Montevideo, 29 de septiembre de 2015

Employee Timeline

Alumnos

Ivana Gruszka

gruszka.ivana@gmail.com

Federico Rodriguez

roqui2000@gmail.com

Gonzalo Rodriguez

gonzalo.fing@gmail.com

Tutor

Ing. Ariel Ron

aronzanon@gmail.com

Resumen

La empresa de desarrollo de software UruIT, procura resolver la poca visibilidad que tienen los supervisores y los empleados, sobre las actividades que se realizan a diario, y mejorar la gestión de sus recursos humanos. El objetivo es promover la cultura motivacional, y de involucramiento entre personal, así como también, la toma de decisiones gerenciales.

La información se encuentra distribuida entre distintos sistemas, para las actividades que se desarrollan en el contexto laboral: participación en proyectos, salida de vacaciones, días libres, eventos corporativos, publicaciones en la intranet, entre otras.

El objetivo del proyecto es desarrollar un producto, que permita normalizar y unificar la información dispersa. Se trata de un sitio web unificado, con visibilidad completa de la historia de los empleados, facilitando el análisis y toma de decisiones.

La solución abarca el uso de varias tecnologías que se encuentran en auge, con especial foco en la arquitectura y diseño de la solución. Es clave la flexibilidad y escalabilidad, en la incorporación y parametrización de futuras fuentes de datos, las cuales podrán ser implementadas y adicionadas, sin cambios en la estructura base de la aplicación. Esto a su vez, permitirá al producto adaptarse a las necesidades de la empresa, acompañando sus cambios, o de futuros clientes.

Introducción y contexto

“Las organizaciones del futuro solo podrán adquirir y mantener ventajas competitivas, mediante el uso adecuado de la información, y sobre todo, del conocimiento, debiendo identificarlo, crearlo, almacenarlo, transmitirlo y utilizarlo de forma eficiente, tanto a nivel individual, como a nivel colectivo, con el fin de resolver problemas o mejorar procesos y servicios”.[1]

UruIT es una empresa de desarrollo de software, con negocios internacionales, que actualmente se enfrenta a un problema de gestión de la información y capital humano. Esta es una dificultad actual que deben enfrentar las empresas, particularmente al procurar ser referente es sus principales líneas de negocio.

La empresa se apoya en el software para la gestión diaria, utilizando distintos sistemas y herramientas que le permiten realizar el registro de las actividades, y hechos que ocurren en la vida laboral de cada uno de sus empleados.

Actualmente, la información se encuentra distribuida entre distintos sistemas, y no se cuenta con una herramienta que la unifique, brindando escasa visibilidad de las actividades que se realizan en la empresa.

Employee Timeline, es un sitio web que provee visibilidad de la historia de los empleados en la empresa, normalizando y unificando la información de los distintos sistemas.

Producto

El producto desarrollado brinda las siguientes principales funcionalidades: integración con distintas fuentes de datos, importación y autenticación con Active Directory, interfaz visual interactiva de la línea de tiempo de los empleados, posibilidad de agregar eventos manuales y administración de permisos.

Es posible comentar los eventos de otros empleados, y recibir notificaciones por comentarios y eventos agregados en la línea de tiempo propia.

Los colaboradores, a su vez pueden visualizar, los proyectos en los que han participado, las vacaciones que han usufructuado, y también generar un Curriculum Vitae, en formato Documento de Word, con su información.

Se destaca por último la exportación de eventos a Excel, permitiendo posteriormente cargarlos en distintos posibles sistemas, para el análisis y graficado de datos.

Tecnología

El presente proyecto, supuso un gran desafío en el análisis y uso de distintas tecnologías y herramientas para el desarrollo, entre las que se destacan entre otras:

Requeridas por el cliente: Visual Studio 2013 , C#, SharePoint, Active Directory, MSSQL Server, GIT.

Definidas por el equipo: Entity Framework, Windows Communication Foundation, AngularJS, NodeJS con ExpressJS, Jenkins, Grunt, Bower.

[1] Post-capitalism society. Peter F. Drucker. Butterworth-Heinemann. Nueva York, 1993.

Arquitectura

Arquitectura lógica

Estructuralmente la aplicación está dividida en dos grandes componentes: frontend y backend.

Frontend

El frontend fue realizado utilizando AngularJs , framework de JavaScript, para la realización de aplicaciones web single page, utilizando el patrón de diseño MVW.

Complementariamente, se hizo uso de un servidor http liviano, ejecutando NodeJs, lo cual permitió construir la aplicación rápidamente y en un mismo lenguaje de programación, utilizando del web framework REST ExpressJS, el sitio web que contiene el punto de entrada a la aplicación.

A continuación se muestra un diagrama con lo mencionado anteriormente:

Figura 1 – Arquitectura del frontend

Backend

El backend está dividido en dos grandes capas lógicas: capa de servicios y capa de persistencia. Adicionalmente, existe un componente llamado SyncService, que se encarga de sincronizar los distintos proveedores de datos configurados, así como también, una capa vertical que contiene todos los Value Objects utilizados para el transporte de dicha información.

A continuación se muestra un diagrama con lo mencionado anteriormente:

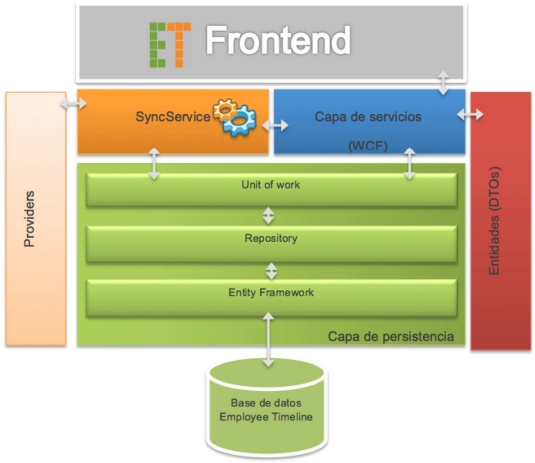

Figura 2 – Arquitectura del backend

Capa de persistencia

La capa de persistencia accede a la base de datos mediante ADO.NET Entity Framework 5, con un enfoque, código primero. Esta perspectiva permitió ir creando la base de datos a medida que el modelo de la aplicación fue evolucionando, en base a los requerimientos que se fueron desarrollando.

Capa de servicios

Esta capa está desarrollada con WCF, que expone los servicios necesarios para que la aplicación cliente pueda interactuar y usar las diferentes operaciones del sistema.

Módulo SyncService

Este módulo está construido para ser ejecutado como un Servicio de Windows, cuyo cometido es verificar

las diferentes fuentes de datos configuradas en el sistema. El mismo controla, según una frecuencia configurada para cada origen de datos, usando el proveedor adecuado, la sincronización de la información hacia la base de datos del sistema.

Este módulo está basado en el patrón de diseño llamado Provider Model propuesto por Microsoft.

Arquitectura física

La distribución física de los componentes de la solución, se corresponde a una arquitectura N-Tier compuesta por los siguientes componentes: clientes que se conectan a través de navegadores, servidor web en donde está alojada la aplicación cliente, servidor de aplicaciones en donde están expuestos los servicios para resolver los distintos requerimientos del sistema y se conecta a los distintos proveedores, servidor de base de datos MSSQL Server para la persistencia de datos del sistema y fuentes de datos como ser SharePoint o MSSQL Server.

A continuación se muestra un diagrama con lo mencionado anteriormente:

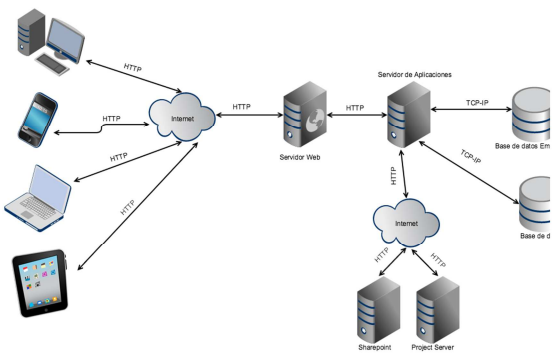

Figura 3 – Arquitectura física

Pruebas realizadas

Se realizaron pruebas funcionales de caja negra, evaluando que cada funcionalidad hace lo esperado.

A su vez, se realizaron pruebas de regresión, para asegurar que los cambios recientes en una parte de la aplicación, no tienen efecto adverso en otras. En cada sprint, también se ejecutaron las pruebas anteriores, para comprobar el correcto funcionamiento de lo liberado anteriormente.

También se realizaron pruebas de humo, antes de cada revisión con el cliente, ejecutando un conjunto de pruebas sencillas de alto nivel, para probar el correcto funcionamiento de las principales funcionalidades.

Finalmente, se realizaron pruebas de aceptación alfa, en cada revisión con el cliente.

Gestión del proyecto

En concordancia con la experiencia del cliente y las características del proyecto, se analizó y definió, para el desarrollo, el uso de la metodología ágil Scrum.

Se comenzó definiendo una pila de producto, con los requerimientos priorizados por el cliente, la cual sirvió como insumo para la estimación de las tareas requeridas y planificación de los sprints.

Se definieron los siguientes planes para dar soporte a la gestión: plan de calidad, plan de comunicación, plan de gestión de control, plan de métricas, plan de riesgos.

Los mismos permitieron, junto con las reuniones definidas por la metodología y sus artefactos, llevar adelante el proyecto, realizando ajustes y tomando decisiones para lograr un producto de calidad, de acuerdo a lo requerido por el cliente, y así cumplir con sus expectativas.

Conclusiones

La evaluación del proyecto es muy satisfactoria, pues se logró producir un producto con la calidad esperada por el cliente, con una arquitectura robusta y mantenible, que facilitará que se pueda seguir evolucionando y trabajando en la herramienta.

Se siguió de manera acorde la metodología planteada, y cumpliendo con los requerimientos formales del proyecto de grado.

Ejecutar el plan de comunicaciones con el apoyo del cliente ayudó a que el equipo contara con retroalimentación y pudiera avanzar en el camino correcto en el desarrollo.

Se recibió apoyo en cuanto a la documentación y proceso de parte del tutor Ing. Ariel Ron, brindando una guía clara de los requerimientos formales necesarios para lograr un proyecto exitoso, en todos sus sentidos.

Referencias

Martín Alaimo. Proyectos Ágiles con #Scrum. Kleer, Buenos Aires, 2014.

Manual del Guerrero: AngularJs. Carlos Solis, 2015.

Ian Sommerville. Ingeniería del software. Séptima edición. Pearson Educación, Madrid, 2005. .

Ingeniería de Software. Un enfoque práctico (Quinta Edición). Roger S. Pressman. ISBN- 84-481-3214-9.

AngularJS, O'Reilly Media, ISBN-13: 978-1449344856.