Contenido

[Resumen. 1](#_Toc60902679)

[Introducción y contexto. 2](#_Toc60902680)

[Flujo de datos. 2](#_Toc60902681)

[Arquitectura. 4](#_Toc60902682)

[1- Azure Event Hubs. 4](#_Toc60902683)

[2- Azure Stream Analytics. 8](#_Toc60902684)

[3- Azure functions. 10](#_Toc60902685)

[4- Azure Notification Hubs. 11](#_Toc60902686)

[5- Azure Datalake Storage. 13](#_Toc60902687)

[6- PowerBI. 14](#_Toc60902688)

[Análisis de los datos con Python 17](#_Toc60902689)

[Presupuesto 22](#_Toc60902690)

[Conclusiones del proyecto 22](#_Toc60902691)

[Bibliografía 24](#_Toc60902692)

[Anexos 24](#_Toc60902693)

Resumen.

El objetivo del proyecto es conseguir desviar una parte del tráfico que recibe un centro de atención telefónica para descongestionar el centro y mejorar los niveles.

En un primer momento, me centraré en aquellas llamadas y aquellos clientes que se pongan en contacto con nosotros con el motivo principal de obtener una cita con un médico.

Estos clientes serán estudiados y se tomarán en cuenta determinadas variables que me permitirán dividirlos y crear segmentos para crear un grupo de clientes que sean susceptibles de utilizar otros medios que no sean una llamada al centro de atención telefónica y llevarlos a que cojan la cita a través de, por ejemplo, la app.

Para ello he decidido utilizar una solución Cloud de Azure que me permita obtener los datos de las llamadas, el tratamiento de estos y una posterior comunicación con los clientes.

Mi intención es que al recibir la llamada y tras la posterior categorización del cliente se le mande una notificación o correo indicándole que tiene la posibilidad de hacer la gestión por la app o la web.

No es necesario que sea Real Time, pero si próximo a ello, para que el cliente tenga un efecto “WOW” y tenga fresca la llamada al servicio.

En definitiva, el objetivo es tratar dos segmentos de clientes.

* Digitales: aquellos que tengan posibilidades de utilización de la app o la web.
* No digitales: aquellos clientes que por sus características van a seguir utilizando el teléfono.

El análisis de estos clientes lo realizaré en un primer momento con Python y las conclusiones y posibles nuevos reportes los plasmaré gracias a PowerBI.

Por lo tanto, todo el proceso se hará en el universo Microsoft.

Introducción y contexto.

La necesidad de realizar este tipo de proyecto se ha visto acentuado tras la caída del confinamiento.

Al estar en una empresa del ámbito de los seguros de salud, durante los meses donde estuvimos confinados en casa no se pudieron realizar distintas pruebas médicas ni revisiones.

Es por ello, que hemos sufrido un aumento de la actividad muy importante, hasta tal punto que hemos recibido más llamadas en agosto, que, en diciembre o enero, que es donde se produce la mayoría de las renovaciones de los seguros y donde más actividad hay a lo largo del año.

Además, dentro de las llamadas totales que se producen a lo largo del día, un porcentaje muy elevado de ellas (cerca del 20%) son para temas de citas.

Esta tipología, por tanto, tiene mucha importancia y podemos usarla de palanca para mejorar nuestros datos, además de conocer qué tipo de clientes utilizan nuestros servicios.

Conseguir disminuir el trafico de llamadas que nos llega al centro de atención telefónica nos permite ahorrar dinero en contrataciones nuevas y nos ayuda a mejorar la atención a nuestros clientes, pudiendo dar más valor añadido que simplemente la obtención de una cita.

Por lo tanto, si conseguimos implementar el proyecto y que los resultados sean satisfactorios podemos ahorrar en costes y a la vez mejorar la imagen de compañía y dando un servicio de 10 a todos los clientes y utilizar este tiempo que no estamos atendiendo a las citas para realizar campañas de información y de outbound.

Flujo de datos.

En primer lugar, hay que tener en cuenta que los datos de las llamadas provienen de un sistema llamado Genesys que gracias a él se pueden medir los volúmenes, tiempos, enrutamiento de llamadas, control de canales, etc. Es decir, es un software que se ha comprado con el objetivo de tener los datos de las llamadas, los correos y los chats bajo una misma base de datos específicamente hecha y diseñada para este tipo de tareas.

Es por ello por lo que los datos los extraeremos de este sistema, podremos ingestarlos cada 15 min que es cuando el software compila los resultados de las llamadas y el de los clientes, por lo tanto, nuestro proyecto se basará en “Near Real Time” por la disponibilidad de los datos.

En nuestro caso el periodo de tiempo del que estamos hablando no nos supone un gran perjuicio, porque en principio lo que queremos es evitar la rellamada del cliente identificando aquellos que puedan ser susceptibles de usar los servicios web, para ello analizaremos una serie de clientes que se pusieron en contacto con nosotros a través del teléfono y luego usaron el mismo servicio en los servicios web y clientes que no han usado la App pero que si han pedido una cita a través de nuestro centro.

Me gustaría encontrar aquellos clientes que tengan más posibilidades de llamarnos y prevenirlos con algún tipo de campaña outbound o con algún tipo de notificación push en la App, de esta manera también podríamos evitar más llamadas al centro de atención telefónica.

En definitiva, queremos conseguir que los datos que nos llegan desde el software del centro de atención telefónica los podamos convertir en datos que una vez analizado nos reporte un beneficio, ya sea, al reducir el numero de llamadas, lo que provocaría menos necesidad de personal o creando un efecto positivo por la atención que se le da al cliente desde la empresa, ya que, conseguiríamos personalización para cada uno de los asegurados mejorando su proceso de contactación con la empresa.

Para ello, me he planteado realizar una arquitectura Cloud utilizando los servicios de Azure.

Creo que puede ser una buena oportunidad para utilizar este tipo de soluciones y aprender un poco sobre ellas.

El recorrido del dato empezará en los datos que tendremos disponibles y que se ingestarán en la nube mediante Azure Event Hubs.

Marcaríamos una ruta donde estos clientes estuvieran disponibles y a ella accedería Event Hub e ingestaría los datos que necesitamos para realizar todo el proceso.

Tras el proceso de ingesta utilizaremos Stream analytics para tratar los datos que necesitamos y generar un disparador que permite la comunicación con el cliente, o bien, guardarlos en el datalake de Azure para tenerlos almacenados por si necesitamos realizar análisis posteriores.

Mi idea es conectar Stream Analytics con PowerBI para tener una imagen en casi tiempo real de lo que esta sucediendo en el servicio y montar una serie de visualizaciones que nos permitan de un vistazo saber cómo está transcurriendo la operativa.

Por otro lado, quería utilizar Azure Functions para crear un disparador que conecte Stream Analytics con Notification Hub, pero viendo la dificultad que esto entrañaba finalmente decidí utilizar un envío de correo a través de las funciones de Azure, que es la herramienta que nos permitirá la comunicación con el cliente.

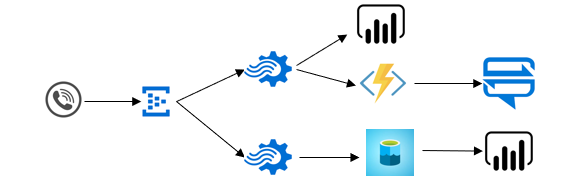
En Stream analytics cogeremos los datos y los enviaremos a distintos outputs para que la comunicación únicamente vaya dirigida a los clientes que nosotros necesitamos.

El objetivo final es que los clientes que cumplan los criterios establecidos sean informados a través de una notificación en la App de las posibilidades de utilizar otro medio que no fuera la llamada al centro de atención telefónica.

Por ello creo que no es excesivamente critico que sea en tiempo real completo, porque la segunda llamada no se produce en un margen tan corto de tiempo y menos aun actualmente, ya que, los niveles de atención son bajos y es complicado que puedas contactar en pocos minutos.

En definitiva, para conseguir el objetivo marcado me he propuesto la siguiente arquitectura.

Tiene una parte en real time que nos permite la comunicación con los clientes y la monitorización de la actividad y luego otra parte batch que nos proporcionará la oportunidad de realizar análisis de distintos tipos según la necesidad del negocio.



Arquitectura.

En esta ocasión creo que la arquitectura elegida debe ser una arquitectura Lambda, que convine el proceso de real time junto con el almacenaje de datos para su posterior análisis y tener la posibilidad de realizar cruces con otros datos que necesitemos.

En la parte Real Time utilizaré los siguientes aplicativos del universo de Azure.

* Azure Event Hubs
* Stream Analitycs
* Azure functions
* Notification hubs/ Azure functions

En cuanto en la parte batch utilizaremos aparte de Azure Event Hub y Stream Analitycs:

* Azure Datalake
* PowerBI

# Azure Event Hubs.

Azure Event Hubs es una plataforma de streaming de datos y un servicio de ingesta de eventos que puede recibir y tratar millones de eventos por segundo, almacenarlos en el cloud, leerlos y usarlos en una enorme cantidad de aplicaciones.

Esta herramienta nos permite la recuperación de datos de una manera rápida para un procesamiento en real-time.

Esta herramienta es una plataforma como servicio (PaaS), es decir, es un entorno de desarrollo e implementación en la nube que tiene un recorrido completo en el ciclo de vida de una aplicación web: compilación, pruebas, implementación, administración y actualización.

Esto nos permite centrarnos únicamente en el análisis de los datos y deja a un lado la dificultad que supone la compra de licencias, la infraestructura, los conectores o las herramientas de desarrollo. De esta manera nosotros solo administramos lo que hemos construido y el proveedor se ocupa de los demás quebraderos de cabeza.

Alguna de las ventajas que tiene este modelo van desde reducir el tiempo de programación, ya que, la herramienta puede tener ya diferentes componentes pre-progamados como los flujos de trabajo o las opciones de seguridad.

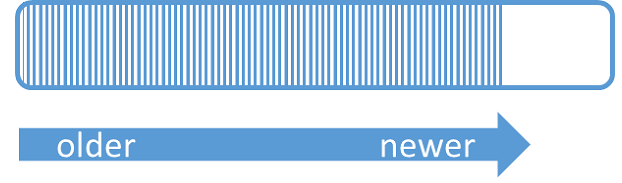
Agregan funcionalidades propias de las herramientas que si las necesitaras de otra manera deberías tener empleados expertos en la materia.

Por supuesto, son herramientas muy avanzadas a un precio más o menos asequible.

Administras el ciclo de vida de la aplicación de manera unitaria y en el mismo entorno, lo que facilita la tarea de administración.

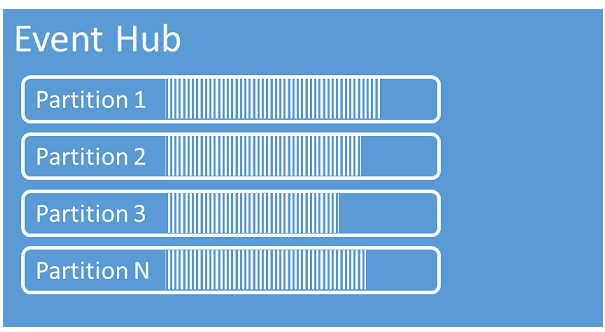
Event hubs usa un modelo de consumidor con particiones que es que cada consumidor lee solo un subconjunto especifico o una partición del flujo de datos. Esto nos permite un escalado horizontal para los procesamientos.

Una partición es una secuencia ordenada de eventos que a medida que llegan mas eventos se agregan al final de la secuencia.



Event hub retiene los datos durante un tiempo configurable que se aplica a todas las particiones, por lo tanto, los eventos terminan en función del tiempo que nosotros hayamos elegido.

Es por esto por lo que las particiones son independientes y cada uno contiene su propia secuencia de datos y claro, a menudo se componen de diferente manera.



El número de particiones debe estar especificado en el momento de la creación y debe estar comprendido entre 1 y 32. Este dato no es modificable por lo que hay que tenerlo en cuenta cuando se está diseñando la solución.

Cada partición se rellena con unas secuencias de datos que contienen el cuerpo del evento, un contenedor de propiedades que definimos nosotros mismos y diversos metadatos, como su número de secuencia.

Una opción que nos da la herramienta es la introducción de una clave de partición, básicamente esta clave es utilizada si el orden de los eventos es importante, en nuestro caso no seria importante ya que, únicamente queremos el cliente sin más.

Además, si ponemos una clave a nuestros eventos podríamos tener el problema de que el proceso fallara si la clave no está hecha correctamente, o, fallara en un momento dado, el proceso de ingesta

Lo que provocaría que si el proceso busca la clave y no la encuentra todo el proyecto fallaría y no seguiría corriendo.

Por supuesto es escalable, puede empezar utilizando unos pocos megabytes de datos y aumentar según necesidad a incluso terabytes.

Tiene una característica de escalado automático que nos permite si la activamos que el proceso de escalamiento se realiza según las necesidades del servicio sin tener que preocuparnos por adaptarlo nosotros a mano.

Por último, los componentes principales en los que podemos dividir Event Hub son los siguientes:

* Productores de eventos: es todo aquello que envía una serie de eventos al centro de eventos.
* Particiones: cada consumidor solo lee una parte de la secuencia del mensaje.
* Grupos de consumidores: nos da una vista general de todo el centro de eventos.

Los grupos de consumidores nos permiten consumir aplicaciones y que cada uno lea únicamente por lo que está interesado consumir.

* Unidades de procesamiento: Controlan toda la capacidad de procesamiento del centro de eventos.
* Receptores de eventos: es cualquier entidad que lea los datos proporcionados por evento hub.

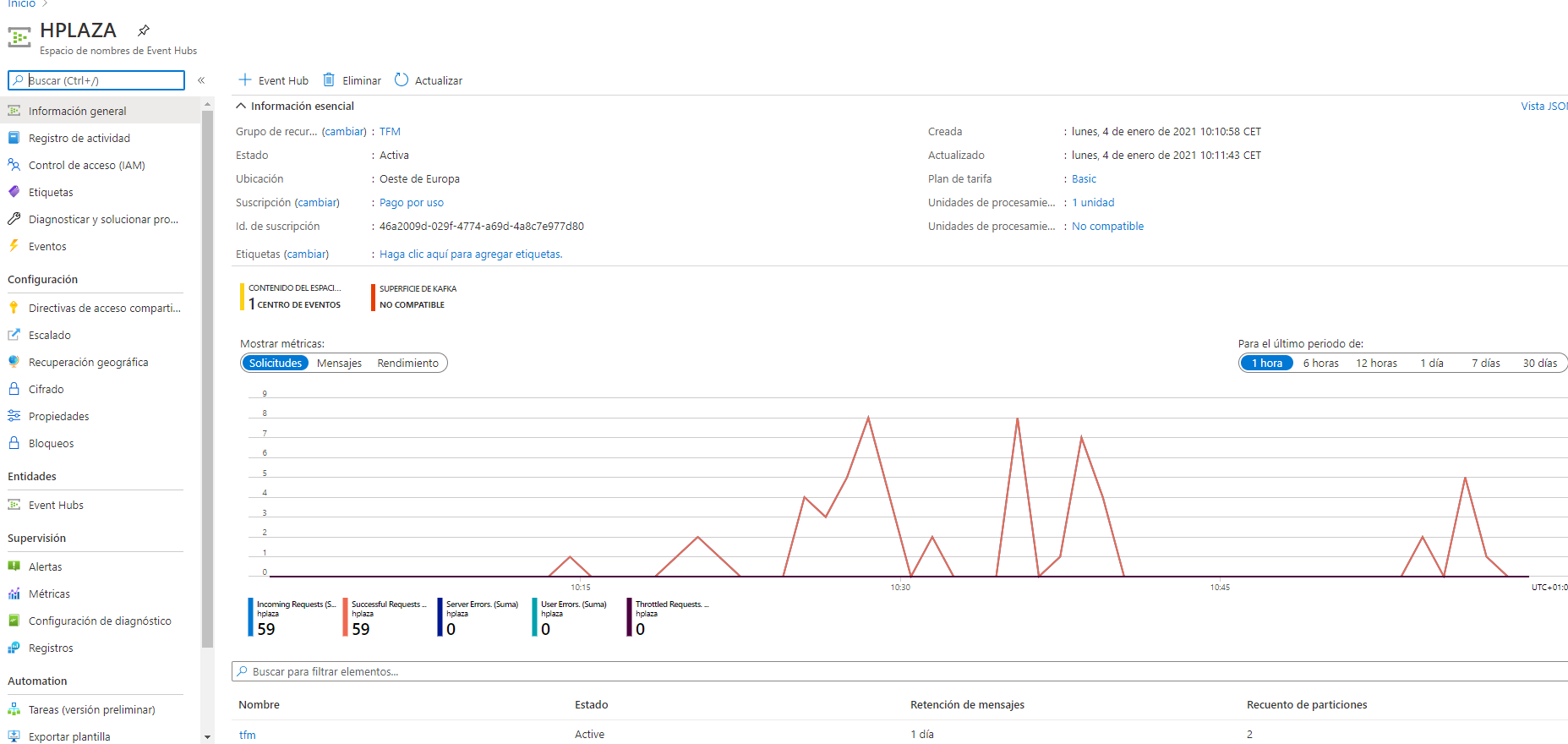
El proceso que he llevado para ingestar los datos ha sido:

Primero cree un recurso un centro de recursos de event hub para tenerlos todos reunidos y posteriormente realicé la configuración del evento de evento hub dejando las opciones predeterminadas que nos venían.

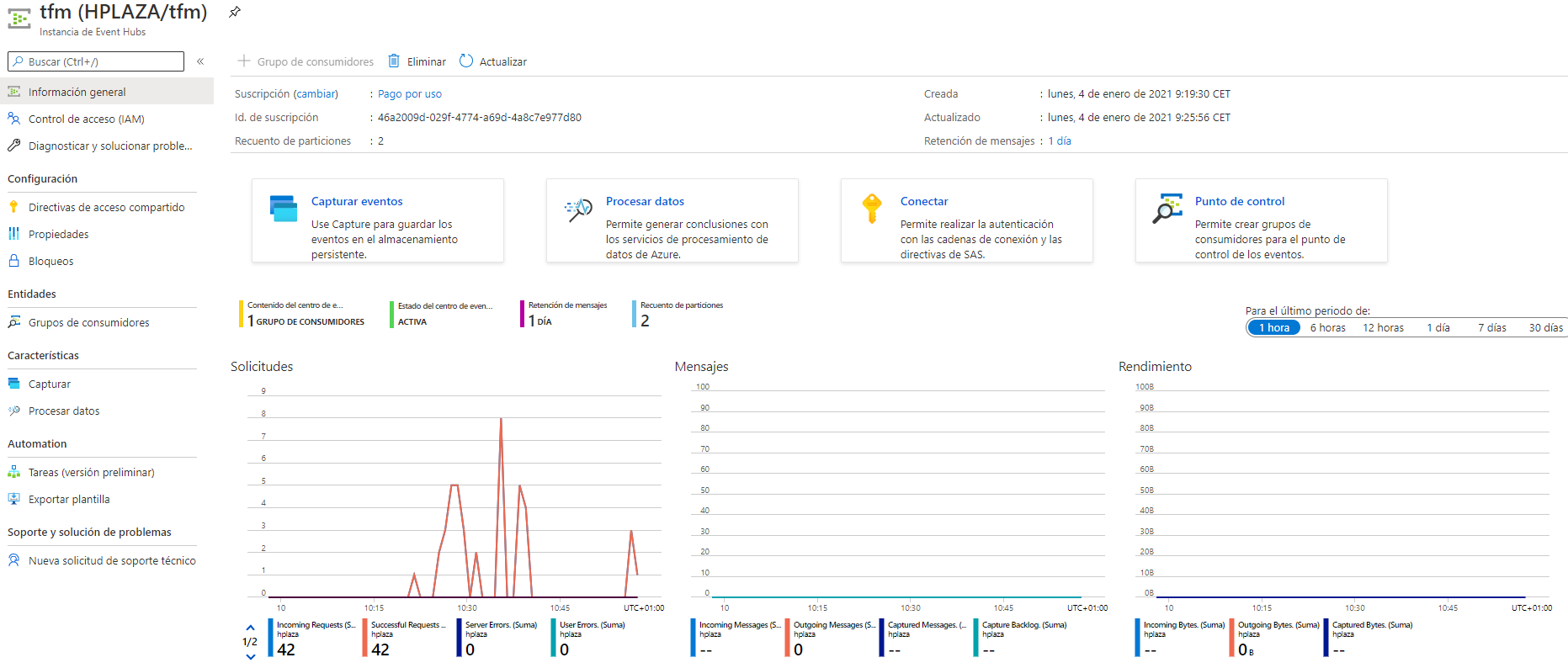
Con dos particiones y una duración de los mensajes de un día.

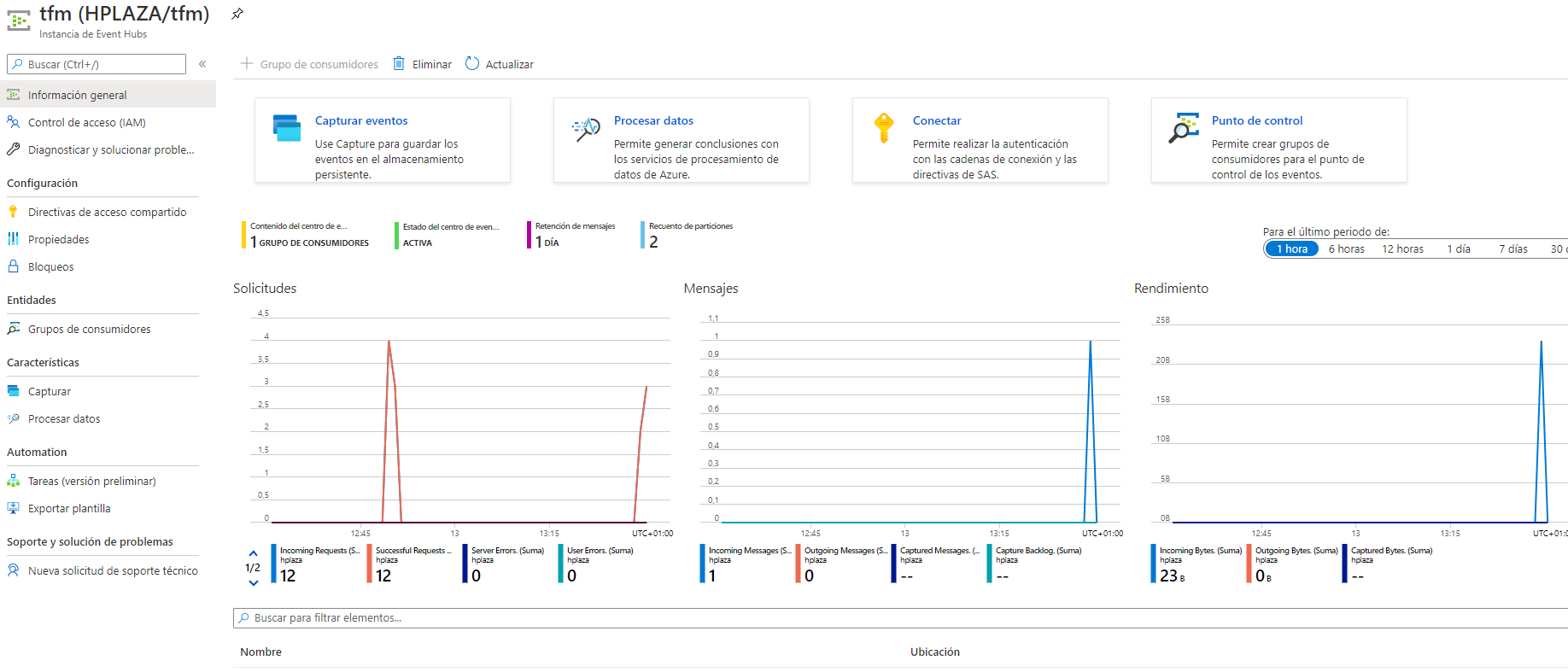
La parte importante de este paso es conseguir las claves para que después a la hora de ingestar los datos podamos ponerla y hacer la llamada desde el servicio.

Las demás partes de la arquitectura no necesitan este tipo de claves, ya que, están bajo el mismo espacio y simplemente hay que ir clicando en cada uno de los compones elegidos para ir encadenándolos.



En esta fotografía se puede ver como se ha creado el space evento hub y su nombre:





Los eventos con la información del cliente vendrán en formato CSV y los conectaremos con la siguiente pieza del puzle.

Los datos que hemos ingestado tienen el siguiente aspecto:

|  |
| --- |
| ID\_CLIENTE,"N1\_DESC\_CORTA","ANTIGUEDAD","SEGMENTO\_ESTRATEGICO","SW\_CLIENTE\_PREFERENTE","SW\_REEMBOLSO","DIGITAL","SW\_RETENIBLE","SW\_ESTRATEGICO","EDAD","PRIMA","PORC\_COASEGURO","TIPO\_SOCIO","TIEMPO","INTERACCIONES","TMO","NOCITA",”CORREO” |
| 7230814.00,"CITA","2.61","BRONCE","N","N","0.00","S","","40.00","498.60","1.00","Pymes","192.00","1.00","192.00","0.00",”user2@gmail.com” |
| 3881707.00,"CITA","36.78","BRONCE","N","N","0.00","S","","37.00","1235.52","1.00","Particular","732.00","2.00","366.00","1.00",”user1@gmail.com” |

Estos datos los hemos ingestado mediante la herramienta visual studio con el formato anteriormente mostrado.

# Azure Stream Analytics.

Azure stream analytics es un motor de procesamiento de eventos complejos y de análisis en tiempo real que esta diseñado para analizar y procesar grandes volúmenes de datos, con la posibilidad de que estos datos lleguen desde diferentes fuentes.

Los patrones y análisis que obtengamos de la herramienta nos pueden servir para lanzar alertas o analizarlos con herramientas de generación de informes o almacenarlos en base de datos.

En primer lugar, un proyecto con Azure Stream Analytics consta de una entrada, una consulta y una salida.

En nuestro caso, los datos vendrán desde Azure event hubs y el tipo de consultas que te permite hacer esta herramienta esta basado en lenguaje SQL, aunque puedes utilizar otro tipo de funciones derivadas de C+ o Java.

Cada proyecto tiene una o varias salidas y estas pueden dirigirse a distintos aplicativos.

Esta herramienta está diseñada para que sea fácilmente manejable, confiable, flexible y escalable, lo que nos permite darla un uso en trabajos de distinta índole.

Puedes crear un flujo de datos en poco tiempo tirando desde distintas fuentes y depositando los datos requeridos en distintos aplicativos, desde base de datos, hasta herramientas de visualización, todo ello dentro del universo de Azure.

Además, es una herramienta totalmente administrada, no necesitas tener hardware, ni infraestructura, ni tienes problemas con actualizaciones o cambios de versión.

La aplicación es barata, ya que, no hay costes iniciales y solo se paga por el uso que le des, no necesita aprovisionamiento de clústeres y se puede escalar de manera automática con la necesidad del negocio.

Por otro lado, Azure Stream Analytics nos permite estar seguros en cuanto a la confiabilidad que vamos a tener en los datos que recibimos, ya que, garantiza el procesamiento de los eventos al exactamente una vez y la entrega de los eventos al menos una vez, por lo tanto, siempre tenemos la seguridad de que los eventos nunca se pierden.

Además, la propia herramienta lleva consigo una serie de medidas que permitirían solucionar una perdida de evento si se produjera algún fallo en el servicio, y por supuesto tiene herramientas de monitoreo para ver el estado actual del servicio.

En cuanto a la seguridad que la herramienta tiene implementada podemos decir que todas las comunicaciones van cifradas de inicio a fin y no almacena ningún dato, ya que todo el procesamiento lo realiza en memoria.

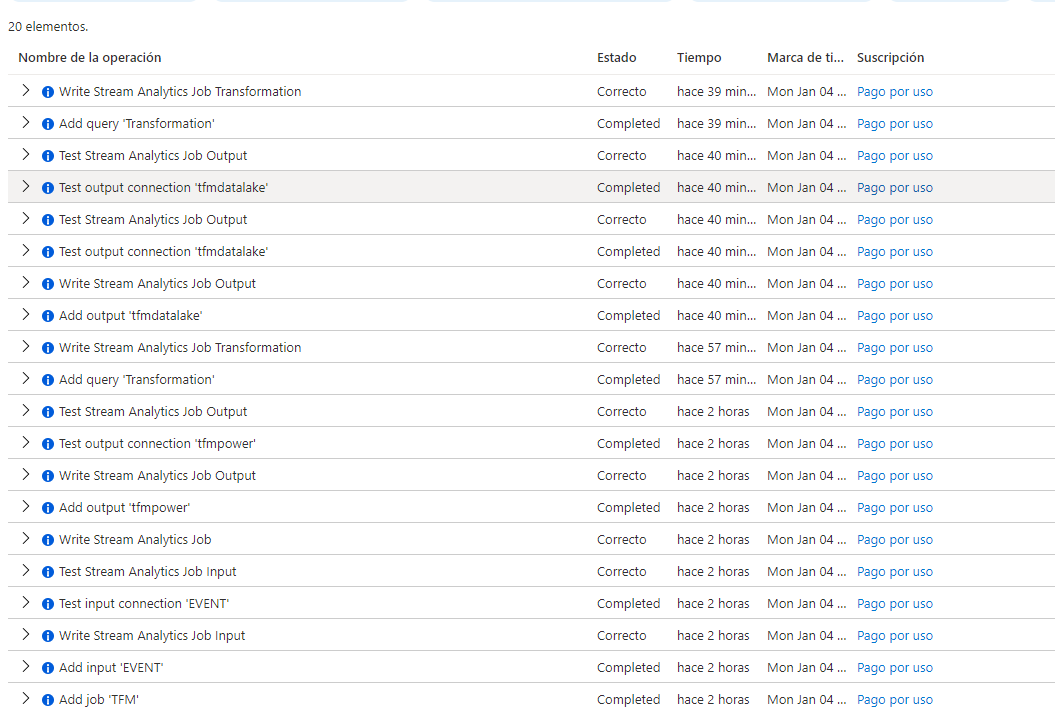
Gracias a Azure Stream Analitycs podemos procesar una gran cantidad de eventos y ofrecer resultados con baja latencia. Permite el escalado tanto horizontal, como vertical para controlar de manera correcta el procesamiento de grandes cantidades de datos y en tiempo real.

Por último, podemos señalar que la aplicación consigue un rendimiento más alto gracias a la creación de particiones lo que consigue que el procesamiento de datos complejos y en tiempo real se pueda llevar a cabo de manera paralela para conseguir una eficiencia muy importante.

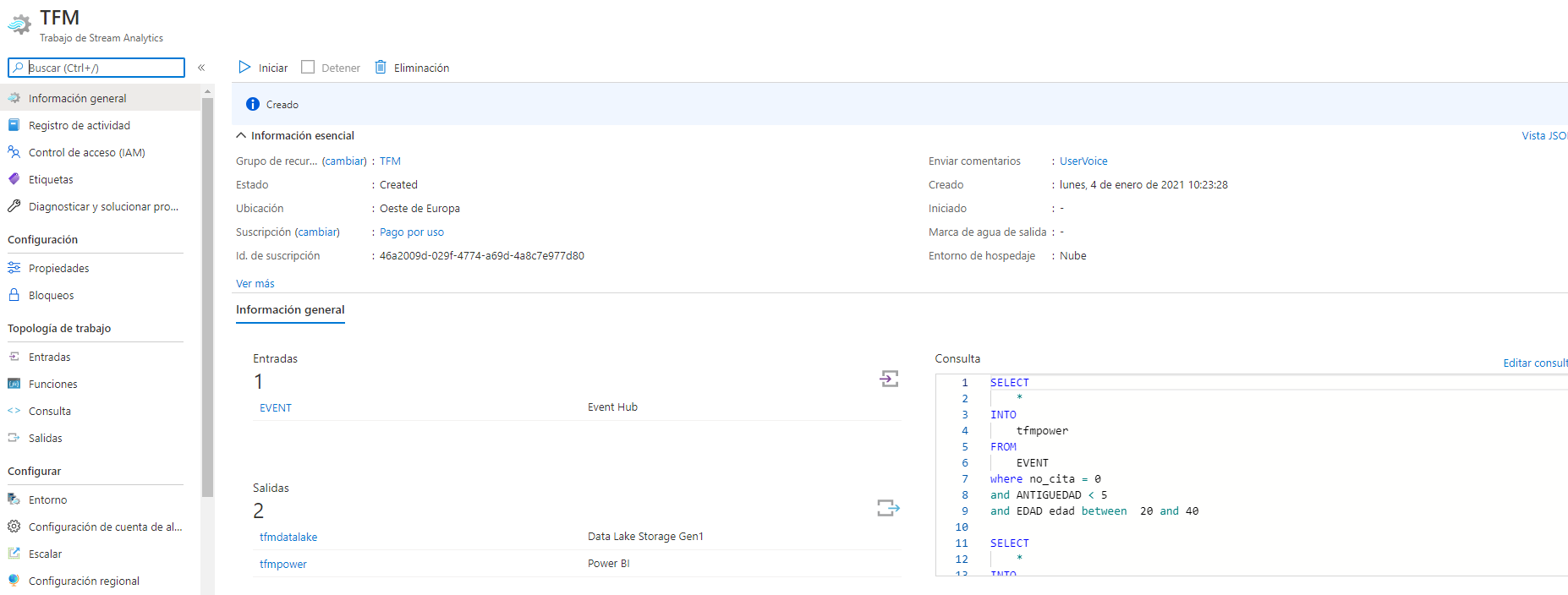
En esta ocasión, no utilizaremos los clusters dedicados que nos pone a disposición Azure, sino que lo haremos a través de un trabajo en la aplicación, pero habría la posibilidad de una vez creado el área de trabajo llevar el proyecto a estos clústeres de manera sencilla.

En cuanto a nuestro proceso, tendremos como punto de partida el punto de acceso de event hub que hemos creado anteriormente, estos datos nos llegaran cada 15 min y los ingestaremos para después quedarnos en primer lugar con los posibles clientes digitales y tras esto otra salida también en stream analitycs que nos mueva todos los datos al datalake para tenerlos por si necesitáramos analizar más adelante los datos en profundidad.

Stream analitycs te da de una forma muy visual que realmente el proyecto funciona y te va dando el ok a cada paso, una cosa que para mi me ha resultado muy útil al ser la primera vez que usaba una tecnología así.



Las entradas y salidas de la aplicación las muestro aquí abajo:



Se puede ver el código que hemos utilizado a la hora de filtrar los datos y la entrada y la salida del proyecto.

De esta manera ya tenemos configurado la parte de stream analitycs para la creación de los dashboards de seguimiento de actividad.

Además, le incorporamos la opción que nos muestre la hora en la que se cargan los datos para tener una idea del volumen general con el formato fecha/dia/hora. Esta opción viene dada en las configuraciones de stream analitycs.

# Azure functions.

Azure functions nos permite desencadenar una determinada acción desde stream analytics lo que nos da la posibilidad a su vez de conectar esta herramienta a Notifications hubs para mediante el tratamiento hecho en stream analytics podamos conseguir en nuestro caso la notificación deseada.

Pero durante la realización del procedimiento no pude realizar mi propósito y cambié de idea a la hora de dirigirme a los clientes.

En lugar de utilizar notification hub para para aplicar la notificación push, decidí utilizar azure Function para enviar directamente un correo a las direcciones indicadas.

De esta manera les comunicamos tras su llamada que tienen disponible otros servicios donde pueden realizar sus gestiones de manera más ágil y sin necesidad de llamarnos.

Creamos una salida con los datos y posteriormente cargamos la función hecha en azure.

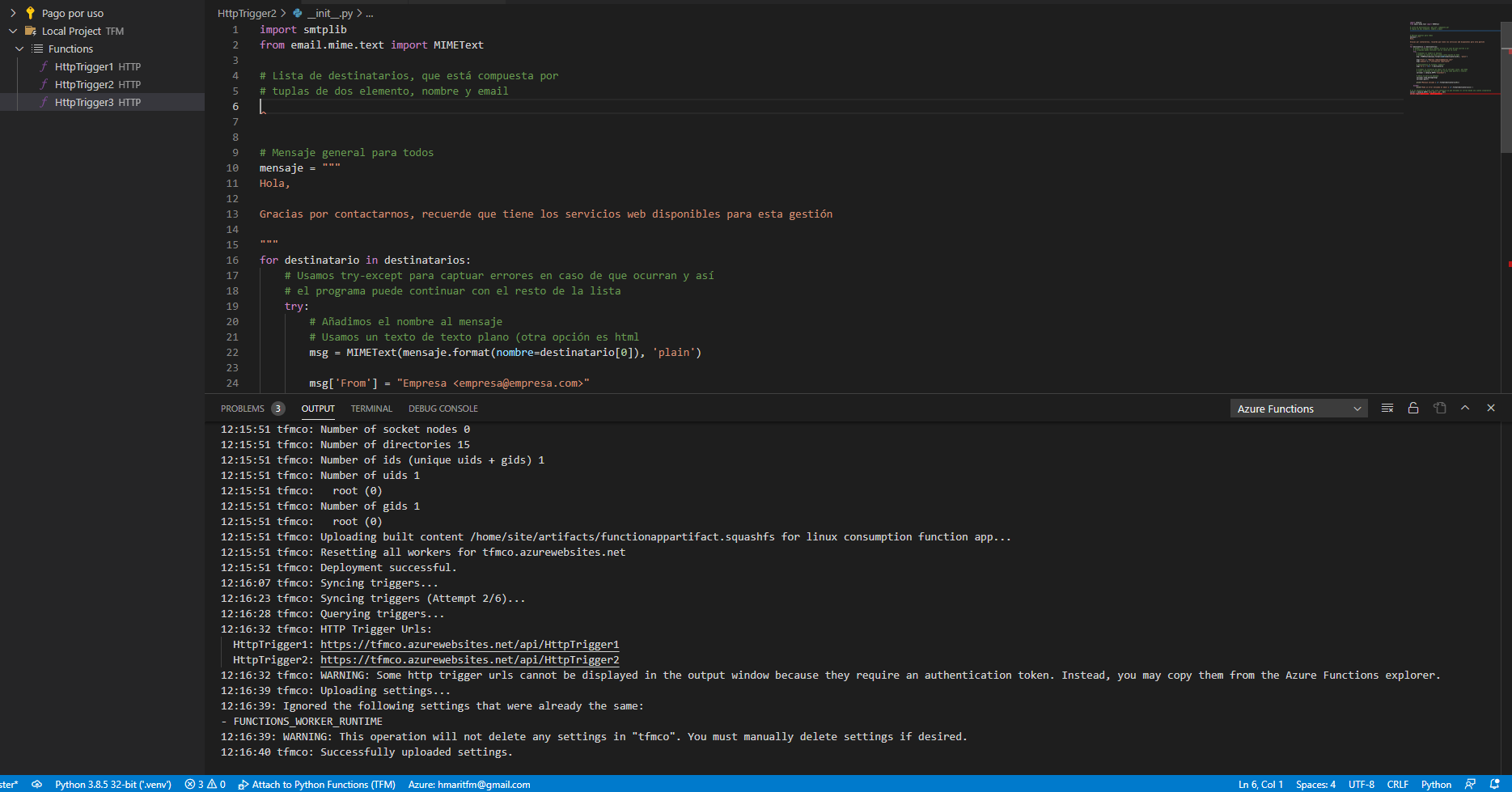
Azure functions es una solución sin servidor que nos permite realizar una serie de funciones sin necesidad de tener infraestructura ni demasiado conocimiento en código.

El código se puede implementar en distintos lenguajes, yo usé Python para diseñarlo a través de visual studio.

Azure Functions permite implementar la lógica del sistema en bloques de código fácilmente disponibles. Estos bloques de código se denominan "funciones". Se pueden ejecutar distintas funciones cada vez que necesite responder a eventos críticos.

En segundo lugar, a medida que aumentan las solicitudes, Azure Functions satisface la demanda con tantos recursos e instancias de función como se necesiten, pero solo cuando sea necesario. A medida que disminuyan las solicitudes, todos los recursos e instancias de la aplicación adicionales se descartarán automáticamente.

Gracias a esta aplicación podemos desencadenar distintos procedimientos que nos dará como resultados las notificaciones a los clientes elegidos.



# Azure Notification Hubs.

Este aplicativo nos proporciona un motor de inserción de escalabilidad horizontal que nos permite enviar notificaciones a cualquier plataforma (Android, ios, etc.) desde un entorno local o en la nube.

Las notificaciones que utiliza esta herramienta son las llamadas notificaciones push, que son mensajes que podemos realizar a través de la App en cuestión con un mensaje personalizado.

En nuestro caso utilizaremos la App de la compañía, de esta manera intentaremos cambiar el modo en el que los clientes se comunican con nosotros.

Intentaremos que la próxima vez que el cliente utilice la aplicación le salte un mensaje recordándole los servicios que puede realizar a través de ella.

Estas notificaciones push son difíciles de realizar y bastante farragosas, ya que, cada una de las notificaciones deberían ser diseñadas para cada plataforma, a este tipo de procedimiento se le llama: *sistemas de notificación de plataforma (PNS).*

Es por ello por lo que realizar programando este tipo de tareas es bastante tedioso y no hay una interfaz común para las distintas plataformas con lo que se complica más aún el trabajo.

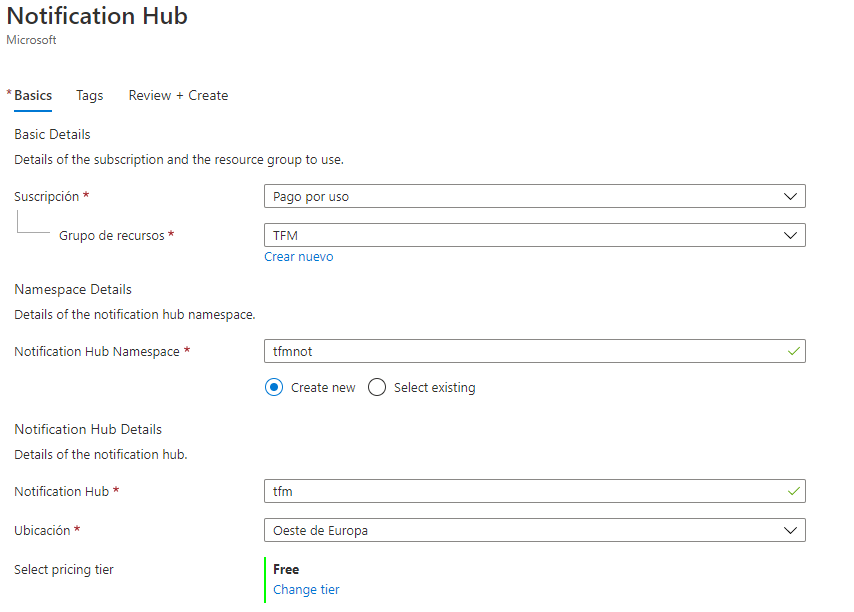
Notification hubs nos aplana y nos facilita la creación de esta notificación por los siguientes motivos:

* Multiplataforma: compatibilidad con las principales plataformas de inserción, una interfaz común que nos permite trabajar con las distintas plataformas a la vez y una administración conjunta.
* Back-ends cruzados: en nube o local y posibilidad de uso distintos lenguajes de programación.
* Conjunto completo de patrones de entrega: podemos con una sola llamada a la API enviar la notificación a distintas plataformas.

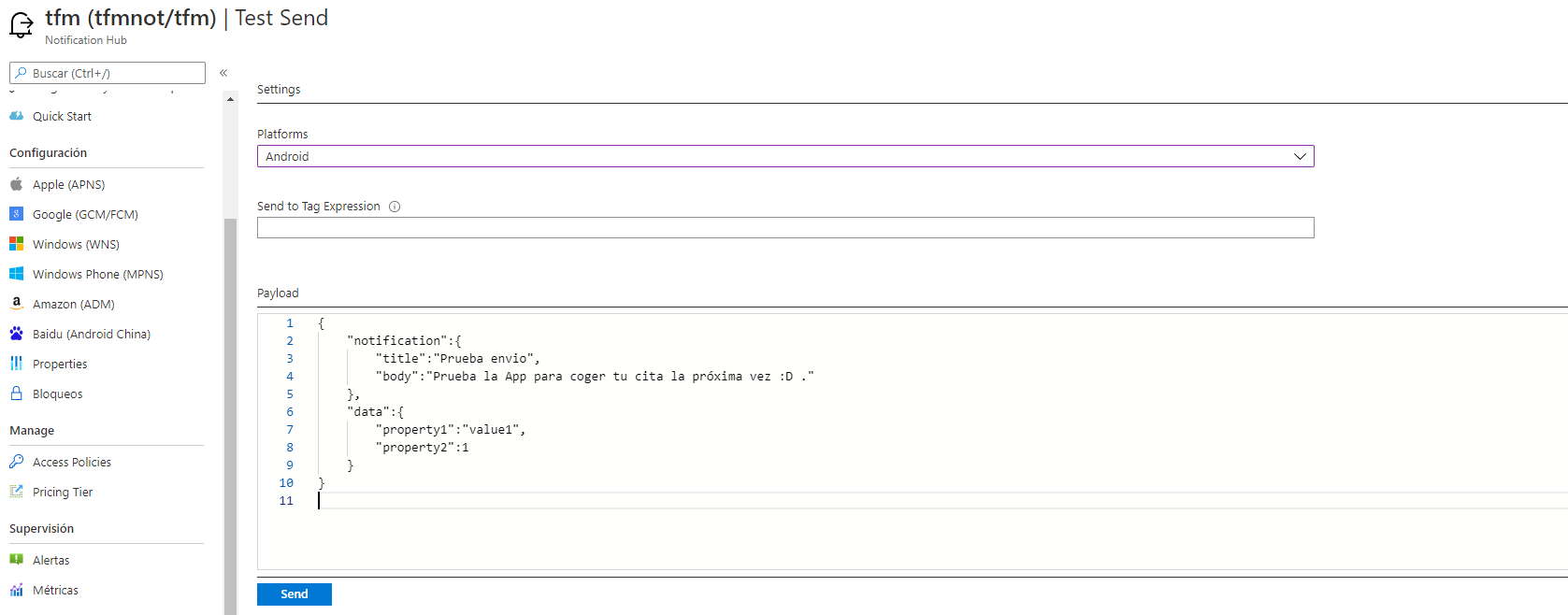
Podemos enviar distintos tipos de notificaciones, directas, indirectas, por usuario, podemos definir etiquetas para, por ejemplo, para filtrar por ubicación.

* Escalabilidad: enviar mensajes rápidos a millones de dispositivos sin tener que volver a construir la arquitectura para cada dispositivo.
* Seguridad: Firma de acceso compartido (SAS) o autenticación federada

Creamos el paso de notification y su espacio dándoles un nombre como vemos aquí abajo:



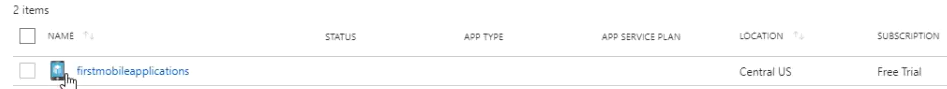
He utilizado la aplicación de Android para probar con el mensaje que aparece en la imagen siguiente:

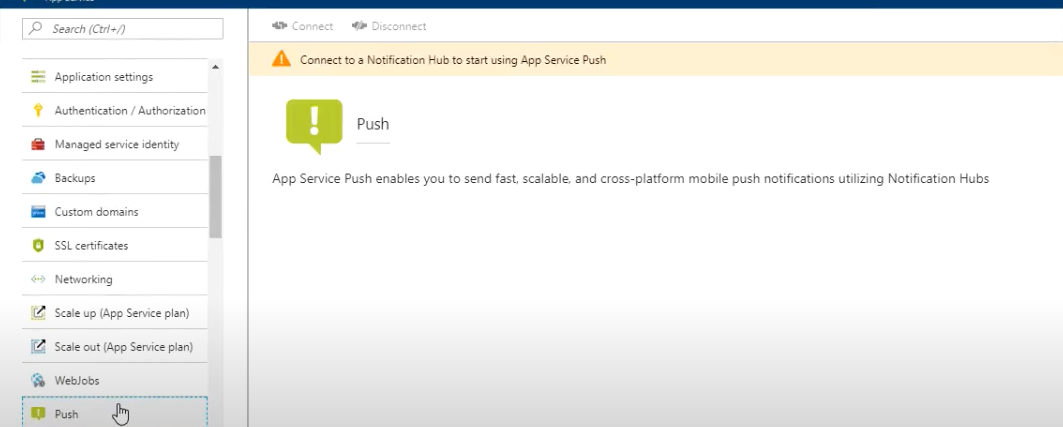


Habría que ir a la app services para dar permisos desde la aplicación corporativa.

Meteríamos los ids\_key y la contraseña y conectaríamos ambos servicios.

Actualmente no tengo la oportunidad de hacerlo, pero un ejemplo sería:

* Pincharíamos en nuestra aplicación: 
* Iriamos a la opcion push de app services tras haber pulsado en la opción anterior y conectariamos con nuestra aplicación de notification hub:



* Una vez hecho esto habría que meter cada uno de los certificados que poseen las aplicaciones de la compañía

Una vez hecho estos pasos la aplicación de notification hub tendría acceso a todas estas características y la notificación elegida saldría en los móviles de los clientes.

De esta manera tendríamos la posibilidad de anunciar a todos los clientes que entren en la app que es posible realizar sus gestiones desde ahí y no llamando por teléfono.

Sin embargo, para que las notificaciones fueran enviadas solo a aquellos clientes que nosotros hemos elegido, debemos crear una función que nos permita tenerlos identificarlos.

Para ello he estado consultando como realizarlo y en los manuales de Azure he visto una serie de códigos que podemos implementarlos para conseguirlo.

Básicamente lo que hace estos códigos es recopilar los datos de la app, es decir, los datos de cada cliente.

Y, por otro lado, otra parte del código nos da la opción de seleccionar a una serie de clientes elegidos para que les salten estas notificaciones.

Gracias a la unión de estos dos componentes podemos comunicarnos con los clientes elegidos.

Ante la dificultad que me he encontrado mientras intentaba implementar la utilización de esta herramienta, decidí utilizar una opción distinta que es envío de correos mediante azure function y el diseño del código a través de Python.

Ambas utilidades nos dan la oportunidad de enviar a los clientes el aviso.

# Azure Datalake Storage.

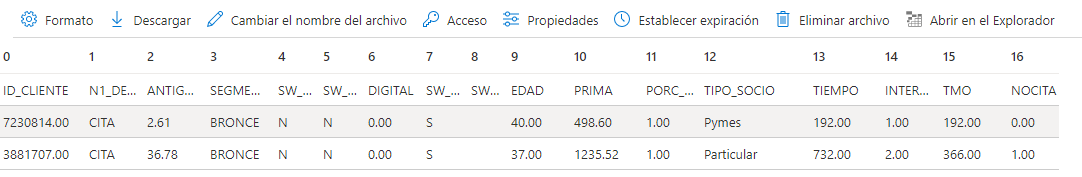
Un Data Lake, es un repositorio donde se guardan tanto datos estructurados como no estructurados que pueden saber sido, o no, procesados, con o sin esquema, para ser analizados posteriormente.

El Data Lake, permite el escalado de forma masiva, así como una escritura rápida de los datos.

Se ha elegido el Data Lake por los siguientes motivos:

* Permite recoger los datos de los diferentes dispositivos con agilidad, y sin estructura.
* En caso de que la estructura de los datos recogidos cambie, se pueden seguir almacenando sin problema.
* En caso de querer recoger otras medidas, se podrían almacenar igualmente.
* La recogida de datos es en tiempo real, por lo que los mismos aumentaran en el tiempo de forma muy importante, y el Data Lake permite escalabilidad.
* Plataforma de almacenamiento única para la ingesta, el procesamiento y la visualización que admite los marcos de análisis más comunes.
* Optimización de costes.
* Tiene un espacio de nombres jerárquicos como si fueran carpetas que nos proporciona una manera fácil en la que ordenador los datos.

El aspecto que tienen los datos tras almacenarlos en el datalake en formato csv de nuestro proyecto es el siguiente:



Las transformaciones de los datos las hemos realizado por un lado en el notebook de Python y por otro en PowerBI para poder manejar los datos a nuestro antojo.

# PowerBI.

PowerBI es una herramienta de visualización y tratamiento de datos donde podemos realizar informes que beban de distintas fuentes de datos.

En nuestro caso los orígenes serán el Datalake y Stream Analytics, el primero para realizar análisis y cruces con datos de distintos registros y el segundo para monitorear la actividad del centro y ver como se está desarrollando.

Todos estos informes se pueden poner a disposición del usuario de negocio a través de la aplicación de PowerBI dotándoles de seguridad y de posibilidades de personalización para dar flexibilidad a la persona que consulte el informe.

Además, esta herramienta cada vez intenta abarcar todo el espectro del BI y el análisis de datos, va desde el tratamiento y el modelaje de los datos a la visualización y los análisis avanzados a través de su Script de R y Python.

En mi caso al utilizar todo el ecosistema Azure, PowerBI nos viene genial, ya que, encaja cada una de las piezas y el flujo de datos se realiza sin ningún problema.

Es un software fácil de utilizar, tanto para el analista que realiza los informes, como para la persona que los consume.

Por otro lado, PowerBI es muy barato con respecto a otras herramientas del mismo estilo como Tableu o Microstrategy.

La parte de escritorio podemos disponer de ella gratuitamente y podemos trabajar en local con total libertad sin tener que desembolsar un euro.

La parte que si vale dinero es la colaborativa, la parte de la nube, donde podemos crear aplicaciones para dejar nuestros informes y que puedan ser consultados fácilmente.

Una licencia Pro vale alrededor de 10 euros al mes lo que supone un gasto muy bajo en el mundo empresarial, gracias a esta licencia podrás consultar los informes en la nube, aparte de poder crearlos.

Luego hay otro tipo de licencias llamadas premiun que tienen más funcionalidades técnicas y que nos permite adquirir lotes de licencias para abaratar más el coste.

Este tipo de licencias suelen tener un coste de alrededor de los 5.000 euros, pero obtienes determinados servicios como análisis de IA o la no necesidad de que cada usuario deba tener licencia, si no, que gracias a este tipo de licenciamiento el usuario no tendrá necesidad de tener licencia propia y habrá una común para la empresa.

En nuestro caso, al ser un departamento pequeño de alrededor de 20 personas podemos usar la licencia pro que apenas nos supondrá unos 2.400 euros de coste y que nos dará servicio para monitorear y analizar las distintas variables del negocio.

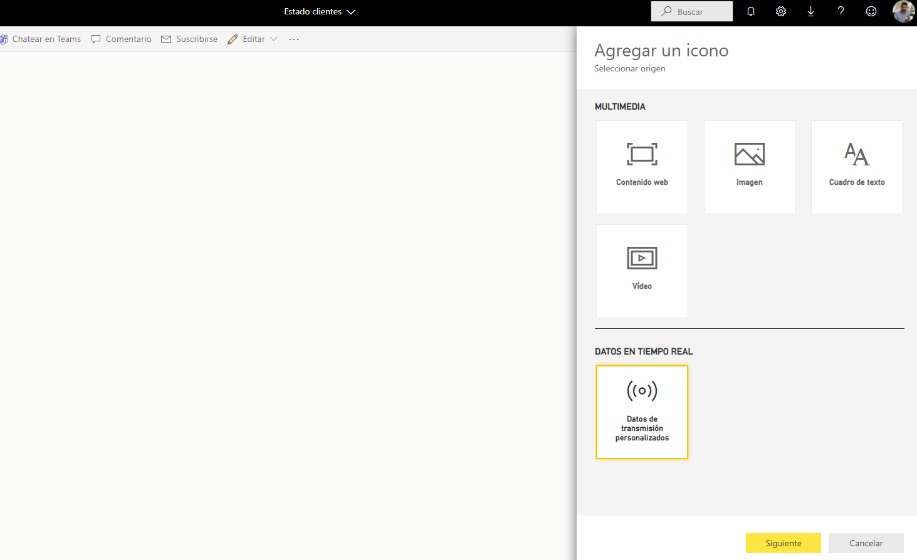
Por último, trabajaré de dos maneras:

La primera, diseñare el reporte a través de la opción de datos en streaming que posee PowerBI en su servicio.

Simplemente cogeré los datos de las llamadas de los posibles clientes digitales de las últimas horas y con una visualización de un tarjetón con el número total de clientes digitales, para que dé un vistazo rápido se vea la marcha del servicio.

En la siguiente imagen muestro el proceso de elaboración del panel.

Simplemente hay que elegir el dataset que necesitemos del servicio que hemos creado en Azure en nuestro caso el nombre es: [tfmpower](https://portal.azure.com/?quickstart=True)



Tras esto únicamente tenemos que coger los campos que necesitemos y realizamos el dataset deseado.

En este caso solo pondremos una tarjeta con el numero de clientes que han entrado bajo el criterio de posible cliente digital que hemos realizado en la prueba y la prima promedio de estos.



La segunda, el reporte podrá cruzarse con otros datos de los clientes o de las llamadas, como con una dimensión tiempo o una dimensión de población si es necesario.

Este informe lo haremos directamente en Desktop PowerBI y teniendo como origen de datos el datalake, donde a aparte de los datos del servicio, tendremos otros que puedan enriquecer el informe.

PowerBI tiene un conector que te permite sin problemas conectarte a la carpeta del datalake que necesites a través de la url y una vez conectado empiezas a manejar los datos a tu antojo dentro de la herramienta.

En un primer momento jugaremos con los datos de antigüedad y primas y las características del cliente en cuanto a su relación con la empresa.

Seria un ejemplo de lo que se podría hacer si entráramos más en detalle de los datos.

Un ejemplo de informe que he realizado con este tipo de información sería la imagen siguiente:

Imagen que contiene Calendario

Descripción generada automáticamente

Como se puede ver, he intentado que el informe tenga 3 partes bien diferenciadas para guiar al usuario en su uso.

Lo principal es que les quede claro la información que tienen disponible y cómo manejarla.

Es por ello por lo que en la parte de arriba se encuentra el nombre del informe con los filtros disponibles en él, de esta manera los usuarios tienen reunidos en la primera parte la información con la que pueden jugar en el informe.

En la segunda parte del informe he puesto las medidas fundamentales, para que de un simple vistazo te puedas hacer una idea de la pinta que tienen, en este caso, los clientes tratados en el dataset.

De un simple vistazo puedes percibir los datos fundamentales.

Y, por último, en la tercera parte he colocado las visualizaciones que profundizan más en los datos.

He realizado un clustering, un estudio de promedios y luego un grafico de barras y líneas que ilustra muy bien los datos que queremos mostrar.

Por lo tanto, es un informe que de un simple vistazo ya puedes empezar a sacar conclusiones de él.

4 clusters por antigüedad y prima, predominancia de cliente no digital, posible falta de información en el segmento empleados, etc.

Estos datos provienen directamente del datalake que a su vez vienen desde Stream Analitycs.

De esta manera tenemos cubiertas las necesidades de un análisis más pormenorizado, así como, la utilización del near real time con una visualización como ejemplo para mostrar la facilidad con la que PowerBI y stream analytics se pueden unir para realizar un informe en real time.

# Análisis de los datos con Python

En primer lugar, es necesario explicar el dataset elegido para el entrenamiento del modelo.

Hay que recordar que queremos tener claras aquellas variables que son importantes para que un determinado cliente deje de usar los servicios de VOZ y se pase a la utilización de medios digitales.

En este primer análisis hemos elegido aquellos clientes que quieren obtener una cita con alguno de los especialistas.

Las variables elegidas son las siguientes.

* ANTIGUEDAD 🡪 Número de años que lleva el cliente en la empresa
* SEGMENTO\_ESTRATEGICO 🡪 Tipo de cliente según la rentabilidad que produce a la empresa.
* SW\_CLIENTE\_PREFERENTE 🡪 indica si el cliente es preferente, es decir, si tiene servicios especiales, como, por ejemplo, un gestor personal.
* SW\_REEMBOLSO🡪Indica si tiene una póliza con servicios de reembolso
* DIGITAL 🡪Nos muestra si ha utilizado un canal digital.
* SW\_RETENIBLE 🡪 Indica si el cliente es retenible en caso de solicitar una baja. Viene determinado por el departamento de bajas de la empresa. “S” o “N”.
* SW\_ESTRATEGICO🡪 Nos proporciona información sobre el origen del cliente. Si proviene de un cliente estratégico o no. “S” o null.
* EDAD🡪 Edad actual del cliente.
* PRIMA 🡪 Pago que realiza el cliente de manera anual.
* PORC\_COASEGURO 🡪 Nos dice si el cliente es totalmente de la empresa o es compartido con otra empresa.
* TIPO\_SOCIO🡪 1-Particular, cuando el valor del campo id\_colectivo es 0

2-Colectivo Abierto

3-Colectivo Cerrado

5-Pymes

Son excluyentes de mayor a menor

* TIEMPO🡪 Suma de segundos que el cliente ha estado en contacto con nosotros.
* INTERACCIONES🡪 Interacciones que ha tenido con nosotros.
* TMO🡪 Número de segundos promedio por interacción.
* NOCITA🡪 Nos dice si han contactado con nosotros por otro tema diferente a obtener una cita.

Todas las variables son relacionadas con el cliente y con el uso que hace de los servicios de atención al cliente.

Estos datos son los primeros que podemos obtener del software de llamadas y sobre los que podemos hacer el análisis en Casi tiempo real y mandar la notificación al cliente para comunicarle la opción de la utilización de los servicios digitales.

Los primeros pasos que he realizado han sido la descripción de las variables elegidas y el conocimiento de ellas.

Por un lado, quitando variables que no iban a ser utilizadas en el análisis, como el id\_cliente o el tipo de interacción elegida, en este caso únicamente estamos tratando las llamadas bajo la tipología de Citas.

Con ello limpiamos el dataset de campos no necesarios.

Tras esto miramos los datos que contienen cada uno de los campos y si es necesario formatearlos para que el modelo funcione correctamente.

En el caso del campo sw\_estrategico hemos sustituido los nulos por la letra “N” para diferenciar los que si son estratégicos de los que no y así evitar los problemas con las celdas nulas.

Con el mismo procedimiento para sustituir los nulos por otra variable, los he sustituido por la media del campo en cuestión.

Estos campos son susceptibles de cambiarlos por la media de la columna, ya que, suelen ser variables bastante poco cambiantes.

En el caso de la prima, prácticamente de hace por grupos y las diferencias entre unos y otros vienen dados por acuerdos o por edad y antigüedad, lo que supone que, al hacer la media de todo, el valor resultante no debe estar muy alejado de su valor real.

Por supuesto esta media es mucho mas real que una Prima 0.

Este método lo he aplicado en los campos:

* TMO
* TIEMPO
* PRIMA

Otros 3 campos han sido modificados, ya que, eran numéricos y necesitábamos categorizarlos.

Es el caso de:

* NOCITA
* PORC\_COASEGURO
* DIGITAL

Estos campos han sido tratados para cambiar los 1 y los 0 con los que venían por las palabras “YES” y “NO”, de esta manera podemos tratar estos campos como categoría.

Con estas modificaciones hechas previamente a la elaboración del modelo conseguimos que los datos queden limpios y listos para empezar a sacar conclusiones.

Lo siguiente que he realizado ha sido un estudio de las variables estadísticas más comunes, como la media, los percentiles, los mínimos y máximos, etc. Para ver por donde se mueven cada una de las variables.

Esto me ayudará a la hora de realizar los tramos que necesito obtener para categorizar estas variables y al final desarrollar el modelo.

Como vemos, es un dataset desbalanceado hay bastante mas clientes que no han utilizado servicios digitales que los que realmente si lo han hecho.

Este tipo de cosas suelen pasar en las creaciones de otros modelos como los de fraude que utilizan los bancos.

El siguiente paso tras observar los datos y la estructura de los campos y de la tabla es empezar a visualizar y buscar conclusiones con los datos numéricos.

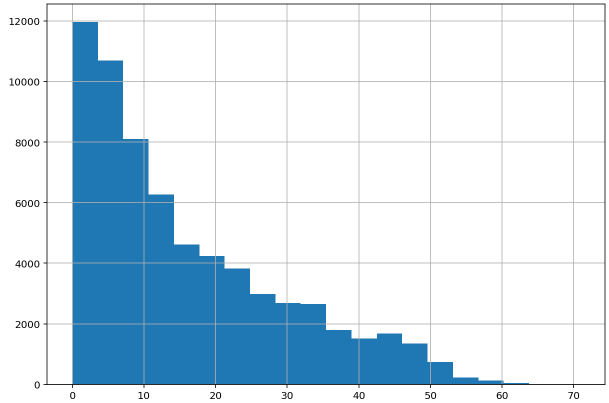
He creado una matriz de correlación que nos muestra la relación entre las variables que tenemos en el dataset, podemos ver que algunas tienen relación entre sí, pero no deja de ser lo esperado, por ejemplo, a mas edad mas prima o a mas edad mas antigüedad, ya que, los seguros priman a ello, a determinada edad el seguro ni te cubriría y es difícil que te hagas un seguro de salud con una edad avanzada.

Por lo tanto, este primer vistazo nos reafirma en nuestras creencias sobre el mercado asegurador de salud.

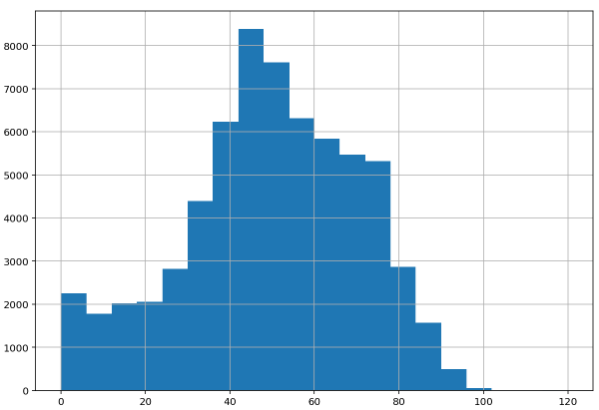
Tras esto empiezo a crear visualizaciones con estos mismos datos.

Gracias a los histogramas puedo controlar como esta repartida la muestra en cada variable. Esto es una información muy valiosa para la creación de los tramos que utilizaré más adelante.

Podemos ver que la variable antigüedad tiene una forma descendente y que tiene gran peso la gente con poca antigüedad.



En cuanto a la edad, las personas de mediana edad son el grueso de la muestra, sobre todo, entre los 40 y 50 años.

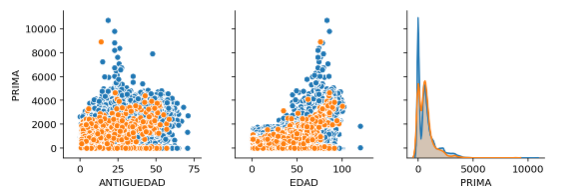


Con estos histogramas podemos ir viendo que los clientes se reparten en la mediana edad y con una antigüedad de menos de 10 años, esto nos dice que son gente que ya utiliza la tecnología y que tienen cierto conocimiento de la empresa y que podemos reforzar con ellos una posición diferente a su relación actual con la empresa, llevándolos al terreno digital y disminuyendo las comunicaciones con nosotros de manera clásica, es decir, por teléfono.

Tras ver la distribución de los clientes utilicé la función pairplot que nos permite contraponer una variable sobre otras.

En este caso si el cliente es digital o no frente a la antigüedad, prima y edad del cliente, de esta manera podemos ver como se distribuye la variable en esas características.

También hacemos algo similar con las variables categóricas, así conseguimos situar la variable Digital dentro de ellas y vemos como se distribuye.



Junto con estas visualizaciones especificamos los tramos que necesitamos para realizar el modelo.

Transformamos las variables numéricas en categóricas y las tramificamos en 3 tramos consiguiendo dividir los datos de manera que podamos modelizarlos.

Como he dicho anteriormente, me he ayudado en las visualizaciones anteriores para crearlos.

Tras esto, empiezo a modelizar separando el dataset en dos para comenzar el entrenamiento y probar diferentes algoritmos para conseguir el que da el mejor rendimiento.

Finalmente vemos que el algoritmo que mejor se comporta es el de la regresión logística.

El modelo después de balancearlo tiene un comportamiento correcto, pero no es excesivamente algo, esto ocurre porque la muestra es demasiado pequeña, viene provocado porque he tenido que trucar los datos por protección de los datos y de la operativa.

Pero, de todas formas, una vez balanceado podemos sacar una serie de conclusiones que nos pueden ayudar a tratar a estos clientes y la manera en la que podemos cambiar su relación con la empresa.

Una vez realizado el modelo, hay que sacar las palancas para que el negocio obtenga redito de él.

En nuestro caso vamos a buscar aquellas variables de los clientes que los hagan mas propensos a usar los métodos digitales.

Con estos datos que tenemos y usando stream analytics, vamos a coger dos caminos, el primero vamos a filtrar estos clientes más propensos siguiendo las variables que hemos detectado y vamos a pintarlos en un informe de PowerBI en real time, para poder hacer seguimiento y por otro lado, vamos a llevar estos clientes a Notification hub por medio de azure function de esta manera nos comunicaremos con estas personas a través de la App.

Las variables claves que nos dice el modelo que hay que tocar son las siguientes:



Estos datos tienen sentido, ya que, la gente que esta acostumbrada y tiene habito de realizar gestiones de citas a través de teléfono ven que es muy ágil y no tardan casi nada a la hora de llevar a cabo la acción no tienen intención a priori de cambiar su modus operandi.

Además, suelen ser más habituales en la con tactación con nosotros y les resulta más cómodo realizar las gestiones por teléfono.

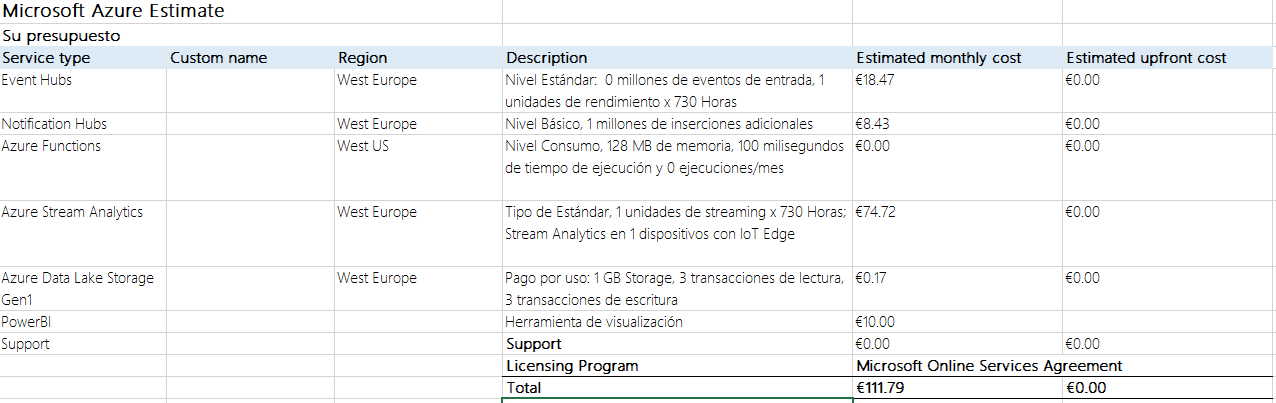
Por otro lado, las personas que quieren tratar solo el tema de citas por algo puntual parece que les resulta más cómodo la opción digital, ya que será una cosa simple de hacer y están acostumbrados a lo digital, ya que, vemos que tienen antigüedad baja por lo que se puede notar el trabajo de comunicación de la empresa y que está primando la opción digital.

Una vez sabiendo esto podemos empezar a montar el sistema para conseguir que este tráfico se vaya a la web o a la app.

# Presupuesto

He preparado gracias a la calculadora de presupuesto que nos da Azure un resumen con los principales costes mensuales que nos acarrearía la utilización de esta arquitectura.

El detalle es el siguiente:



Vemos que el montante final es cerca de los 110 euros mensuales, lo que es una cifra poco importante para una compañía multinacional y a que a poco que consiga su objetivo este dinero rentará de manera muy rápida.

Se calcula que cada llamada que se atiende en el centro de atención telefónica tiene un coste de alrededor de 3 euros, por lo tanto, con alrededor de 37 llamadas desviadas a la web nuestro coste ya estaría sufragado.

Los costes que esta solución nos ofrece son escasos, esta claro que si hubiéramos optado por una solución open source el coste tendería a cero, pero claro no contaríamos con la parte del support y con temas derivados de ella.

En definitiva, creo que esta propuesta de costes se ajusta a lo que podríamos pagar si quisiéramos realizar el proyecto.

# Conclusiones del proyecto

En primer lugar, el proyecto me ha ayudado a entender la arquitectura cloud y como usarla junto con el análisis de los datos.

La verdad que al escoger este tipo de arquitectura me pudo más las ganas por ver como se hacían que la facilidad de hacerlo por otros métodos que hemos aprendido más, pero me pareció una gran oportunidad para investigar un poquillo.

Esto me ha llevado a darme mas de una vez cabezazos contra la pared, pero creo que ha sido una experiencia muy enriquecedora.

Si que es verdad que la parte de comunicación con el cliente me ha resultado muy difícil por el desconocimiento de los lenguajes y aplicaciones.

Y al final he intentado poner las dos opciones que veía mas factibles, como era la utilización de un sistema como Notification hub y, por otro lado, el envío de correo electrónico gracias a las funciones de Azure

Al final, el proyecto me ha dado una visión completa de un proceso de desarrollo de un trabajo en el ámbito del Big data.

Quizás por mi perfil, la parte que me ha gustado más ha sido el análisis con Python, desde investigar como hacer las diferentes cosas hasta plasmarlo en el notebook y la parte la visualización de los datos en PowerBI.

Sin embargo, la arquitectura Azure me ha parecido muy interesante y como con pocos conocimientos se es capaz de montar algo útil y que sirve para mejorar el negocio.

Sobre todo, lo que más me ha llamado la atención es no tener que apenas picar código, lo cual para mí es una parte muy para tener en cuenta, ya que, únicamente controlo SQL y un poco de Python, pero más orientado al análisis de los datos.

Sin embargo, para gente más especialista si se puede acceder a los códigos que Azure proporciona para la ingestión y la unión de los diferentes componentes.

En los anexos dejaré algún ejemplo del código creado.

En líneas generales este proyecto me ha ayudado a entender el flujo del dato desde su origen hasta su destino, ya sea el cliente interno o el externo, mediante notificaciones en el caso de nuestros clientes externos y mediante la visualización de informes en caso de nuestros clientes internos.

Además, los costes que acarrea este tipo de soluciones son mínimos y es fácil probar y tener una idea de si realmente estos proyectos son rentables para las compañías o no y todo esto en un periodo corto de tiempo.

Los pasos siguientes que habría que dar es ver si realmente estas medidas podrían llevarnos a mejorar el servicio que damos, analizando el NPS, los niveles o incluso si se ha aumentado el tráfico en la web.

Durante todo el curso hemos estado trabajando las partes del proyecto y este TFM nos ha servido para unirlas y crear algo que se pueda utilizar en un entorno laboral.

Por finalizar con un balance general, me gustaría decir que ha sido un año bastante malo en general para estar pendiente de nada y que todos hemos hecho un esfuerzo, tanto profesores como alumnos, para adaptarnos, creo que la sensación general es que no hemos aprovechado lo suficiente el máster por no tener la concentración necesaria durante este tiempo desde ambas partes.

Sin embargo, tras unos meses sin clases creo que los alumnos tenemos una base interesante sobre el mundo big data y que nos va a permitir seguir avanzando en la parte que más nos ha gustado y la que mejor se nos ha dado, en mi caso, la parte de visualización y análisis es la que más he disfrutado y la que tengo ganas de seguir exprimiendo.

Este máster da unas nociones de todo un poco y son demasiadas cosas que asimilar, creo que es una introducción buena al día a día del big data, pero realmente, no se llega a entrar en profundidad en nada por falta de tiempo, ya que, todo este temario en un año es demasiado.

Un placer y feliz año.

# Bibliografía

* <https://docs.microsoft.com/es-es/azure/event-hubs/>
* <https://docs.microsoft.com/es-es/azure/stream-analytics/stream-analytics-introduction>
* <https://docs.microsoft.com/es-es/azure/stream-analytics/stream-analytics-with-azure-functions>
* <https://docs.microsoft.com/es-es/azure/notification-hubs/>
* <https://powerbi.microsoft.com/es-es/>
* <https://code.visualstudio.com/>
* <https://pythones.net/>
* <https://livebook.manning.com/book/azure-storage-streaming-and-batch-analytics/chapter-6/13>
* <https://docs.microsoft.com/es-es/azure/data-factory/load-azure-data-lake-storage-gen2>
* <https://docs.microsoft.com/es-es/windows/uwp/publish/send-push-notifications-to-your-apps-customers#getting-started-with-notifications>
* <https://docs.microsoft.com/es-es/azure/notification-hubs/notification-hubs-aspnet-backend-windows-dotnet-wns-notification>
* <https://docs.microsoft.com/es-es/azure/azure-functions/functions-run-local?tabs=windows%2Ccsharp%2Cbash>
* https://docs.microsoft.com/es-es/azure/azure-functions/create-first-function-vs-code-python

# Anexos

Todo el código que he generado tanto yo, como los componentes de la arquitectura han sido subidos a github puedes encontrarlas en esta ruta: https://github.com/hplaza1992/TFM