2019

Taller .Net

SEBASTIÁN DIAZ (SEBADIAZ95@HOTMAIL.COM)
MAURO PISTÓN (PECAALTA@GMAIL.COM)
LUCAS MONTELONGO (LUCASMONTELONGO@OUTLOOK.COM)

RESUMEN
Se realizó una plataforma para hacer seguimiento de paqueterías, resolver la logística orientada a notificar en todo momento las ubicaciones con sus correspondientes tiempos estimados en los envíos, proporcionando un entorno web para el cliente estándar, facilitando de esta forma la visualización del estado de sus encomiendas. También se implementó una Api Rest para utilizar las funcionalidades desde otros servicios o aplicaciones externas, una aplicación móvil pensada para uso interno de los empleados, y para empresas externas que utilicen nuestro servicio de notificación una Api Rest y Soap con un método universal donde se les permita editar, eliminar y crear todos los datos requeridos desde un mismo método simplificando el uso de los servicios.

Palabras Clave
NET, Servicios Web, MVC, Xamarin, App, Api Rest, Soap, plataforma

INTRODUCCIÓN
Se desea el desarrollo de una aplicación para la gestión de envíos, permitiendo a la plataforma notificar el estado de estos a sus destinatarios. Se permite a las empresas gestionar las agencias y trayectos para poder trabajar y a los clientes administrar sus encomiendas y proporcionar un estimado de los tiempos de entrega.
Inspirado en esto se planteó una plataforma dividida en 4 proyectos: una Api Rest, pensada para uso interno, liberando todos los métodos utilizados en los otros proyectos para centralizar la lógica de negocio, facilitando la modificación en esta, por otro lado el sitio web para uso público, donde se visualizarán de mejor forma los datos relevantes tanto para administradores como para clientes, una aplicación para uso interno y una aplicación que proporciona un servicio tanto en Api Rest como Soap.

MARCO CONCEPTUAL

Multi-tenant
Tenencia múltiple o multitenencia en informática corresponde a un principio de arquitectura de software en la cual una sola instancia de la aplicación se ejecuta en el servidor, pero sirviendo a múltiples clientes u organizaciones.

Api Rest
La transferencia de estado representacional (en inglés representational state transfer) o REST es un estilo de arquitectura software para sistemas. Enfocado en la exposición de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global a través de una interfaz estándar (HTTP).

Soap
SOAP (originalmente las siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Código QR
Un código QR es la evolución del código de barras. Es un módulo para almacenar información en una matriz de puntos o en un código de barras bidimensional.

Xamarin
Xamarin es una plataforma para desarrollar aplicaciones para plataformas iOS, Android, Windows Phone, Windows Store y Mac usando el lenguaje de programación C#.

Azure
Azure es la plataforma de computación en nube pública de Microsoft.

ADO.Net
ADO.NET es un conjunto de componentes del software que pueden ser usados por los programadores para acceder a datos y a servicios de datos. Es parte de la biblioteca de clases base que están incluidas en el Microsoft. NET Framework.

Entity Framework
Es un conjunto de API de acceso a datos para el Microsoft .NET Framework.

NuGet
NuGet es un administrador de paquetes gratuito y de código abierto diseñado para la plataforma de desarrollo de Microsoft.

DESCRIPCIÓN DEL PROBLEMA

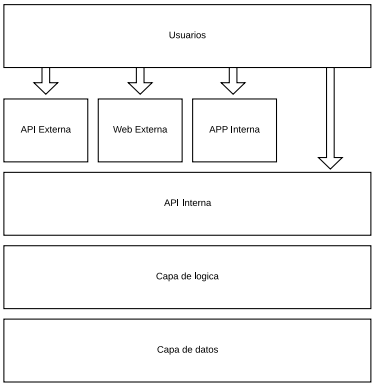
Se planteó un sistema de envíos de encomiendas online, que pueda gestionar los envíos orientado a ventas online.
Existiendo para esto varios roles en los distintos proyectos con distintas funcionalidades a su disposición que serían el super administrador quien puede gestionar las empresas y usuarios, administrador de cada empresa que maneja los trayectos con sus agencias y puntos de control correspondientes, además podrá visualizar reportes gráficos de los paquetes en tránsito, funcionarios, los cuales podrán avanzar, retroceder o entregar paquetes, mediante la utilización de una aplicación que les permita esto y además de agregar, editar y modificar clientes. Estos serían el ultimo rol, (cliente) quienes podrán ver su historial de paquetes y el estado de estos mediante un código de traqueó.
El sistema por su parte deberá enviar un mail en el cambio de estado del paquete para notificar su nuevo estado además del envío del código el cual deberá de estar tanto en texto como en código QR para su entrega final. También se generará una etiquetas en PDF.
Finalmente, se deberá de contar con un login y registro para los visitantes externos y un despliegue en Azure.

SOLUCIÓN PLANTEADA

Se presentó una división de 4 proyectos básicos, donde la api interna sería la API Rest, la cual funciona como núcleo del proyecto, conteniendo separadas, el acceso a datos, lógica y la interfaz que publica los métodos.
A parte se presenta una Aplicación Xamarin que se divide en la lógica de Xamarin y la aplicación Android que ejecuta esta lógica.
Además, se desarrolló una api externa SOAP y Api Rest con un patrón de diseño MVC que expone un único método, pero este solo hace de salida, ya que los datos y lógica están en la antes mencionada api interna.
Por último, se presenta una web que expone los datos necesarios también con un patrón MVC.

ARQUITECTURA DEL SISTEMA

El sistema se separó en 4 partes, una centraliza la lógica y las otras 3 hacen de interfaces ya sea al funcionario, el usuario o a empresas que usen los servicios parciales.
Por un lado, tenemos lo que es el api interno que utiliza el estándar Rest para publicar los métodos, es la única que tiene acceso a la capa de lógica.
Es de aclarar que por el diseño proporcionado las otras partes solo utilizan métodos de este api y por tanto proporciona ciertas ventajas como el centralizar la lógica.
Luego tenemos una Aplicación móvil desarrollada en Xamarin, la cual consiste en dos proyectos, uno propiamente Xamarin que contiene el llamado al api y otro con un proyecto Android el que ejecuta el anterior.
Luego está la web externa, brindando un acceso rápido a los usuarios que usen los servicios brindados.
Por último, estaría el api externo el que utiliza los estándar Rest y SOAP y proporcionan un único método publico



IMPLEMENTACIÓN

En la implementación se utilizaron tecnologías de Microsoft como Azure, Xamarin, Entity Framework o ADO.Net

Todas ellas tienen en común ser muy robustas, con mucha personalización y denotan un enfoque empresarial.

Siendo Azure el más completo en servicios de alojamiento en la nube, pero también lo hace el más complejo de utilizar, sumado a cierta lentitud en su panel. Dado la gran cantidad de opciones lo hacen una herramienta muy capaz, pero solo recomendable para proyectos de mayor envergadura o con un conocimiento mínimo de la misma.

Xamarin es otra de las herramientas muy útiles que permite con un mismo código generar aplicaciones en varias plataformas, robusto, genera una aplicación ligera y eficiente, pero presenta ciertas limitaciones como en la personalización.

Productos y Herramientas

Tabla 1. Evaluación de Productos y Herramientas

Producto	Puntos Fuertes	Puntos Débiles	Evaluación General
Azure	Tiene toda herramienta que puedas necesitar, y opciones para configurar cada elemento.	Tiene demasiadas opciones, lo hace complicado y lento en muchos casos, incluso es caro al darte cosas que no se necesitan.	No te deja muchas opciones si trabajas con aplicaciones de Microsoft, realmente fue complicado.
Xamarin	Es practico y ligero, y proporciona ahorro de tiempo si se desea crear aplicaciones multiplataforma.	Al ser un código no nativo hay ciertas limitaciones.	Lo usaría si es necesario, pero prefiero otras alternativas más sencillas y que producen un código realmente nativo.
Chart.js	Muy flexible y fácil de usar, con muchas configuraciones.	Es en JavaScript por lo que no se integra con un proyecto .NET tal cual	Es muy potente y fácil de usar.
NuGet	Muy potente, facilita la descarga de paquetes y su utilización.	No distingue tipo de proyecto por lo que a veces terminas descargando paquetes incompatibles.	Muy útil y eficiente, los problemas se dan con casos muy concretos y particulares.

Problemas Encontrados

- En Azure la interfaz es poco intuitiva con demasiadas opciones, sumado a que dio problemas con la suscripción de estudiante y el saldo otorgado se gastaba con rapidez.
- En Xamarin existen los componentes de nuget que en muchos casos no son aptos para aplicaciones de Xamarin, sumado a que no es retro compatible con sus versiones viejas.
- El api externo presentó problemas de configuración para poder ser SOAP Y Rest.

EVALUACIÓN DE LA SOLUCIÓN

La solución implementada parece una buena implementación, falta test, y detalles estéticos.
Por su diseño es relativamente fácil de entender y de actualizar.
Al estar casi toda la lógica en el api, aunque no se repite código, esto también provoca que cualquier bug se pueda presentar en cualquier proyecto.

DESARROLLO DEL PROYECTO

Se desarrolló durante un periodo de 3 meses con 3 integrantes, las tareas se dividieron de tal forma que el desarrollo no se solape siendo un mimbro para el desarrollo de la web, su diseño y accesibilidad, otro para el api externo y la aplicación Xamarin y otro para el desarrollo de la lógica de negocio. Más allá de la división cada integrante retoco ciertas instancias de cada uno para tener un mínimo de conocimiento en las tecnologías.
De igual manera se perdió mucho tiempo en pruebas dado que realmente ninguno conocía las tecnologías aplicadas.
Los tiempos de desarrollo fue de un 77% del tiempo contando tipos de pruebas mencionado, problemas encontrados con Azure, con el proyecto web, problemas con la lectura del código QR en la app y demás, siendo donde se desvió más el tiempo planificado quitándoselo a la etapa de test.
Se destinó un estimado de 9% a diseño meramente estético, 10% investigación y 4% a test.

CONCLUSIONES Y TRABAJOS A FUTURO

Se encontraron problemas por falta de experiencia y conocimientos en las herramientas, falta de planificación y de trabajo en equipo en algunas ocasiones.
A pesar de estos el producto final fue el alcanzado, aunque carece ciertos detalles, el desarrollo fue instructivo conociendo tecnologías de Microsoft como Azure, Entity Framework o ADO.Net entre otras comprendiendo algunas ventajas, desventajas como cierto alcance y limitaciones que estas pueden tener.

Con respecto a trabajo a futuro algunas funcionalidades a agregar podrían ser:

- Agregas sección de consultas, formulario o incluso chat.
- Implementar aplicación para el público.
- Agregar etiquetas a trayectos para indicar distintos tipos de envíos.
- Agregar etiquetas a los paquetes o categorías para indicar paquetes frágiles, aclaraciones o indicaciones para enviar animales, etc.

REFERENCIAS

1. Laboratorio de Integración de Sistemas. Taller de Sistemas de Información 1 – Trabajo Obligatorio. Año 2014.
2. Microsoft .NET Framework http://www.microsoft.com/net/ - [Consulta: Mayo 2010]
3. ACM Proceedings Templates http://www.acm.org/chapters/policy/toolkit/template.html [Consulta: Mayo 2010]