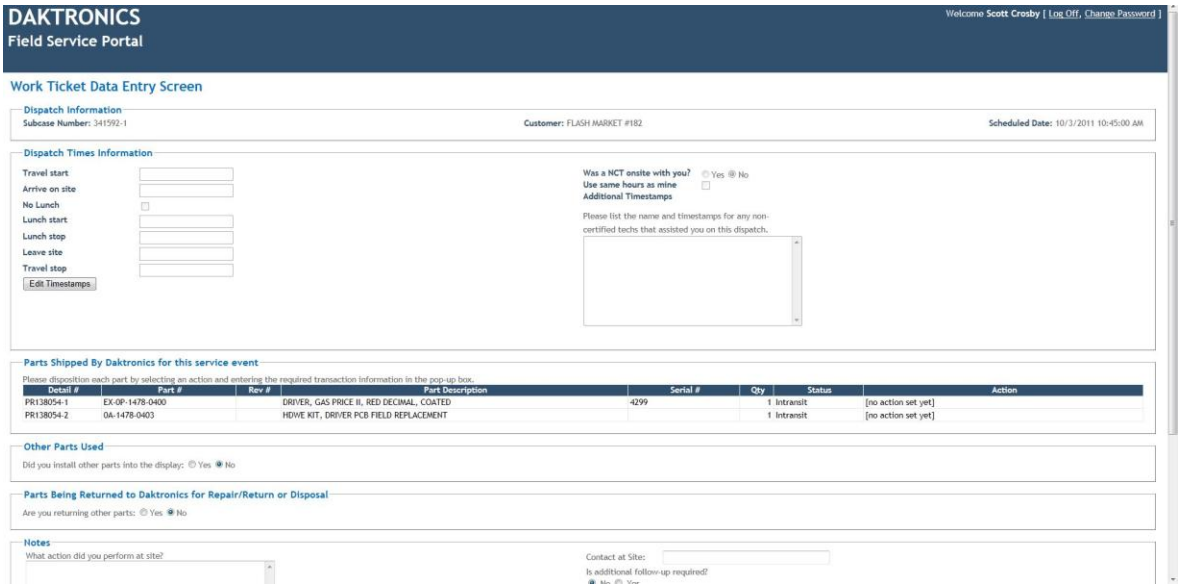


Figura 4.5 Pantalla principal

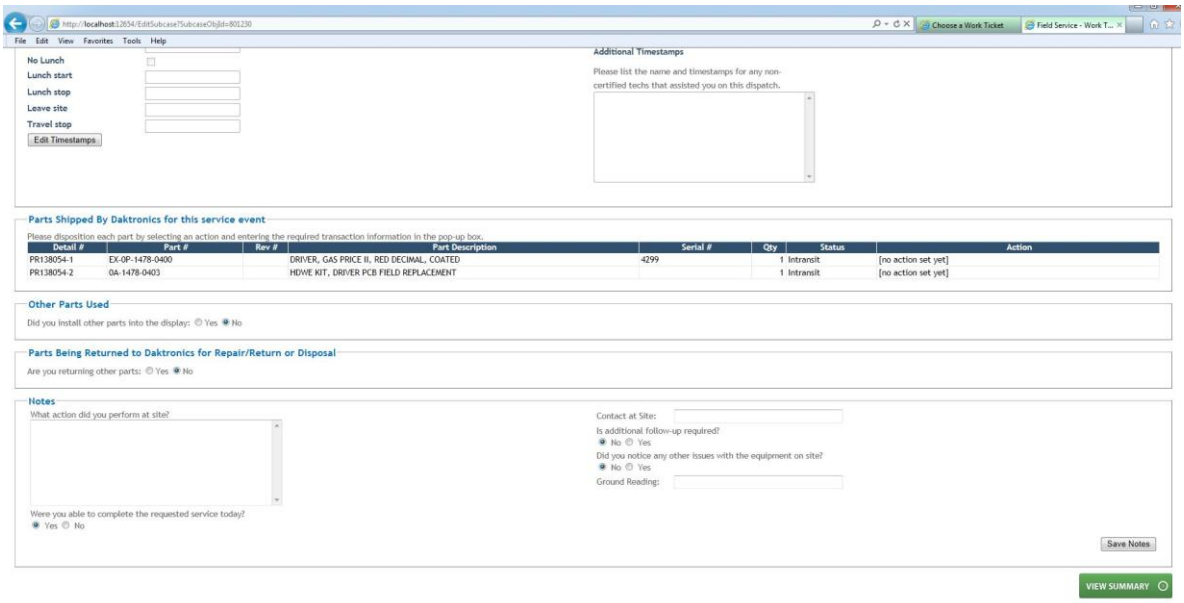


Figura 4.6 Pantalla principal 2

Uno de los pop ups que mencioné arriba que están hechas con JQuery lo podemos ver en la siguiente figura (Figura 4.7):

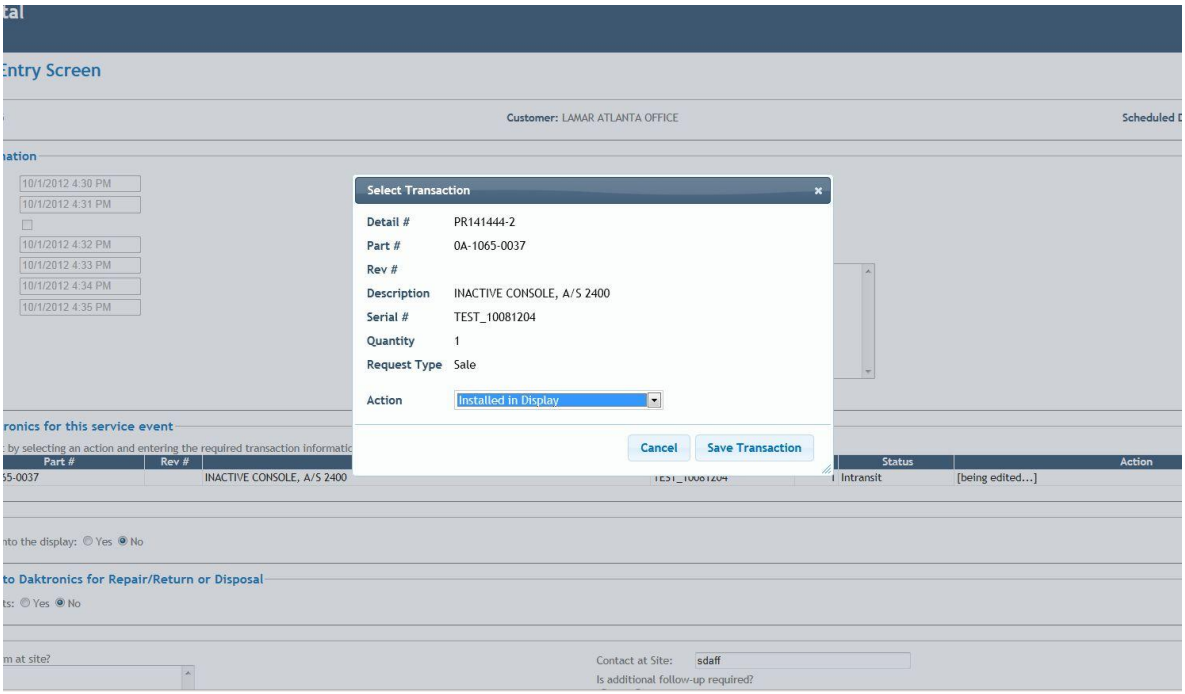


Figura 4.7 Pop up

Este es una parte del code behind de este pop up, podemos ver que es JavaScript, para crear el pop up usamos la función dialog que es una utilería de JQuery que nos facilita el hacer este pop up de una manera impresionante.
Se construyen la uncción Open, la función Save, la función Cancel, la función Closing, entre muchísimas más que tenemos en el código.

```
function PartUseDialog() {
  var self = this;

  self.dialogObj = $('#PartUsePopup').dialog({
    autoOpen: false,
    modal: true,
    width: 600,
    title: "Select Transaction",
    close: function () { self.Closing(); },
    buttons: {
      "Cancel": function () { self.Cancel(); },
      "Save Transaction": function () { self.Save(); }
    }
  });


  self.Open = function (part, afterSaveFunc) {
    self.afterSaveFunc = afterSaveFunc;
    self.part = part;
    part.StartEditing();
    self.closeTrigger = '';
    self.dialogObj.dialog('open');
  };
```

```
    self.Closing = function () {
        if (self.closeTrigger != 'save')
            self.part.CancelEdit();
    };

    self.Save = function () {
        self.closeTrigger = 'save';
        self.part.Save(
            function () {
                self.dialogObj.dialog('close');
                self.afterSaveFunc();
            },
            function (msg) {
                self.closeTrigger = '';
                alert('There was an error. The Transact has not been saved:\n\n' + msg);
            }
        );

    };

    self.Cancel = function () {
        self.dialogObj.dialog('close');
    };
}
```

Mostraré a continuación algo de knockout únicamente para ver como se utilizó en nuestro sistema.

Tenemos al menos 12 MVVM

```
__extends(ReturningPartsViewModel, OtherPartsViewModel);

function ReturningPartsViewModel(subcaseObjId, ReadOnlyMode) {
    var self = this;

    this.ReadOnlyMode = ko.observable(ReadOnlyMode);

    this.SubcaseObjId = subcaseObjId;
    this.Parts = this.PartsCreate();
    this.HasOtherParts = ko.observable('no');
    this.TemporaryParts = ko.observableArray(self.Parts());


    this.HasOtherParts.subscribe(function () {
        if (self.HasOtherParts && self.HasOtherParts() == 'no') {
            self.ClearParts();
        } else if (self.HasOtherParts && self.HasOtherParts() == 'yes') {
            if (_.isEmpty(self.Parts())) {
                self.RetrieveParts();
            }
        }
    });
```

Mostramos primero el nombre de este view model ReturningPartsViewModel este view model se extiende desde OtherPartsViewModel, hacemos observable a HasOtherPart y com default le ponemos “no”.

Veamos ahora como se ve en HTML

```
<script type="text/javascript">
  $(document).ready(function () {
    var editableEntry = _editSubcaseViewModel.OtherPartsInformation().OtherPartEditableEntry;
    var otherPartsDialog = new OtherPartsDialog(editableEntry, 'OtherPartInstalledPopUp');

    _editSubcaseViewModel.OtherPartsInformation().init(otherPartsDialog);
    ko.applyBindings(_editSubcaseViewModel.OtherPartsInformation, $('#OtherPartsSection')[0]);
    ko.applyBindings(editableEntry, $('#OtherPartInstalledPopUp')[0]);
  });
</script>

<fieldset style="padding-top:10px;" id="OtherPartsSection">
  <legend>Other Parts Used</legend>

  <div id="LoadError" data-bind="visible: NoData">No Data Available</div>

  <div id="OtherPartsForm" data-bind="visible: !NoData()">
    Did you install other parts into the display:
      <input type="radio" name="whereOtherPartsUsedGroup" value="yes" data-
bind="checked: HasOtherParts, disable: ReadOnlyMode" />Yes
      <input type="radio" name="whereOtherPartsUsedGroup" value="no"  data-
bind="checked: HasOtherParts, disable: ReadOnlyMode" />No

 <div id="OtherPartsUsed" data-bind="visible: HasOtherParts() == 'yes'" style="width: 100%; text-align: right">
      <table class="standardTable" style="margin: 15px 0; width: 100%">
        <thead>
         <tr>
          <th>Source</th>
          <th>Installed Part #</th>
          <th>Installed Rev #</th>
          <th>Installed Serial #</th>
          <th>Quantity</th>
          <th>Removed Part #</th>
          <th>Removed Rev #</th>
          <th>Removed Serial #</th>

         </tr>
        </thead>
        <tbody data-bind="visible: OtherPartsEmpty"><tr><td colspan="7" style="text-
align: center">No parts have been added yet.</td></tr></tbody>
        <tbody data-bind="foreach: Parts">
        <tr data-bind="click: function() { $root.Select($data); }, attr: { 'class': $root.RowClass($data) }">
          <td class="display-field" data-bind="text: PartSource"></td>
          <td class="display-field" data-bind="text: PartNumber"></td>
          <td class="display-field" data-bind="text: RevisionNumber"></td>
          <td class="display-field" data-bind="text: SerialNumber"></td>
          <td class="display-field" data-bind="text: Quantity"></td>
          <td class="display-field" data-bind="text: ExchPartNumber"></td>
          <td class="display-field" data-bind="text: ExchHardRevisionNumber"></td>
          <td class="display-field" data-bind="text: ExchSerialNumber"></td>
        </tr>
        </tbody>
      </table>
```

```
        <button type="button" data-
bind="click: ClickDeleteButton, enable: HasSelected() && !$root.ReadOnlyMode()">Delete</button>
        <button type="button" data-
bind="click: ClickEditPartButton, enable: HasSelected() && !$root.ReadOnlyMode()">Edit</button>
        <button type="button" data-
bind="click: ClickAddPartButton, disable: $root.ReadOnlyMode">Add a Part</button>
      </div>

    </div>
</fieldset>


<div id="OtherPartInstalledPopUp" class="modalDialog" style="position: relative">
    <span class="legend-like">From:</span><br />
    <select data-
bind="options: SourceOptions, optionsCaption: 'Select Part Origin...', value: Source, enable: !BeingDeleted()" style=
"margin-bottom: 10px;"></select>

    <div data-bind="visible: TrunkInventoryFields_AreVisible">

      <span class="legend-like">Select the Installed Part:</span>
      <div style="max-height: 350px; overflow-y: auto; overflow-x: hidden">
        <table class="standardTable" id="TrunkInventory" style="min-width: 750px; font-size: 85%">
          <thead>
          <tr>
            <th>Part #</th>
            <th>Rev #</th>
            <th>Description</th>
            <th>Serial #</th>
            <th>Quantity</th>

          </tr>
          </thead>
          <tbody data-bind="visible: TrunkInventoryParts().length == 0">
            <tr>
              <td colspan="4" style="text-align: center">There are no parts in your Trunk Inventory</td>
            </tr>
          </tbody>
          <tbody data-bind="foreach: TrunkInventoryParts">
            <tr data-bind="click: $parent.Select, attr : { 'class': $parent.ClassFor($data) }">
              <td data-bind="text: PartNumber"></td>
…
Etc…
```

Podemos ver como en la línea:

```
<div id="OtherPartsUsed" data-bind="visible: HasOtherParts() == 'yes'" style="width: 100%; text-align: right">
```

Le decimos que este div sea visible solo si HasOtherParts es igual a yes, recordemos que lo tenemos inicializado por defecto a “no”. Esto lo podemos hacer gracias a Knockout.

Podríamos hacerlo con JavaScript crudo, pero es mucho más código y no es tan simple.

Tenemos una función de JavaScript que inicializará el valor a “yes” dependiendo de las condiciones.

Una prueba de Jasmin se muestra mediante el siguiente código:

```
describe("Part Action View Model", function () {
  var pact;

  beforeEach(function () {
    pact = new PartActionViewModel();
  });

  it("Part Action View Model must exist and be instanciable ", function () {
    expect(pact).toBeDefined();
  });

  describe("Part Action Tests", function () {

    it("Returning Action Transact texts", function () {
      pact.Action('Returning');
      pact.TrackingNumber('foo');
      expect(pact.TransactText()).toBe("Returning (tracking # foo)");

      pact.Quantity(42);
      expect(pact.TransactText(1)).toBe("Returning (42) (tracking # foo)");
    });
```

Podemos observar primero la sintaxis tan clara y descriptiva para la prueba, es una prueba al Part Action View Model, a la variable pact es un viewmodel que estamos probando, primero vemos si existe con expect.(pact).toBeDefined();

La siguiente prueba es:

Se le asignan valores a pact.Action y a pact.TrackingNumber y le decimos que esperamos que el valor del view model para la función TransactText() sea “Returning(tracking # foo)” .

Estas pruebas se diseñan antes de codificar y se corren antes y después de construir el código.

La primera etapa del proyecto fue el tener la interfaz lista para recibir datos y guardarlos en la base que se construyó. La siguiente etapa fue el procesar estos datos almacenados, esto es, las piezas a regresar y las piezas instaladas.

Cuando una pieza se ha desinstalado del display se lleva a cabo algún proceso dentro del sistema que lo hace actualmente y sólo vemos el resultado en la base de datos y como es que se ve en ese sistema al final de la transacción, así el problema que nos toca es saber cuál es ese proceso y replicarlo.

Lo mismo con la instalación y reinstalación de las piezas

Al tener acceso a la base de datos y al sistema pudimos hacer un trace de lo que sucedía en ella con la herramienta del SQL Server Managment Studio, SQL Profiler, pudimos ver que hace la base de datos y

con mucha ayuda del proceso por parte del área de business, logramos entender en qué consistía el proceso.

En la figura 4.8 podemos ver un screenshot del trace que se hizo del proceso de reinstalar una pieza.

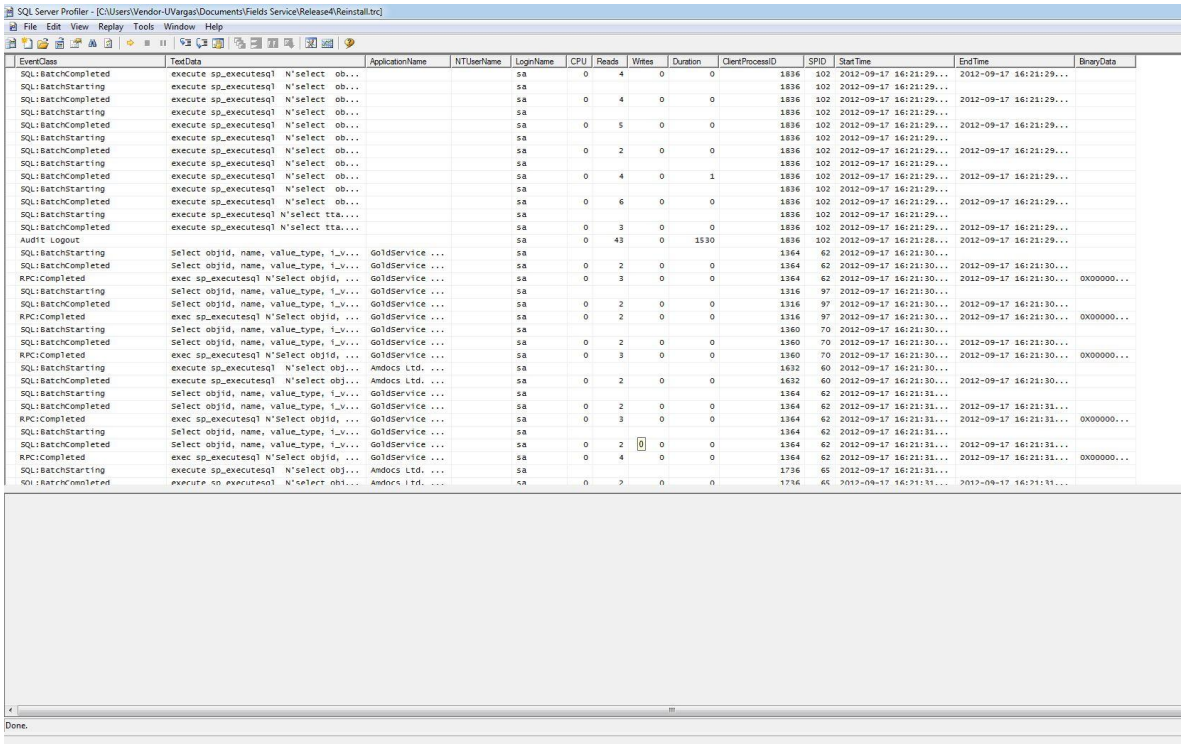


Figura 4.8 Trace de la Base de Datos

El profiler se pone a andar cuando se tiene lista la prueba, para no generar tanto código, y procurar que no haya nadie más usando en ese instante la base de datos para no tener código que nos provocaría un poco de confusión.

Al terminar la transacción lo detenemos y lo analizamos como cualquier código.

En la figura 4.9 se muestra una imagen más cercana, ha sido una gran herramienta que si bien la hemos ocupado poco ha sido indispensable para aclararnos dudas, esta base de datos es compleja y tiene muchísimos campos y tablas y gracias al SQL Server Profiler hemos podido disipar dudas como el

saber qué tabla leyó y qué campos para poder hacer determinado paso, o qué datos ha guardado para lograr que la tabla quede de cierta manera.

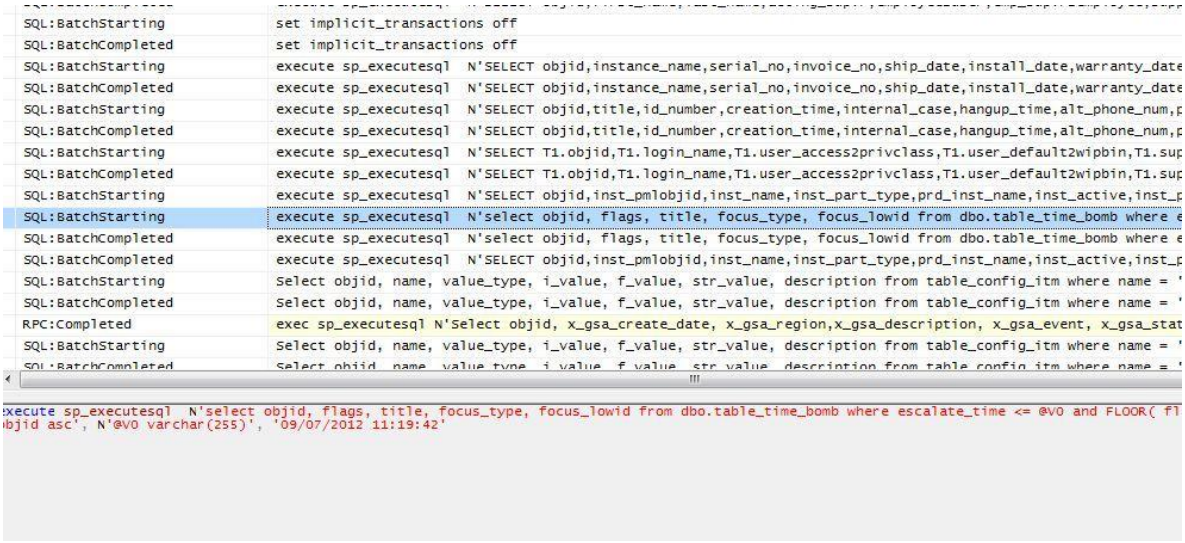


Figura 4.9 Acercamiento del Trace

Con ayuda de estos trazado pudimos comenzar con el proceso de la transacción de las piezas este proceso se ha llevado a cabo todo del lado del servidor de Web Services, ya que sólo se manda la instrucción de comenzar la transacción y comienza el reacomodo de la base de datos para la instalación, la reinstalación y la remoción de piezas.

Este proceso ha sido bastante complejo pues hay demasiadas condiciones que validar, si la pieza es tiene número de serie, si la pieza está en el inventario, si la pieza ha sido instalada anteriormente, si el número de serie es temporal, etc., un sinfín de detalles que cuidar.

Cuando se termina el proceso verificamos con el sistema actual si al hacer una transacción similar las tablas quedan de manera correcta en la base, y si en el sistema la transacción hecha con el WorkTicket se ve igual que una hecha de manera tradicional.

La figura 4.10 contiene un diagrama proporcionado por el área de business y podemos observar que hay una serie de condiciones y validaciones que traducidas a código se hicieron bastante complejas, hay una serie de clases y métodos para cada caso y hay una tabla log que tenemos que actualizar con cada una de las actividades que se hicieron para cada pieza.

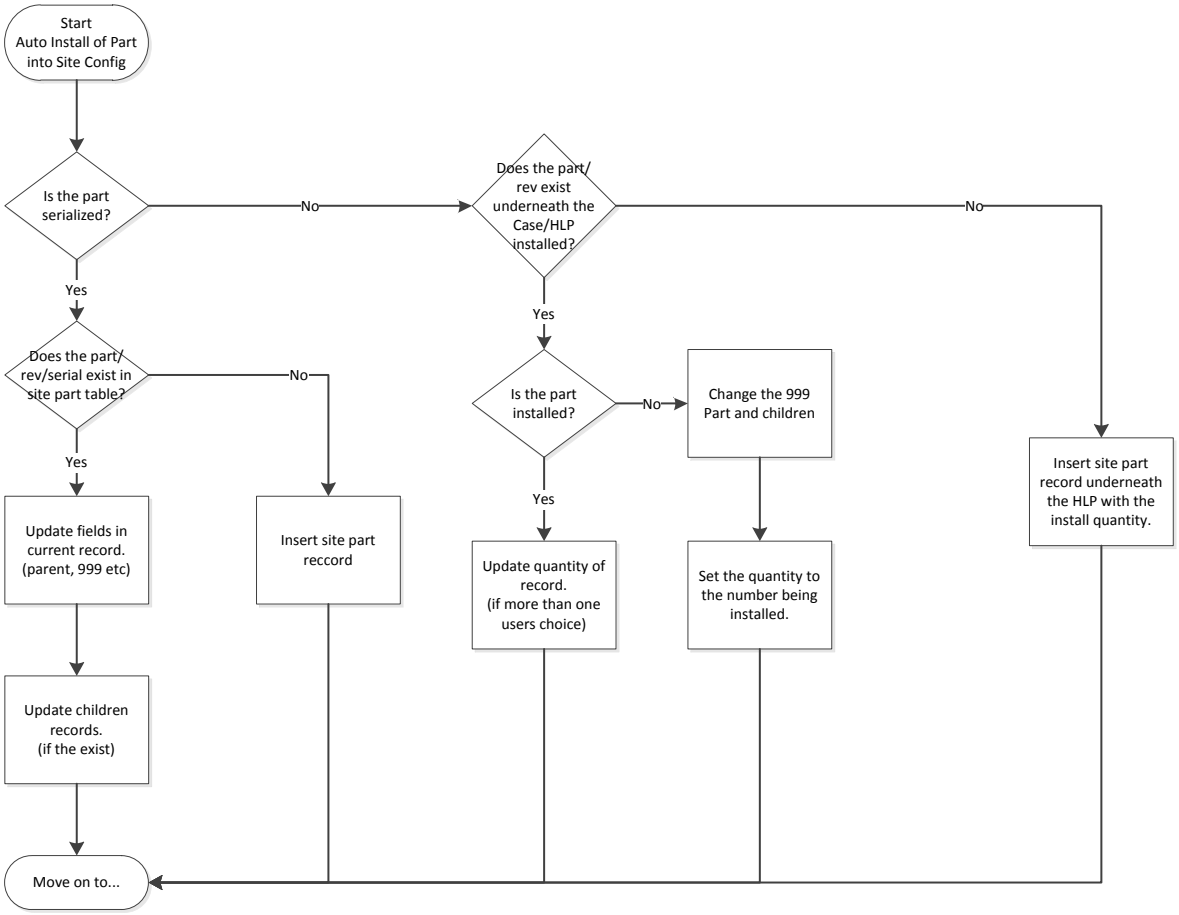


Figura 4.10 Diagrama de flujo del proceso

El proyecto en su fase 2 no ha sido terminado, estamos actualmente en la transacción de las partes intercambiables. Y la complejidad aunque ha bajado debido a que hemos estado reutilizando métodos para las transacciones, el proyecto parece extenderse afortunadamente pues se le ven buenos alcances.

Capítulo 5. Resultados

5.1 Conclusiones

El proyecto comenzó a cobrar mucha importancia debido a que se le vio potencial para resolver el problema de hacer manualmente las transacciones de las piezas, punto que se alcanzo después de mucho tiempo y cuando ya se había entregado aproximadamente la primera mitad del primer release. Muy interesante ver el cambio que representó ver completada la interfaz pues el área de business comenzó a imaginar la dimensión de la ayuda que sería este sistema en la empresa.

Comenzaron a asignar más ingenieros y comenzaron a asignar expertos en esta ‘área del departamento de negocios, para que pudieran ayudarnos con los procesos y con el entendimiento del flujo del negocio.

La importancia que tiene la comunicación, la sana y buena convivencia entre las distintas áreas como es el caso de nosotros con el ‘área de operaciones o negocios (Business) y dentro de la misma área con los distintos roles, como son los DBAs, o los QAs e incluso con los mismos compañeros desarrolladores es fundamental para el éxito de cualquier proyecto.

Afortunadamente el ambiente que existe en esta empresa es de compañerismo, que ha sabido la directora de IT por una parte fomentar y por otra, seleccionar al personal, ha sido muy cuidadosa y eso lo sé porque ha sido una característica que ha enfatizado mucho en el proceso de contratación que llevé y que ha comentado después de ser contratado hemos tenido una buena actitud y gracias a ello fui contratado con mis compañeros. Y durante los primeros meses siempre enfatizaba la importancia de crear un buen ambiente de compañerismo y colaboración.

Otro factor igual de importante es la planeación de los proyectos y me refiero a la planeación directiva, pues a pesar de tener que hacerlo lo más rápido posible, no existe una presión por tenerlo en determinada fecha, pues los directivos saben que es un proyecto grande y lo más importante es que han visto el avance. Esto gracias a Scrum, ya que como hemos descrito uno de los objetivos es tener entregables al final de cada sprint.

Y acerca de entregables puedo comentar que efectivamente al final de cada sprint, tenemos un demo con el personal de operaciones, aunque no es regla, pues debido a que hay ocasiones en que en el sprint no hay entregables que se “visibles”. Pero continuamente se da un demo con las nuevas características que se han agregado al sistema.

Elogio mucho la metodología de trabajo Scrum, porque me ha parecido a lo largo de los años que la he usado una herramienta excelente, no sólo de planeación y de trabajo, sino de integración del personal, una filosofía de compañerismo y de colaboración entre áreas y una forma de trabajo increíblemente efectiva, gracias a los entregables, a la flexibilidad que ofrece para cambiar el rumbo del proyecto (claro tiene sus límites) a la forma de planear cada sprint que como ya lo hemos mencionado consiste en una junta de retrospectiva donde cada uno habla de cómo ha sentido el proyecto en cuanto al

esfuerzo, a los compañeros etc., y proponer como mejorar el sprint siguiente, en esta junta participan todos los que componen el equipo, desde QAs y Business hasta los desarrolladores.

La planeación también involucra mucho dialogo e intercambio de puntos de vista, con la votación secreta se producen muchas diferencias que se aclaran en el momento y esto genera despeje de dudas y teniendo al personal de operaciones presente ayuda mucho.

El proyecto ha sido una enseñanza muy grande y un logro importante tanto para mi como para el personal de México para este proyecto, pues hemos ganado credibilidad y experiencia.

El siguiente paso para mí en lo profesional es convertirme en Project Manager, pues creo que he llegado a un punto importante de conocimientos sobre liderazgo, planeación y estrategias de trabajo. Ganado todo esto gracias a las diferentes empresas y proyectos donde he tenido la oportunidad de desarrollarme, así como la formación y educación que he recibido tanto por mis profesores como por mis amigos y familia.

5.2 Glosario

BDD

Behavior Driven Development o Desarrollo Guiado por Comportamiento, es una técnica de Desarrollo Agil De Software que fomenta la colaboración entre desarrolladores, testers, analistas y personas del negocio en un proyecto de software.

BDD permite un entendimiento más global del sistema a crear, eliminando el efecto túnel de los desarrolladores que trabajan sólo con TDD. También crea un entendimiento común de todos los involucrados y ayuda a eliminar malos entendidos sobre lo que el sistema debe hacer.

De la misma forma que en TDD se escriben primero las pruebas unitarias, en BDD se escriben primero el comportamiento esperado de la aplicación, haciéndolo demostrable y automatizándolo en pruebas de aceptación, lo que se denomina especificación activa.

BDD y TDD no son excluyentes, sino que se pueden aplicar ambas prácticas, complementándose, comenzando el ciclo con las especificaciones con BDD y utilizando TDD en el ciclo de la codificación.

Prácticas de BDD

- Involucrar a los interesados en el proceso
- Usar ejemplos para describir el comportamiento de la aplicación, o de unidades de código
- Automatizar estos ejemplos para proveer un rápido feedback y tests de regresión

Ciclo de desarollo BDD

1. Discusión. Desarrolladores, testers, expertos de dominio y el dueño de producto se juntan y discuten la historia, gradualmente descomponiendo y destilando el comportamiento en un conjunto de especificaciones simples.
2. Decisión. El dueño de producto decide cuando la especificación cumple suficientemente el comportamiento esperado y está de acuerdo con los casos de ejemplo y demostración de la historia y cierra el alcance de la misma.
3. Desarrollo. Los testers refinan los ejemplos de la especificación y los desarrolladores instrumentan las especificaciones creando primero pruebas que fallan y después implementando la funcionalidad.

4. Demostración. Una vez que todos los tests pasan la historia puede ser demostrada al dueño de producto.

JQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos.1 jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

JS

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3 basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, en bases de datos locales al navegador.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Jasmine

Es un framework de pruebas unitarias para JavaScript.

El BDD se ofrece como respuesta a las limitantes del TDD, es una manera nueva de describir comportamiento mas que a las pruebas que acompañan el software.

LDAP

Lightweight Directory Access Protocol. (Protocolo Ligero de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

MVC

Modelo Vista Controlador (MVC) es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Modelo: Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

MVVM

Model view-viewModel, Resumiéndolo podríamos decir que se basa en organizar la aplicación en torno a 3 conceptos:

Model. Es el modelo completo de la aplicación, donde reside la lógica de negocio y todas esas cosas importantes. Knockout no se mete en esta parte y puedes implementarla como quieras. Lo más habitual es que resida en un servidor y la aplicación se comunique con él a través de AJAX.

View. La vista, dedicada únicamente a mostrar información. No contiene ningún tipo de lógica, ni siquiera lógica relacionada con el interfaz de usuario, porque para eso está la última parte del trío.

ViewModel. El modelo de presentación, que se encarga de adaptar el modelo de la aplicación a las necesidades de la vista y contiene además toda la lógica relacionada con el interfaz de usuario, como por ejemplo cuándo se debe habilitar un control o qué hay que hacer cuando se pulsa un botón.

REST

Representational State Transfer. Transferencia de Estado Representacional o REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. Los sistemas que siguen los principios REST se llaman con frecuencia RESTful;

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan

recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)

- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.

- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.

- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

Scrum

Es un proceso ágil y liviano que sirve para administrar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad.

TDD

Test Driven Development o Desarrollo Guiado por Pruebas.

Es una práctica de programación que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización (Refactoring). Para escribir las pruebas generalmente se utilizan las pruebas unitarias (unit test en inglés). En primer lugar, se escribe una prueba y se verifica que las pruebas fallan. A continuación, se implementa el código que hace que la prueba pase satisfactoriamente y seguidamente se refactoriza el código escrito. El propósito del desarrollo guiado por pruebas es lograr un código limpio que funcione

TFS

Team Foundation Server. Es un complemento de Visual Studio que comprende la control de versiones de código y las herramientas para la planeación del trabajo.

Referencias

http://www.quarksoft.net

http://es.wikipedia.org/

http://blog.koalite.com/2012/01/tutorial-jquery-mobile-knockout-i-sentando-las- bases/

http://knockoutjs.com

http://www.chuidiang.com/java/herramientas/test-automaticos/tdd-test-driven-development.php

http://www.dosideas.com/wiki/Behavior_Driven_Development