

▪ Proyecto Fin de Grado ▪

Ingeniería del Software

Solución para prevención de fallos y monitorización de maquinaria industrial utilizando tecnologías provistas por la nube de *Microsoft*.

---

David Prieto Montero

Junio 2016

Dirigido por: Oscar Díaz García y Cristóbal Arellano

## Agradecimientos

A Mikel Cañizo, por haberme aportado detalles sobre el diseño de los modelos predictivos y el proceso de monitorización, además de su inestimable ayuda a la hora de resolver mis dudas.

A Santi Charramendieta, por haber realizado una gran labor en el seguimiento y control del proyecto, aportando ideas y valorando el trabajo realizado.

A Oscar Díaz, por haberme dado la posibilidad de trabajar en un proyecto del cual he podido obtener nuevos conocimientos, útiles para futuros trabajos.

A Cristóbal Arellano, por haberme orientado con la gestión del proyecto.

A Arturo Orbegozo y Guillermo Lapuente, por haberme asesorado con aspectos de paralelización, dándome la posibilidad de avanzar cuando más dudas tenía.

A mi prima Myriam, por haberme orientado a hacer lo que más me gusta. Sin ella, no habría llegado hasta aquí.

A mis padres y mi hermano, por haberme apoyado, soportado y dado importantes lecciones de vida a lo largo de los últimos años.

A todos mis tíos, primos y abuelos que se han preocupado por mí, y que directa o indirectamente, han hecho que este proyecto haya sido posible.

Por último, querría agradecer a Mikel Arzak su apoyo y amistad durante los primeros años de la carrera.

## Resumen

---

Este proyecto trata el problema de la degradación en las estructuras físicas para la producción de energía renovable, más concretamente en aerogeneradores.

En el trabajo que se presenta en esta memoria, se estudia una solución existente para este tipo de problemas y se propone una adaptación para la solución mencionada, utilizando tecnologías y servicios provistos por la nube de *Microsoft*, con el fin de aportar una alternativa para las empresas que requieran soluciones basadas en la nube, y para dotar de una referencia a los investigadores de IK4-Ikerlan sobre dichas tecnologías y servicios.

**Palabras clave:** Sistema predictivo, Microsoft Azure, prevención, monitorización, Machine Learning.

## Abstract

---

This project is about the problem of renewable energies physical structures' deterioration, more specifically in wind turbines.

In the work introduced in this document, an existing solution for this kind of problems is studied and an adaptation for mentioned solution is proposed, using technologies and services provided by *Microsoft* cloud, in order to deliver an alternative for companies that require cloud based solutions, and to supply a reference for IK4-Ikerlan researchers about those technologies and services.

**Keywords:** Predictive system, Microsoft Azure, prevention, monitorization, Machine Learning.

Proyecto desarrollado en empresa

Empresa: IK4-IKERLAN

Área: Tecnologías de la Información y las Comunicaciones

Equipo: Arquitecturas *Big Data*

---

## Tabla de contenido

---

Resumen .....	3
Abstract.....	3
Tabla de contenido.....	4
Tabla de ilustraciones .....	7
1. Introducción.....	10
1.1. Objetivos.....	10
1.2. Antecedentes .....	10
1.3. Descripción General .....	11
2. Aerogeneradores y energías renovables .....	12
2.1. ¿Qué es un aerogenerador?.....	12
2.2. Problemática general .....	14
3. Diseño de la solución existente .....	16
3.1. Diseño del sistema predictivo.....	16
3.2. Diseño de la monitorización .....	20
4. Motivación del proyecto.....	22
5. Tecnologías de <i>Microsoft</i> .....	23
5.1. <i>Microsoft Azure</i> .....	23
5.1.1. ¿Qué es <i>Microsoft Azure</i> ? .....	23
5.1.2. <i>Azure Event Hubs</i> .....	24
5.1.2.1. Particiones .....	25
5.1.2.2. Unidades de procesamiento (Throughput Units, TU) .....	26
5.1.2.3. Publicadores de eventos.....	26
5.1.2.4. Consumidores de eventos .....	30
5.1.2.5. Desplazamiento ( <i>Offset</i> ) .....	31
5.1.2.6. Puntos de control ( <i>Checkpointing</i> ) .....	32
5.1.3. <i>Azure Stream Analytics</i> .....	33
5.1.3.1. Fuentes de datos ( <i>Input</i> ) .....	34
5.1.3.2. Receptores de resultados ( <i>Output</i> ).....	35
5.1.3.3. Consulta de transformación .....	36
5.1.4. <i>Machine Learning</i> .....	38
5.1.5. <i>Azure Blob Storage</i> .....	45
5.1.6. Otras tecnologías .....	46
5.1.6.1. <i>Azure Service Bus</i> .....	46



5.1.6.2. <i>OpenSSL</i> .....	47
5.1.6.3. API de Azure Service Bus.....	50
5.1.6.4. API de Azure Stream Analytics.....	50
5.1.6.5. API de Power BI.....	51
5.1.6.6. DocumentDB y Azure SQL.....	52
5.2. <i>Office 365</i> .....	53
5.2.1. ¿Qué es <i>Office 365</i> ?.....	53
5.2.2. <i>Power BI</i> .....	53
5.3. Relación entre <i>Microsoft Azure</i> y <i>Office 365</i> .....	57
6. Solución utilizando tecnologías de <i>Microsoft</i> .....	59
6.1. Descripción de una solución al problema inicial.....	59
6.2. Objetivo de las tecnologías utilizadas.....	59
6.2.1. Aplicación emisora de datos (simulador de un aerogenerador).....	59
6.2.2. <i>Azure Event Hubs</i> (ingestión de la información).....	60
6.2.3. <i>Azure Stream Analytics</i> (análisis y gestión de la información).....	60
6.2.4. <i>Machine Learning</i> (creación y uso del sistema predictivo).....	60
6.2.5. <i>Azure Blob Storage</i> (persistencia de información de control y preprocesado).....	60
6.2.6. <i>Power BI</i> (visualización de los datos procesados).....	60
6.3. Esquema general de la solución adoptada.....	61
6.4. Diagrama de componentes.....	62
6.5. Diagrama de secuencia.....	63
6.6. Diagrama de actividades.....	64
6.7. Requisitos no funcionales.....	65
7. Implementación de la solución.....	66
7.1. Desarrollo de la solución.....	66
7.1.1. <i>Azure Event Hubs</i> (ingestión de la información).....	66
7.1.2. <i>Machine Learning</i> (creación y uso del sistema predictivo).....	68
7.1.1.1. Fase 1a: Rellenar los minutos de las alarmas generadas (experimento no predictivo).....	70
7.1.1.2. Fase 1b: Generación de los conjuntos de datos (experimento no predictivo).....	72
7.1.1.3. Fase 2: Entrenamiento de los modelos predictivos.....	76
7.1.1.4. Fase 3: Experimento de monitorización.....	79
7.1.3. <i>Azure Stream Analytics</i> (análisis y gestión de la información).....	81
7.1.4. Aplicación emisora de datos (simulador de un aerogenerador).....	84
7.1.5. <i>Power BI</i> (visualizar los datos procesados).....	88

7.2. Problemas encontrados .....	91
7.2.1. <i>Azure Event Hubs</i> (ingestión de la información).....	91
7.2.2. <i>Azure Stream Analytics</i> (análisis y gestión de la información).....	91
7.2.3. <i>Machine Learning</i> (creación y uso del sistema predictivo).....	92
7.2.4. <i>Power BI</i> (visualización de los datos procesados) .....	93
7.3. Evaluación de los requisitos no funcionales.....	95
8. Extensión: Raspberry Pi como emisora de datos .....	96
8.1. Descripción de la extensión.....	96
8.2. Implementación de la extensión .....	97
8.2.1. Disposición física de los componentes .....	97
8.2.2. Configuración realizada en la Raspberry Pi .....	99
8.2.3. Creación y enlace del IoT Hub .....	100
8.2.4. Adaptación y despliegue de la aplicación UWP .....	100
8.3. Problemas encontrados .....	101
8.3.1. Aplicación y Raspberry Pi .....	101
8.3.2. <i>Azure IoT Hubs</i> (ingestión de la información).....	102
9. Gestión del proyecto.....	103
9.1. Planificación .....	103
9.1.1. Identificación de tareas .....	103
9.1.2. Diagrama de Gantt.....	105
9.1.3. EDT .....	106
9.1.4. Gestión de riesgos.....	106
9.1.5. Gestión de comunicaciones .....	107
9.1.6. Análisis de viabilidad.....	107
9.2. Seguimiento y control.....	108
10. Conclusiones.....	111
11. Líneas futuras.....	113
12. Referencias .....	114
13. Anexo: Glosario de términos .....	117

---

## Tabla de ilustraciones

---

Ilustración 1. Aerogenerador de eje horizontal (foto extraída de <a href="http://gstrium.com">gstrium.com</a> ) ....	13
Ilustración 2. Interior de una barquilla (esquema extraído de <a href="http://opex-energy.com">opex-energy.com</a> ).....	14
Ilustración 3. Curva de degradación del aerogenerador (extraído de la tesis de Anoop Verma).....	16
Ilustración 4. Esquema general de un sistema predictivo .....	17
Ilustración 5. Generación del conjunto de datos para realizar modelos predictivos (extraído de la tesis de Anoop Verma) .....	18
Ilustración 6. Generación del conjunto de datos en t+10 (extraído de la tesis de Anoop Verma).....	18
Ilustración 7. Ejemplo de monitorización (extraído de la tesis de Anoop Verma)....	21
Ilustración 8. <i>PaaS</i> de ámbito público que compiten en el mercado a día de hoy (2016).....	23
Ilustración 9. Algunos servicios de <i>Azure</i> (imagen extraída de <a href="http://mountainss.wordpress.com">mountainss.wordpress.com</a> ).....	24
Ilustración 10. <i>Azure Service Bus</i> y los servicios que gestiona (extraído de <a href="http://buildazure.com">buildazure.com</a> ) .....	24
Ilustración 11. Ejemplo de particiones con datos en un <i>Event Hub</i> .....	25
Ilustración 12. Esquema para la publicación de información en un <i>Event Hub</i> .....	27
Ilustración 13. Publicación por partición (no recomendada) .....	28
Ilustración 14. Publicación por clave de partición.....	29
Ilustración 15. Publicación por política de publicadores .....	29
Ilustración 16. Esquema para la consumición de información en un <i>Event Hub</i> ....	31
Ilustración 17. Esquema para realizar el <i>checkpointing</i> .....	32
Ilustración 18. Esquema de recuperación de <i>offset</i> .....	32
Ilustración 19. Esquema de funcionamiento habitual de un <i>job</i> .....	33
Ilustración 20. Posibles receptores de resultados definibles para un <i>job</i> (2016) .....	35
Ilustración 21. Ejemplo de <i>Tumbling Window</i> (extraído de <i>blogs</i> de <i>Microsoft</i> ).....	37
Ilustración 22. Ejemplo de <i>Hopping Window</i> (extraído de <i>blogs</i> de <i>Microsoft</i> ).....	37
Ilustración 23. Ejemplo de <i>Sliding Window</i> (extraído de <i>blogs</i> de <i>Microsoft</i> ) .....	37
Ilustración 24. Entrenamiento de un modelo (imagen realizada por Philippe Desjardins-Proulx) .....	38
Ilustración 25. Uso y consecuente entrenamiento progresivo de un modelo (imagen realizada por Philippe Desjardins-Proulx).....	39
Ilustración 26. Proceso para crear un <i>dataset</i> de pruebas a partir de datos en un fichero local .....	39
Ilustración 27. Elementos destacables de la interfaz de desarrollo de un experimento .....	40
Ilustración 28. Módulo " <i>Project Columns</i> " (izquierda) con su acción definida (derecha) .....	40
Ilustración 29. <i>Dataset</i> entrante del módulo " <i>Project Columns</i> " .....	41
Ilustración 30. <i>Dataset</i> resultado del módulo " <i>Project Columns</i> " .....	41
Ilustración 31. <i>Feedback</i> de un experimento de ejecución satisfactoria.....	42
Ilustración 32. <i>Feedback</i> de un experimento de ejecución fallida.....	42

Ilustración 33. Breve descripción del error que ha ocasionado la parada del módulo .....	43
Ilustración 34. Proceso para desplegar un experimento como un servicio web. ....	44
Ilustración 35. Cuenta de almacenamiento (izquierda), que tiene contenedores (centro), cada uno con sus <i>blobs</i> almacenados (derecha) [extraído de <i>blogs de Microsoft</i> ] .....	45
Ilustración 36. Proceso para subir el certificado de un <i>namespace</i> a <i>Microsoft Azure</i> .....	49
Ilustración 37. Flujo para gestionar recursos de <i>Power BI</i> programáticamente (imagen extraída de <i>blogs de Microsoft</i> ) .....	51
Ilustración 38. Servicios de <i>Office 365</i> (2016) .....	53
Ilustración 39. Posibilidad de extender la prueba de 60 días de <i>Power BI Pro</i> .....	54
Ilustración 40. Resultado del ejemplo de visualización descrito.....	55
Ilustración 41. Estructura del ejemplo de visualización descrito .....	55
Ilustración 42. Aviso en la descarga de una visualización desarrollada por la comunidad .....	56
Ilustración 43. Relación entre servicios de <i>Microsoft Azure</i> y <i>Office 365</i> .....	57
Ilustración 44. Cuentas <i>Microsoft</i> necesarias para acceder a los servicios descritos	58
Ilustración 45. Esquema general de la solución adoptada .....	61
Ilustración 46. Diagrama de componentes .....	62
Ilustración 47. Diagrama de secuencia .....	63
Ilustración 48. Diagrama de actividades (preprocesado y despliegue del sistema predictivo).....	64
Ilustración 49. Proceso para crear un espacio de nombres .....	66
Ilustración 50. Proceso para crear un <i>Event Hub</i> .....	67
Ilustración 51. Proceso para añadir políticas de acceso al <i>Event Hub</i> .....	67
Ilustración 52. Consultar cadenas de conexión de un <i>Event Hub</i> .....	68
Ilustración 53. Experimento de desarrollo de la fase 1a.....	70
Ilustración 54. Datos originales (izquierda) y la sustitución realizada tras ejecutar el <i>script</i> (derecha).....	70
Ilustración 55. Comparaciones a realizar para el relleno de los minutos .....	71
Ilustración 56. Resultado de rellenar los minutos de todas las alarmas generadas ...	71
Ilustración 57. Experimento de desarrollo de la fase 1b.....	72
Ilustración 58. Pivotar la tabla por <i>timestamp</i> .....	73
Ilustración 59. De la tabla pivotada (izquierda), obtener las reglas de asociación (derecha) .....	73
Ilustración 60. Eliminación de reglas de asociación repetidas .....	74
Ilustración 61. Eliminación de subconjuntos de reglas de asociación.....	74
Ilustración 62. Ejemplo de comparación y relación entre tabla pivotada y reglas de asociación .....	75
Ilustración 63. Contenido de "DatosOperacionales.zip" .....	75
Ilustración 64. Entrenamiento del modelo predictivo t+10 .....	76
Ilustración 65. Generación del modelo t+10 entrenado.....	78
Ilustración 66. Experimento para monitorización de un aerogenerador.....	79
Ilustración 67. Manual autogenerado del servicio web. ....	80
Ilustración 68. Proceso para crear un <i>job</i> .....	81
Ilustración 69. Proceso para añadir una fuente de datos a un <i>job</i> .....	82
Ilustración 70. Proceso para añadir un receptor de resultados a un <i>job</i> .....	82
Ilustración 71. Proceso para añadir una función a un <i>job</i> .....	83

Ilustración 72. Proceso para aplicar una consulta de transformación a un <i>job</i> .....	83
Ilustración 73. Parámetros definidos en <i>App.config</i> .....	85
Ilustración 74. Implementación del método <i>ReadSASRulesOnNamespaceRoot</i> (control de acceso mediante certificados).....	86
Ilustración 75. Implementación del método <i>obtainRootManageRule</i> (obtener la regla gestora por defecto).....	86
Ilustración 76. Implementación de <i>SendMessageToEH</i> (envío de datos).....	87
Ilustración 77. <i>Dataset</i> generado automáticamente en el portal de <i>Power BI</i> .....	88
Ilustración 78. Crear una visualización a partir de un <i>dataset</i> (1/3) .....	88
Ilustración 79. Aplicar filtros a la visualización (2/3).....	89
Ilustración 80. Modificar el aspecto de la visualización y anclarla a un panel (3/3) 90	
Ilustración 81. <i>Dashboard</i> de monitorización final .....	90
Ilustración 82. Esquema general de la solución adoptada (extendido).....	97
Ilustración 83. Arquitectura de red de la solución extendida.....	98
Ilustración 84. Circuito de verificación de envío desarrollado (extraído del tutorial “Blinky”).....	99
Ilustración 85. Diagrama de Gantt .....	105
Ilustración 86. EDT.....	106
Ilustración 87. Incertidumbre del proyecto .....	107
Ilustración 88. Diagrama de Gantt con desviaciones.....	110

---

## 1. Introducción

---

Este documento describe el proceso y los resultados obtenidos del proyecto desarrollado.

### 1.1. Objetivos

---

Este proyecto tiene como objetivo analizar el problema que existe con la degradación de estructuras físicas orientadas a producción de energías renovables (más concretamente en aerogeneradores), y aportar una implementación en la nube de una solución ya existente utilizando tecnologías de *Microsoft*, que requieran empresas de este ámbito como alternativa en este tipo de servicios. Esta solución consiste en una plataforma que obtiene, analiza, procesa y muestra los resultados del procesamiento de datos obtenidos de dichas estructuras. Para ello, se busca completar los siguientes objetivos:

- Presentar y explicar cuál es la problemática de estas estructuras y como afecta al negocio subyacente, con el fin de dar una visión general de la importancia del problema.
- Presentar una breve descripción de la solución existente que corrige este problema, describiendo el desarrollo del sistema predictivo y la monitorización realizada, indicando quién es su autor original y cuál ha sido el alcance de su solución.
- Definiciones y características de las distintas tecnologías de *Microsoft* utilizadas, indicando las distintas ventajas y desventajas que ofrecen.
- Descripción de la solución aportada, utilizando tecnologías de *Microsoft*, indicando las limitaciones de la solución aportada.
- Implementación de dicha solución, indicando los problemas encontrados.
- Informar de la gestión del proyecto, explicando las actividades realizadas, cuándo han sido desarrolladas, y el esfuerzo (en tiempo) dedicado a las mismas. Se incluirán aspectos como gestión de comunicaciones y riesgos.

### 1.2. Antecedentes

---

Las energías renovables tienen como objetivo ser la alternativa que sustituya a las fuentes de energía más utilizadas en la actualidad, como el carbón, el petróleo y el uranio. Se intentan sustituir tanto por la contaminación que estas producen a la hora de realizar su transformación a energía eléctrica u otro tipo de energía, como que son recursos que tarde o temprano acabarán agotándose.

Sin embargo, uno de los mayores problemas en la producción de energía (sean renovables o no), es la degradación y consecuente parada de las estructuras físicas generadoras, las cuales son la base para que el sistema de producción de energía funcione.

Por lo tanto, es necesario corregir este problema, previniendo los posibles fallos que puedan ocasionar una discontinuidad en la producción de energía.

### 1.3. Descripción General

---

Este documento está formado por varios apartados principales:

- Aerogeneradores y energías renovables (Contextualización)
- Diseño de la solución existente (Solución aportada por el autor original)
- Motivación del proyecto
- Tecnologías de *Microsoft* (Definiciones y características)
- Adaptación de la solución
- Implementación de la adaptación
- Extensión realizada sobre la implementación
- Gestión del proyecto
- Conclusiones
- Líneas futuras
- Referencias

Además, se incluye un anexo con los conceptos que requieren de una explicación.

---

## 2. Aerogeneradores y energías renovables

---

En este apartado se describe en breves líneas el contexto en el que se orienta este proyecto.

Como bien define [Twenergy](#):

*“Las energías renovables son la alternativa más limpia para el medio ambiente. Se encuentran en la naturaleza en una cantidad ilimitada y, una vez consumidas se pueden regenerar de manera natural o artificial.”*

Existen distintos tipos de energías renovables:

- ∞ Energía hidráulica: Producida por la caída del agua. Las centrales hidroeléctricas utilizan el agua retenida en embalses o pantanos a gran altura, para que esa energía potencial acumulada, se convierta en energía eléctrica.
- ∞ Energía solar: Proporcionada por el sol en forma de radiación electromagnética. Puede derivar en energía térmica (calefacción) o fotovoltaica (electricidad).
- ∞ Energía geotérmica: Energía almacenada bajo la superficie terrestre en forma de calor, ligada a volcanes o aguas termales.
- ∞ Energía mareomotriz: Producción de energía eléctrica por el movimiento de las mareas y las corrientes marinas.
- ∞ Energía de la biomasa: Aprovechamiento de la materia orgánica animal y vegetal.
- ∞ **Energía eólica: Es la energía cinética producida por el viento. A través de los aerogeneradores o molinos de viento se aprovechan las corrientes de aire y se transforman en electricidad.**

Este proyecto se centra en el mantenimiento de los aerogeneradores que transforman la energía cinética del viento en energía eléctrica.

---

### 2.1. ¿Qué es un aerogenerador?

---

En este apartado se define qué es un aerogenerador y cuáles son sus características y su funcionamiento, con el fin de entender a partir de que aspectos se desarrolla la solución.

Un **aerogenerador** (ver **ilustración 1**) es un dispositivo que convierte la energía cinética del viento en energía eléctrica.



La vida útil de la estructura (de media) es superior a 25 años. La rápida evolución tecnológica en este ámbito ha hecho que aumente la durabilidad de estos dispositivos.

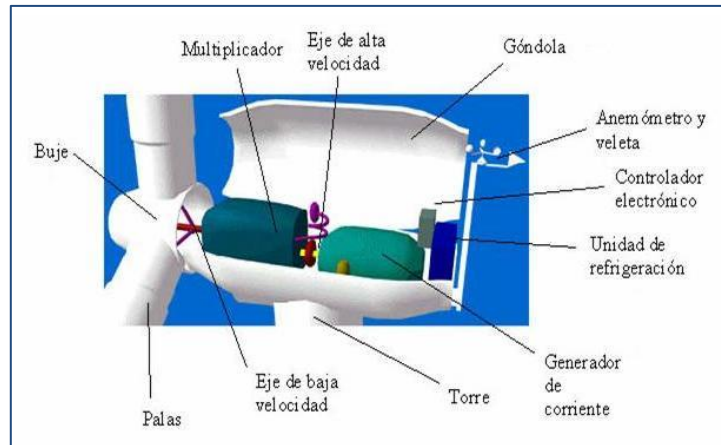


**Ilustración 1. Aerogenerador de eje horizontal (foto extraída de gstriatum.com)**

Las palas de un aerogenerador giran a una velocidad constante o bien a velocidad variable, donde la velocidad del rotor varía en función de la velocidad del viento para alcanzar una mayor eficiencia.

El aerogenerador realiza una orientación automática para aprovechar al máximo la energía cinética del viento. Utiliza los datos registrados por la veleta que incorpora en la parte superior para realizar dicha orientación.

La barquilla gira sobre una corona situada al final de la torre (ver **ilustración 2**), lo que le permite realizar giros de 360°.



**Ilustración 2. Interior de una barquilla (esquema extraído de opex-energy.com)**

Utiliza los datos registrados por el anemómetro que incorpora en la parte superior para medir la velocidad del viento. Si el viento que recibe es demasiado fuerte, las palas se colocan en bandera y el aerogenerador se frena para evitar tensiones excesivas.

El rotor hace girar un eje lento conectado a una multiplicadora que aumentara su velocidad y a su vez transfiere su energía a través de un eje rápido al generador que tiene acoplado, el cual produce la electricidad.

La energía generada es conducida a través de una línea subterránea hasta la subestación, donde se eleva su tensión para inyectarla a la red eléctrica y distribuirla a los puntos de consumo.

Todas las funciones críticas del aerogenerador están monitorizadas y se supervisan desde la subestación y el centro de control, para detectar y resolver cualquier incidencia.

Algunos de los datos que se monitorizan de los aerogeneradores son, por ejemplo, **los datos operacionales** y **las alarmas generadas** por el propio aerogenerador. Estos datos se aprovechan en la fase de uso y creación del sistema predictivo respectivamente.

## 2.2. Problemática general

---

Como bien se indica en el apartado anterior ([apartado 2.1.](#)), la vida útil de la estructura de un aerogenerador es limitada, es decir, no funciona correctamente eternamente. Existe una degradación gradual que va deteriorando dicha maquinaria a largo plazo, lo que hará que llegue a un punto en el que falle y deje de funcionar (ver **ilustración 3**).

Esto requiere que los productores de energía eólica, propietarios de estas estructuras, necesiten monitorizar el comportamiento de las mismas.

Para poder monitorizar una gran cantidad de estos dispositivos que pueden estar distribuidos por todo el mundo, una opción interesante es desarrollar una **solución basada en la nube**.

Para desarrollar este tipo de soluciones, una de las opciones es hacer uso de una plataforma ofrecida como servicio de ámbito público. Una **plataforma ofrecida como servicio** (*Platform as a Service* o **PaaS**) de ámbito público es un conjunto de herramientas y servicios, ofrecidos por un proveedor de servicios en la nube, los cuales son accesibles a través de un navegador de Internet. El uso de este tipo de plataformas para desarrollar soluciones ofrece varias ventajas:

- **No es necesaria la compra y mantenimiento de la infraestructura subyacente**, lo que posibilita el acceso a recursos “ilimitados”, pagando en función del uso que se haga de ellos. Los propietarios de esta se encargan de su mantenimiento.
- La solución es **accesible desde casi cualquier parte del mundo**, ya que lo único que requiere es acceso a Internet.
- La solución es **accesible desde cualquier tipo de dispositivo**, ofreciendo la posibilidad de aprovechar la solución tanto desde un ordenador convencional, como desde un dispositivo portátil.
- **El acceso a la solución está protegida** contra intrusiones y las características de seguridad son personalizables.
- **Flexibilidad y escalabilidad de la solución**, adaptando los recursos necesarios para la solución en **tiempo real**.
- **Reduce cualquier incidencia relacionada con el software**, ya que no es necesario instalar herramientas en equipos de trabajo personales o corporativos para desarrollar la solución.

De esta forma, la monitorización de estas estructuras puede realizarse a través de una solución escalable, accesible y que trabaja en tiempo real. Con esto, se consigue una reducción del tiempo de parada de los aerogeneradores, controlando la degradación progresiva de las estructuras, y logrando que la continuidad en la producción de energía eléctrica sea lo más estable posible.

### 3. Diseño de la solución existente

En este apartado se describe la parte del trabajo donde el equipo de IK4-Ikerlan ha diseñado y desarrollado un sistema predictivo, el cual es capaz de predecir el estado de un aerogenerador en un futuro próximo, y la fase de monitorización utilizando dicho sistema. Se puede decir que gracias a estos diseños es posible implementar la parte nuclear del proyecto; el sistema predictivo.

Cabe destacar que la última implementación que se ha realizado sobre este diseño por el equipo de IK4-Ikerlan (la solución original), utiliza tecnologías de código abierto como *Apache Spark* y *HDFS*, utilizando el lenguaje de programación *Java*, sobre una infraestructura de la que ellos disponen localmente. Utiliza **datos reales** de 20 aerogeneradores distintos, en base a los cuales se realiza el sistema predictivo y la monitorización.

#### 3.1. Diseño del sistema predictivo

Esta parte de diseño, consiste en obtener un historial de datos de las alarmas de los aerogeneradores desde un sistema de almacenamiento distribuido, procesar los datos para transformarlos y generar modelos predictivos para detectar fallos en base a esos datos.

Dado que los componentes de los aerogeneradores no se estropean de un momento a otro sino que se degradan de forma gradual (ver **ilustración 3**), el objetivo es ir monitorizando el comportamiento de los aerogeneradores, para poder predecir un fallo con suficiente antelación, y así poder solucionarlo para que el aerogenerador afectado esté el menor tiempo posible parado.

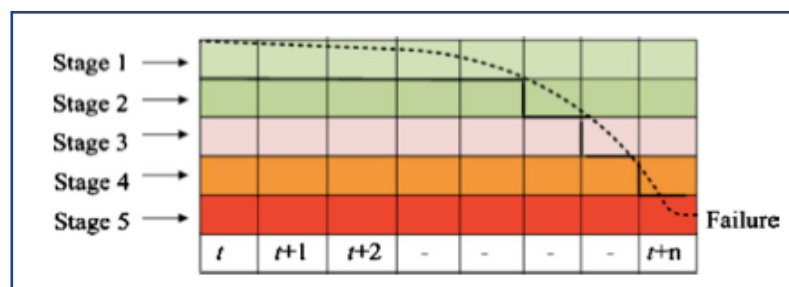
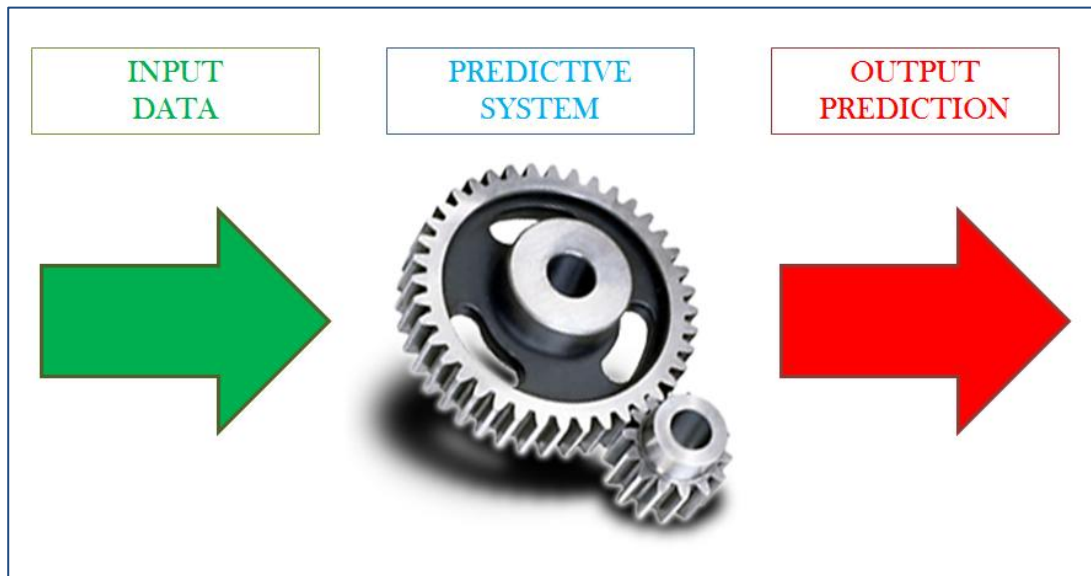


Ilustración 3. Curva de degradación del aerogenerador (extraído de la tesis de Anoop Verma)

La degradación de un aerogenerador se observa con el cambio de valor de las variables y cada vez que la función de degradación entra en un nuevo sector, se transmite un cambio de estado. En un aerogenerador se pueden crear 400 códigos de estado diferentes, donde cada uno de ellos, representa un fallo potencial.

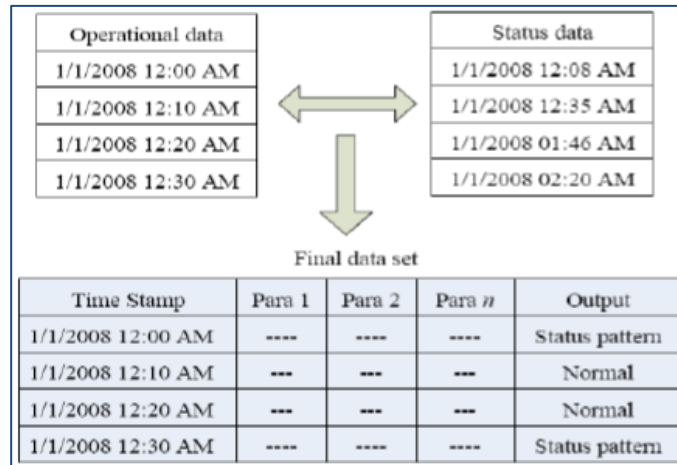
En este proyecto, los códigos de estado son referidos a las diferentes alarmas que puede crear un aerogenerador. En general, un código de estado individual no proporciona mucha información acerca del estado de salud del sistema. No obstante, una secuencia de estados (que se denomina a partir de ahora como **patrón de estados**) puede hacerlo. Por ello, la predicción de los patrones de estado es necesaria para una monitorización y mantenimiento efectivos.

Para detectar los patrones de estados, se deben obtener los datos sobre las alarmas generadas en un aerogenerador y aplicar el algoritmo de “Reglas de Asociación”. Estas reglas posteriormente servirán como posibles clases “*output*” o parámetro de salida de las predicciones, en un sistema predictivo (ver **ilustración 4**).



**Ilustración 4. Esquema general de un sistema predictivo**

Una vez identificados los patrones de estados, se deben obtener **los datos operacionales** del aerogenerador, que contienen el valor de los diferentes parámetros de rendimiento y control. Estos datos se obtienen cada diez minutos, y tienen el valor medio de cada parámetro en ese tiempo, de forma que cada dicho rango de tiempo se obtiene un registro. Para crear un modelo predictivo, es necesario relacionar los datos operacionales con los de estado (alarmas) [ver **ilustración 5**].



**Ilustración 5. Generación del conjunto de datos para realizar modelos predictivos (extraído de la tesis de Anoop Verma)**

Cabe destacar que, si ocurre un patrón de estados en una determinada fecha y hora, no es tan crítico como la repetición en el tiempo de dicho patrón, que indica una avería. Por consiguiente, para ver cuántas veces se repite un patrón en concreto, se realizan seis predicciones que corresponden a las marcas de tiempo de t+10, t+20, t+30, t+40, t+50 y t+60, donde “t” es la fecha y hora actual.

Las predicciones se realizan para prever que ocurrirá a los sesenta minutos. Para poder hacer esas seis predicciones, se necesitan seis modelos predictivos diferentes.

Para crear seis modelos predictivos, se necesitan seis conjuntos de datos diferentes, y para ello, se debe coger el conjunto de datos creado anteriormente y asignar a cada marca de tiempo de los datos operacionales la secuencia de patrones asociada a la siguiente marca de tiempo.

La cantidad de posiciones que se deben sumar para asignar otro estado a un registro de datos operacionales, es directamente proporcional a la suma de la marca de tiempo correspondiente al modelo predictivo. En otras palabras, el modelo en t+10 sumará una posición y el modelo en t+60 sumará seis posiciones, debido a que las marcas de tiempo incrementan de diez en diez (ver **ilustración 6**).

Time Stamp	Para 1	Para 2	Para n	Output
1/1/2008 12:00 AM	----	----	----	Status pattern
1/1/2008 12:10 AM	---	---	---	Normal
1/1/2008 12:20 AM	---	---	---	Normal
1/1/2008 12:30 AM	----	----	----	Status pattern

**Ilustración 6. Generación del conjunto de datos en t+10 (extraído de la tesis de Anoop Verma)**

Una vez generados los seis conjuntos de datos, se puede proceder a generar y entrenar los modelos predictivos. Para ello, por cada uno de los conjuntos de datos, se aplica un algoritmo de *Machine Learning* denominado **Random Forest**, con el cual se crea y entrena cada uno de los modelos predictivos clasificando la información entrante, y que posteriormente serán utilizados en la fase de monitorización.

Estos son, a modo de resumen, los pasos a seguir para desarrollar los modelos predictivos. Los detalles sobre el preprocesado de la información, entrenamiento y puesta en marcha de los modelos se desarrollan en el [apartado 7.1.](#)



## 3.2. Diseño de la monitorización

---

En este apartado, se explica cómo se aprovechan los modelos predictivos descritos en el apartado anterior, con el fin de realizar una monitorización previsorora del estado de la maquinaria.

La monitorización se realiza en base a estados, donde cada estado viene representado por un nivel de criticidad, por probabilidad de avería en el aerogenerador. Se plantean 4 posibles estados (ordenados por gravedad creciente):

- Normal: Indica que el funcionamiento del aerogenerador es el correcto.
- Information: Informa de que puede ocurrir alguna anomalía, pero es algo puntual.
- Warning: Indica que el aerogenerador puede tener un nivel de degradación que puede afectar a su rendimiento y que puede estar acercándose a el fin de su ciclo de vida.
- Error: Indica que el aerogenerador puede averiarse y que debe ser parado y reparado cuanto antes.

La predicción se realiza a sesenta minutos y para ello, es necesario utilizar los seis modelos predictivos descritos en el apartado anterior ([apartado 3.1.](#)). Cada vez que llegan nuevos datos operacionales (cada diez minutos), se individualizan los registros por aerogenerador y se realiza una predicción por cada modelo predictivo. Se obtienen seis predicciones diferentes por aerogenerador que, como se ha descrito en el [apartado 3.1.](#), individualmente dichas predicciones no tienen mucho peso. Sin embargo, la repetición de un mismo patrón es el que indica el nivel de criticidad.

Para calcular el nivel de criticidad, deben sumarse la cantidad de veces que se ha predicho un patrón de secuencias en concreto. Esa suma determinará el nivel de criticidad según los siguientes criterios (donde “ $n$ ” es la cantidad de veces que aparece el mismo patrón y “ $e_i$ ” el posible estado del aerogenerador en la marca de tiempo “ $i$ ” [ $i \in \{t+10, t+20, t+30, t+40, t+50, t+60\}$ ], donde “ $e$ ” puede ser predicho como “*Normal*” o como un patrón concreto):

$$\begin{aligned}
 n \geq 3 \ \&\& \ e_i = e_{i-1} = e_{i-2} = \text{pattern} \rightarrow \text{Error} \\
 2 \leq n \leq 3 \ \&\& \ e_i = e_{i-1} = \text{pattern} \rightarrow \text{Warning} \\
 1 \leq n \leq 2 &\rightarrow \text{Information} \\
 n = 0 &\rightarrow \text{Normal}
 \end{aligned}$$

Un ejemplo de monitorización del estado de un aerogenerador concreto es la **ilustración 7**:



Fecha	Hora	Marca de tiempo	Predicción realizada	Estado del aerogenerador	Intervalo de tiempo
01/04/2016	12:00:00 AM	t - 40	Patrón de estados		
01/04/2016	12:10:00 AM	t - 30	Patrón de estados		
01/04/2016	12:20:00 AM	t - 20	Normal		
01/04/2016	12:30:00 AM	t - 10	Normal		
01/04/2016	12:40:00 AM	t	Normal		
01/04/2016	12:50:00 AM	t + 10	Normal	Information	{t-40, t-30, ... t+10}
01/04/2016	01:00:00 PM	t + 20	Normal	Information	{t-30, t-20, ... t+20}
01/04/2016	01:10:00 PM	t + 30	Patrón de estados	Information	{t-20, t-10, ... t+30}
01/04/2016	01:20:00 PM	t + 40	Patrón de estados	Warning	{t-10, t, ... t+40}
01/04/2016	01:30:00 PM	t + 50	Patrón de estados	Error	{t, t+10, ... t+50}
01/04/2016	01:40:00 PM	t + 60	Patrón de estados	Error	{t+10, t+20, ... t+60}

Ilustración 7. Ejemplo de monitorización (extraído de la tesis de Anoop Verma)

Uno de los objetivos de este tipo de monitorización, como ya se ha comentado al principio de este mismo apartado, es poder realizar una monitorización preventiva, con el fin de anticipar fallos con suficiente antelación, y así que el correcto funcionamiento de los aerogeneradores monitorizados sea lo más continuo posible.

En el [apartado 7.1.](#), se describe como se implementa esta monitorización utilizando las distintas tecnologías de *Microsoft*.

---

## 4. Motivación del proyecto

---

La motivación del desarrollo de este proyecto se debe a que, en IK4-Ikerlan, aunque ya ofrece una solución basada en tecnologías de software libre, se ha encontrado con clientes potenciales que, debido a su *background* de soluciones basadas y desarrolladas con tecnologías de *Microsoft*, prefieren dar el salto a soluciones en la nube desarrolladas en base a *Microsoft Azure*.

Por lo tanto, aprovechando esa oportunidad de negocio, se ha decidido obtener una solución alternativa a la actual, buscando la equivalencia de servicios de la plataforma de procesado de datos actual de IK4-Ikerlan, en los servicios que ofrece *Microsoft Azure*. Se ha desarrollado una prueba de concepto, para así enriquecer a los investigadores de IK4-Ikerlan en esta ramificación de tecnologías, posibilitando un crecimiento de clientes potenciales.

---

## 5. Tecnologías de *Microsoft*

---

En este apartado se definen las tecnologías de *Microsoft* utilizadas, además de sus características más destacables, funcionamiento, ventajas y desventajas que aportan.

### 5.1. *Microsoft Azure*

---

En este apartado se define qué es *Microsoft Azure* y cada una de las tecnologías utilizadas sobre esa plataforma.

#### 5.1.1. ¿Qué es *Microsoft Azure*?

*Microsoft Azure* (anteriormente conocida como *Windows Azure*) es una **plataforma ofrecida como servicio** (*Platform as a Service* o ***PaaS***) de ámbito público que está alojada en los centros de datos de *Microsoft*. Ofrece un conjunto de tecnologías y servicios en la nube, las cuales facilitan una serie de servicios a los desarrolladores de aplicaciones.

En este tipo de plataformas, el anfitrión de la plataforma suministra la infraestructura de recursos y ofrece lo necesario para la construcción y puesta en marcha de aplicaciones y servicios web disponibles a través de Internet. No es necesario realizar ninguna instalación para desarrollar o realizar configuraciones sobre las mismas, lo que ofrece cierta flexibilidad para interactuar con estos sistemas desde distintos puestos de trabajo. Los múltiples servicios que proporcionan, están provisionados como una solución integral en la web.

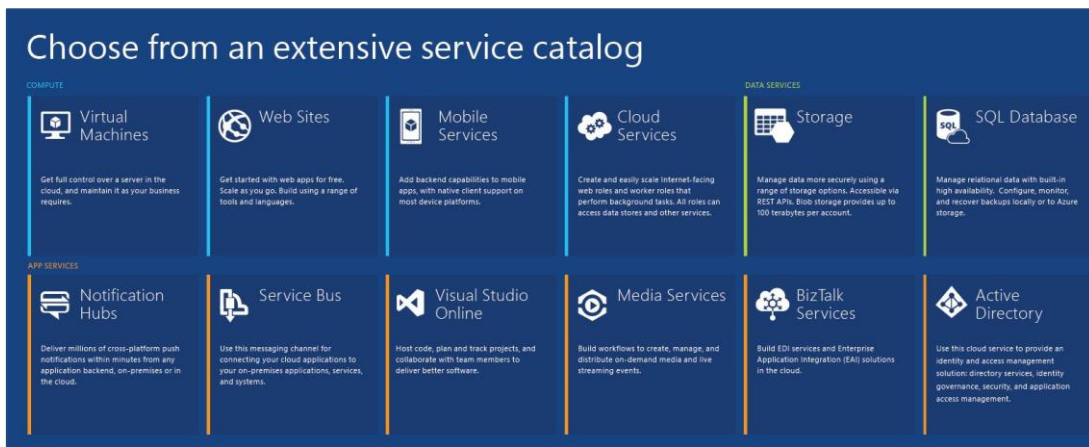
Actualmente, *Microsoft* no es la única *PaaS* de ámbito público, es decir, igual que en otros servicios o productos que ofrece, tiene una competencia en el mercado. Las compañías que con más fuerza compiten con *Microsoft* en este tipo de servicios son *Google* y *Amazon*, con sus respectivas *PaaS* de ámbito público, *Google Cloud Platform* y *Amazon Web Services* o *AWS* (ver **ilustración 8**).



Ilustración 8. *PaaS* de ámbito público que compiten en el mercado a día de hoy (2016)

*Microsoft Azure* dispone de una colección cada vez mayor de servicios integrados en la nube (análisis, procesamiento, almacenamiento, web, adaptación

para dispositivos móviles, ingesta de datos, aprendizaje automático o *Machine Learning*...) [ver **ilustración 9**]



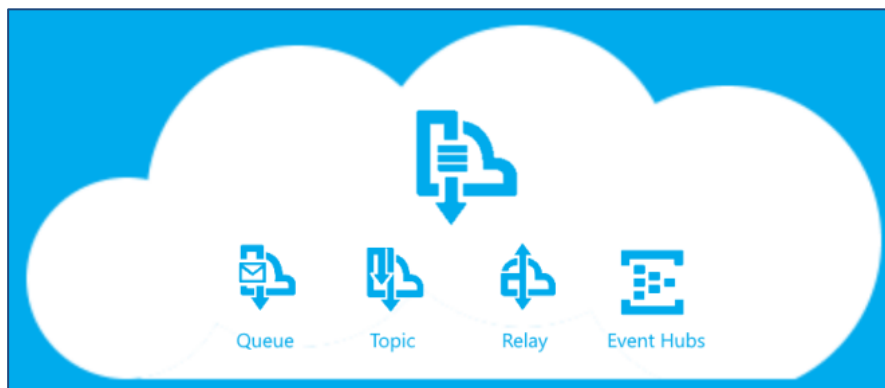
**Ilustración 9.** Algunos servicios de *Azure* (imagen extraída de [mountainss.wordpress.com](http://mountainss.wordpress.com))

En los siguientes apartados se describen varios de los servicios integrados en *Microsoft Azure* que se han utilizado para diseñar y desarrollar una solución orientada al contexto de este proyecto.

### 5.1.2. *Azure Event Hubs*

*Azure Event Hubs* (o centros de eventos de *Azure*) es un servicio de plataforma administrada que ofrece una base para el consumo de datos a gran escala en una amplia variedad de escenarios. Como escenarios de ejemplo, se pueden considerar la monitorización de tráfico de granjas de servidores web, **maquinaria industrial** o vehículos conectados.

Se encuentra dentro de otro servicio que actúa como canal de mensajería llamado *Azure Service Bus* (cuya instancia se denomina espacio de nombres o *namespace*), desde el cual se gestiona este servicio. Se habla de este servicio en el [apartado 5.1.6.](#) (ver **ilustración 10**).



**Ilustración 10.** *Azure Service Bus* y los servicios que gestiona (extraído de [buildazure.com](http://buildazure.com))

El papel que desempeña *Azure Event Hubs* es de “entrada principal”, es decir, se encarga de hacer la ingesta de los datos.

### 5.1.2.1. Particiones

Este servicio utiliza un sistema de particiones para distribuir y persistir temporalmente la información que obtiene. El comportamiento de estas particiones es de “registro de confirmación”, es decir, las aplicaciones consumidoras que utilicen este servicio tendrán la posibilidad de *confirmar* que han leído la información que han obtenido.

Las particiones se relacionan más con el paralelismo de bajada para consumo de datos que para el procesamiento de eventos, están dotadas de cierta persistencia temporal de eventos, son independientes entre ellas y crecen a ritmos distintos (ver **ilustración 11**).

La información que un *Event Hub* (instancia del servicio *Azure Event Hubs*) recibe y envía, se manipula en una estructura de datos denominada “*EventData*” (estas estructuras alimentadas con datos a partir de ahora se denominan **eventos**) y cuya arquitectura es la siguiente:

- Cuerpo: Los datos a tratar.
- Propiedades: Características opcionales que se pueden definir sobre los datos a tratar.
- Metadatos: *Offset*, número de secuencia y demás datos de control.

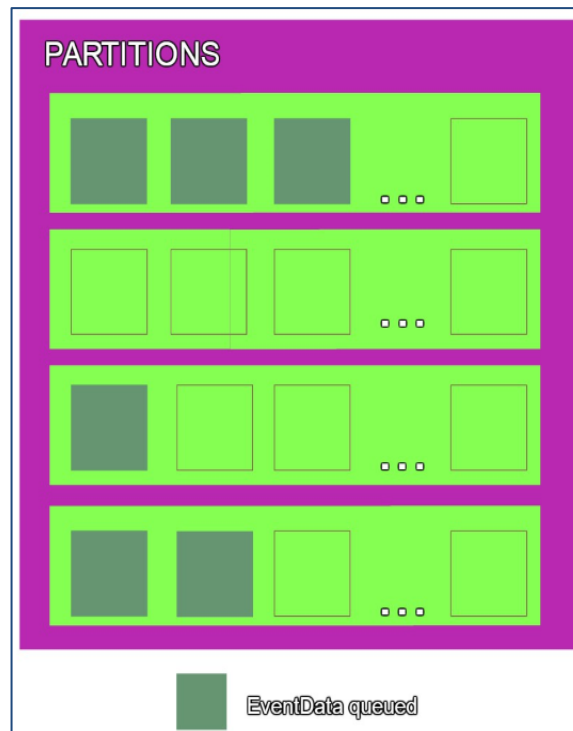


Ilustración 11. Ejemplo de particiones con datos en un *Event Hub*

Estos eventos no se pueden eliminar de las particiones. La única causa por la que dichos eventos puedan desaparecer de las mismas es por el tiempo de persistencia (o caducidad) que se defina en el *Event Hub*.

Cada partición no puede tener activo más de un consumidor simultáneo, sin embargo, puede llegar a haber más consumidores que número de particiones (dependiendo del número de particiones) y que además, todos los consumidores de forma simultánea, puedan obtener datos de las particiones que deseen. Esto, aunque parezca contradictorio con lo mencionado al principio de este párrafo, es posible gracias a los **grupos de consumidores**, los cuales se explican en un apartado posterior ([apartado 5.1.2.4. \(Consumidores de eventos\)](#)).

A pesar de ello, el número de consumidores está directamente relacionado con el número de particiones (por el paralelismo de bajada o consumo de datos). Un *Event Hub* puede llegar a tener “*n*” particiones, donde “*n*” por defecto es 4 y como máximo de 32 (además, se puede contactar con el soporte de *Microsoft* para solicitar más de 32). El número de particiones sólo se puede especificar en la creación del *Event Hub*, por lo que hay que elegir el número de particiones pensando a largo plazo.

### 5.1.2.2. Unidades de procesamiento (Throughput Units, TU)

La capacidad de procesamiento de los *Event Hub* se controla mediante estas unidades.

Las características principales de estas unidades son (por cada TU):

- 1MB/s de ingreso de datos o 1000 eventos por segundo
- 2MB/s de egreso de datos

Se gestionan a nivel de espacio de nombres. Por defecto, a la hora de crear el espacio de nombres, se configura un TU, sin embargo, se puede incrementar hasta 20 TU.

El número de TU debe ser menor o igual al número de particiones, ya que **aunque haya más TU que particiones, no mejorará el rendimiento.**

### 5.1.2.3. Publicadores de eventos

Son los que se encargan de enviar la información (estructurada en eventos) a un *Event Hub*. Cada evento puede llegar a ocupar hasta 256 KB. Hay dos formas en las cuales los publicadores pueden comunicarse con el *Event Hub*: utilizando peticiones HTTPS o mediante conexiones AMQP.

Peticiones HTTPS:

- Ventaja: No hay sobrecarga de comunicaciones inicial, lo que ofrece una comunicación rápida entre ambos puntos.

- Desventaja: Requiere de una sobrecarga SSL adicional por solicitud, es decir, el publicador se autentica en cada envío de datos.

Conexiones AMQP:

- Ventaja: Abre un *socket* bidireccional persistente y realiza la autenticación únicamente al principio de la sesión. A partir de ahí, el publicador sólo tiene que preocuparse de enviar los datos.
- Desventaja: Ocasiona grandes costes respecto a tráfico de red. Por ejemplo, si el publicador realiza la conexión pero no comunica nada.

Para publicadores frecuentes es recomendable usar las conexiones AMQP, ya que **proporcionan un ahorro en rendimiento, latencia y procesamiento**.

Además, los publicadores necesitarán obtener un *token* de firma de acceso compartido (*Shared Access Signature* o *SAS*) a partir de una política SAS con derechos de envío sobre el *Event Hub*. Tiene que definirse previo al envío de información, y es necesario para poder realizar dicho envío.

Estas firmas de acceso compartido, son el mecanismo de autenticación de los *Event Hub*. Las políticas de estas firmas se crean especificando su nombre y los derechos de uso que tienen (envío, escucha y/o gestión).

Las políticas SAS se encuentran tanto a nivel de espacio de nombres como a nivel de *Event Hub*, con lo que ofrecen derechos de uso a distintos niveles. El *token* SAS se genera a partir de una clave SAS (*hash* SHA de una dirección URL con un formato concreto), la cual se genera al crear la política SAS con sus derechos correspondientes. Con el nombre de la política y el *token*, se puede volver a generar el *hash*, y consecuentemente, se puede autenticar al publicador.

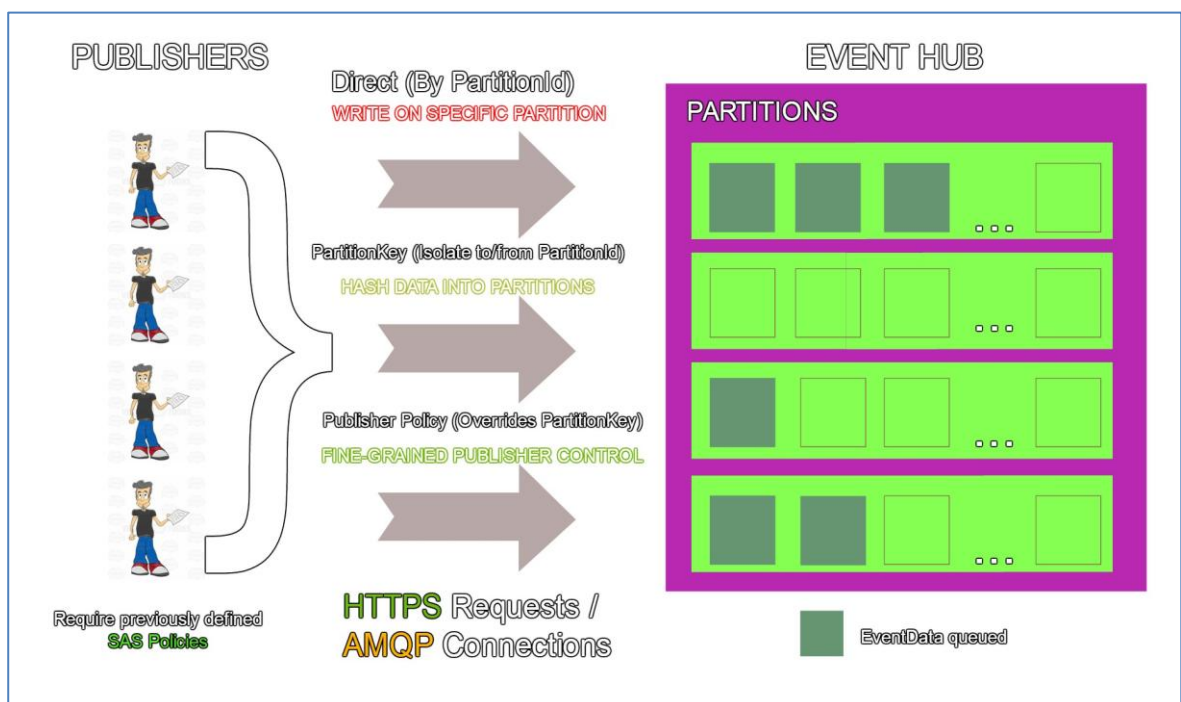
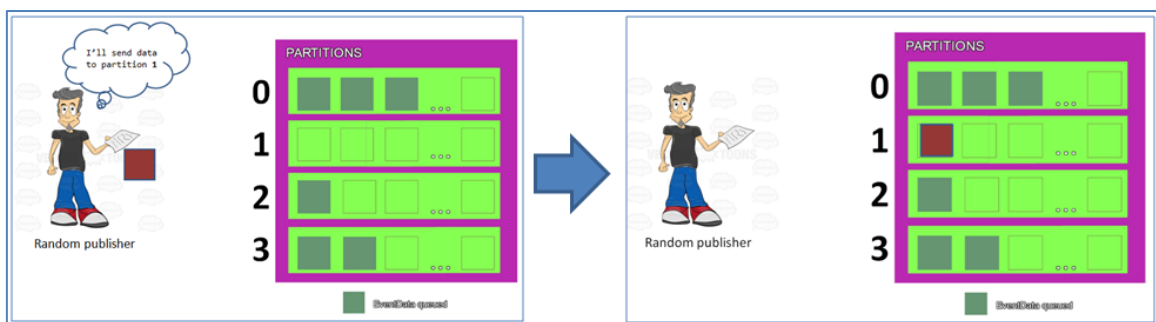


Ilustración 12. Esquema para la publicación de información en un *Event Hub*

Según la **ilustración 12**, además de las características descritas hasta el momento, existen tres formas distintas de que un publicador envíe información a un *Event Hub*:

- Por publicación directa: indicando a que partición se envían los datos; método no recomendado, por cuestiones de organización de la información.
- Por claves de partición: mapea los datos entrantes por **clave de partición** (valor único ofrecido por el publicador) en particiones específicas, con el objetivo de organizar la información.
- Por política de publicadores: similar al anterior, pero con una **capacidad de control** extra, para que sea posible gestionar los publicadores (revocar los derechos de envío, por ejemplo).

Las siguientes ilustraciones muestran de forma más gráfica el funcionamiento de estos métodos de publicación:



**Ilustración 13. Publicación por partición (no recomendada)**



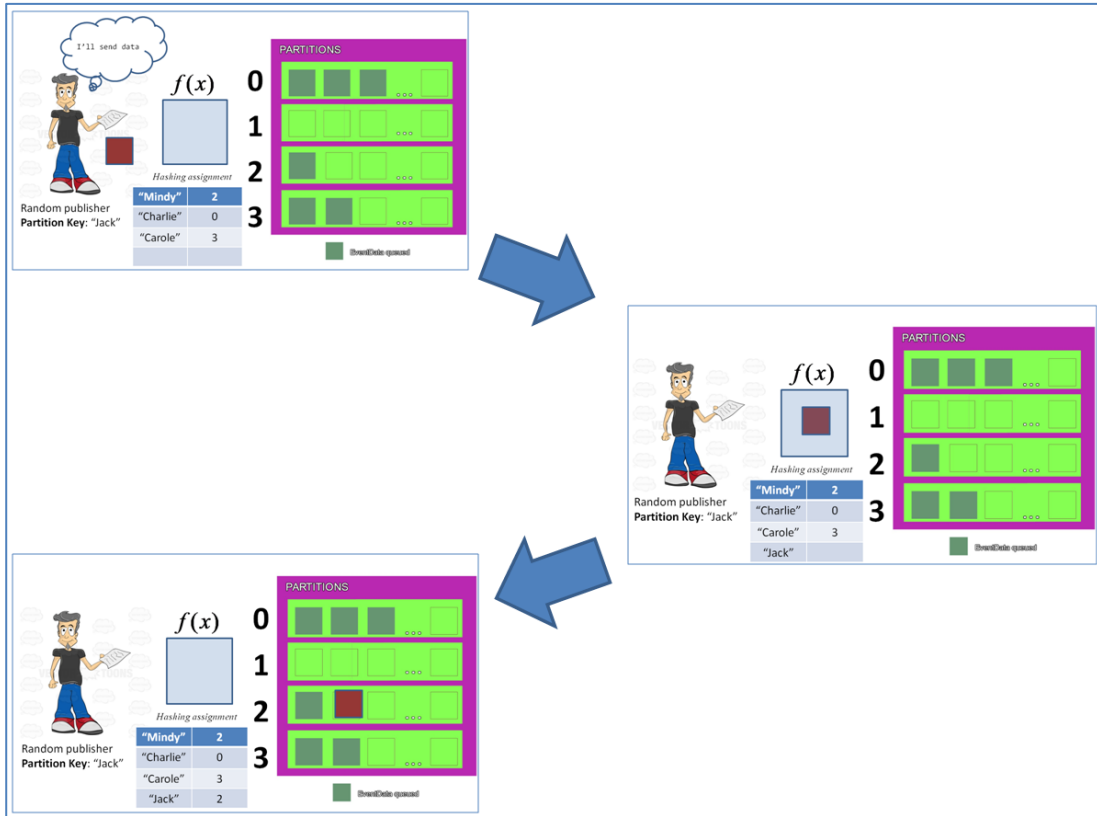


Ilustración 14. Publicación por clave de partición

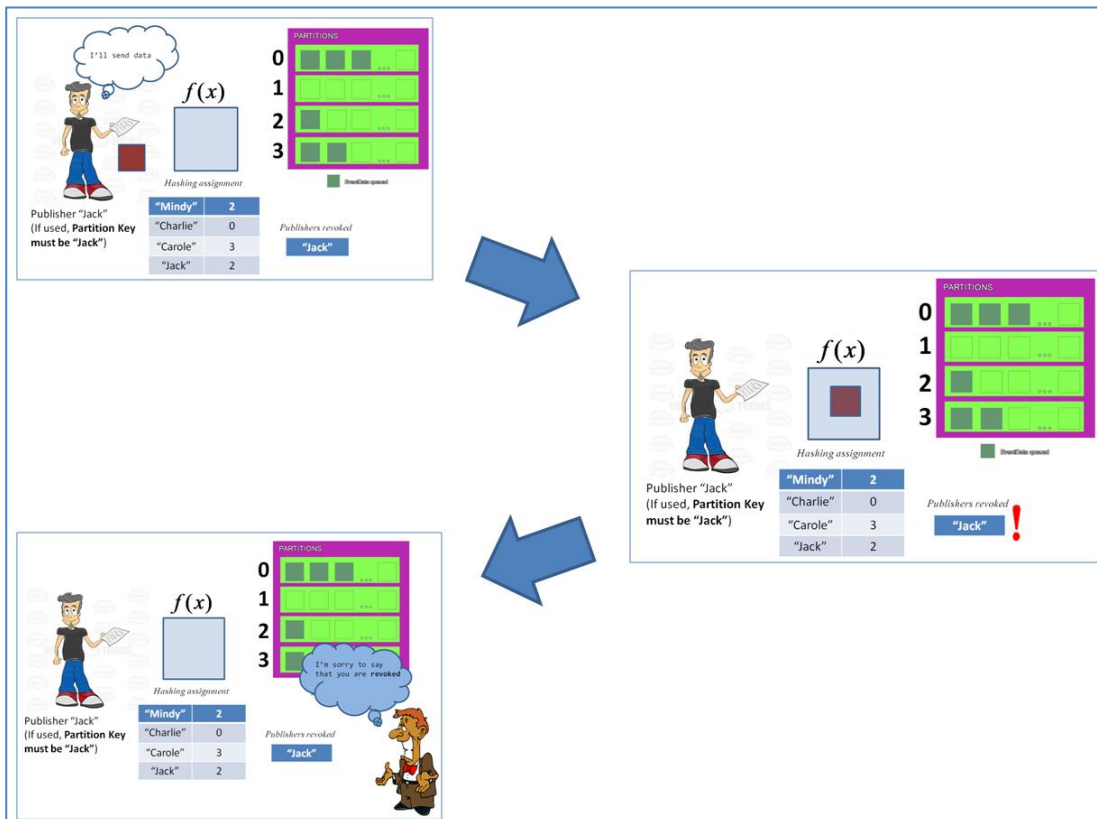


Ilustración 15. Publicación por política de publicadores

Unas claves de partición ideales pueden ser el UUID de un dispositivo o la localización geográfica del mismo.

En la publicación por política de publicadores, los publicadores están identificados con un nombre concreto, en lugar de una clave de partición. En el caso de que además utilicen una clave de partición (posible en algunos casos), esta clave y el nombre del publicador deben coincidir.

En el caso de que el publicador no especifique ni partición concreta ni clave de partición, los eventos se persistirán en las particiones por *Round Robin*. Este es el comportamiento **por defecto**.

El *Event Hub* además, garantiza que los eventos con misma clave de partición se entreguen en el mismo orden y partición en el que se publicó.

#### 5.1.2.4. Consumidores de eventos

Son los que se encargan de consumir la información (estructurada en eventos) de un *Event Hub*. La comunicación entre los consumidores y el *Event Hub* se realiza mediante conexiones AMQP, las cuales ofrecen al consumidor las siguientes ventajas:

- Una recepción de mensajes en **tiempo real**
- Un canal bidireccional con **estado y sesión**
- **Facilita el transporte de eventos** que se solicita de distintas particiones.

Los consumidores obtienen los datos a través de grupos de consumidores ([apartado 5.1.2.1. \(Particiones\)](#)). Estos grupos dotan a los consumidores de una vista independiente del *Event Hub*, es decir, cada grupo de consumidores dispone de sus *offset*, datos de control, estados... del *Event Hub*. De esta forma, dos aplicaciones distintas pueden consumir los mismos datos a distintos ritmos (ver **ilustración 16**).

El número máximo de grupos de consumidores que se pueden crear por cada *Event Hub* varía en función del nivel de mensajería definido al crear el espacio de nombres. Por ejemplo, si el espacio de nombres creado tiene un nivel de mensajería de tipo “*Standard*”, se pueden llegar a crear hasta 20 grupos de consumidores (sin contar el grupo “*\$Default*”, que es el grupo que se crea por defecto al crear el *Event Hub*). Sin embargo, si el espacio de nombres creado tiene un nivel “*Basic*”, no permite crear grupos. Este es otro aspecto a tener en cuenta si se quiere desarrollar pensando a largo plazo.

Al igual que se ha comentado en el [apartado 5.1.2.3. \(Publicadores de eventos\)](#), los consumidores de eventos necesitan obtener un *token* de firma de acceso compartido (o SAS, explicado en el apartado mencionado) a partir de una política SAS con derechos de escucha sobre el *Event Hub*. Tiene que definirse previo a la escucha de información, y es necesario para poder realizar dicha escucha.

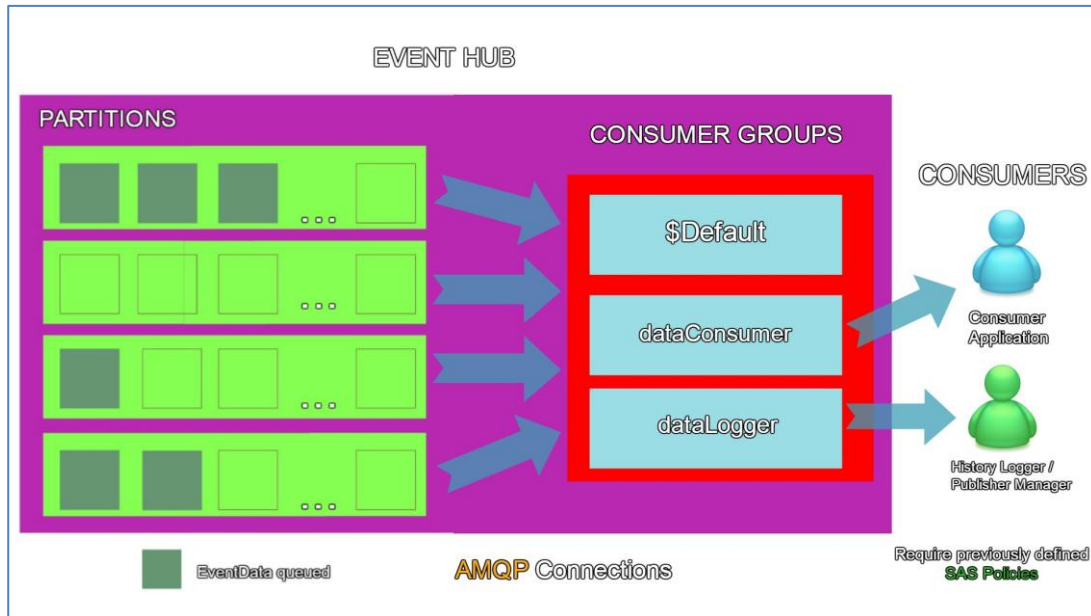


Ilustración 16. Esquema para la consumición de información en un *Event Hub*

Un grupo de consumidores puede tener varios consumidores simultáneos. Un **consumidor**, por defecto, lee de todas las particiones de su grupo de consumidores. Si se conecta un consumidor mientras otro está leyendo todas las particiones, al consumidor que estaba leyendo de todas a la vez se le desconecta algunas para que el nuevo consumidor pueda leer simultáneamente. Esto ocurre si no requiera dejar a un consumidor ya conectado sin datos que consumir. Es decir, los consumidores simultáneos se conectan, por lo menos, a una partición. Esto significa que si dentro de un mismo grupo de consumidores, donde el número de particiones es igual a el número de consumidores simultáneos conectados en ese momento, se conecta un nuevo consumidor, este último no puede obtener datos, debido a lo explicado anteriormente (una partición por consumidor como mínimo). Por estas razones hay que definir bien el número de particiones, los grupos de consumidores y los propios consumidores que van a participar en la ingesta y tratamiento de datos.

#### 5.1.2.5. Desplazamiento (*Offset*)

En los apartados anteriores se habla de este atributo como un metadato de los eventos. Más concretamente, se define como la posición de un evento dentro de una partición.

Este metadato permite que los consumidores puedan elegir dónde empezar a leer. Los valores posibles para elegir la posición para comenzar a leer pueden ser un *timestamp* (marca de fecha y hora) o un valor posicional (numérico).

El *Event Hub* no almacena ninguna referencia similar a “último evento leído”. Con lo cual, si un consumidor desea continuar obteniendo datos a partir del “último evento leído”, deberá encargarse de almacenar la referencia (u *offset*) de dicho evento, especificando su grupo de consumidores (a este concepto se le

denomina **puntos de control** o *checkpointing*, y se habla de él en el siguiente apartado).

### 5.1.2.6. Puntos de control (*Checkpointing*)

Un *checkpoint* es un *offset* persistido para su posterior uso. Se realiza una marcación o confirmación de la posición en la secuencia de eventos dentro de una partición y grupo de consumidores concretos. Además, ofrece la posibilidad de volver a datos anteriores, indicando un *offset* anterior al *checkpoint*, siempre y cuando los datos que se solicitan no hayan caducado. Si el *checkpoint* apunta a un dato no existente o caducado, el consumidor comienza a leer desde el principio de la secuencia.

La persistencia se realiza en cuentas de almacenamiento, como por ejemplo, en un contenedor de *Azure Blob Storage*. Este servicio de *Microsoft Azure* se explica con más detalle en el [apartado 5.1.5.](#)

Las siguientes ilustraciones muestran de forma más gráfica como se realiza el *checkpointing* y como se recuperan los *offset* para continuar la ingesta de datos respectivamente:

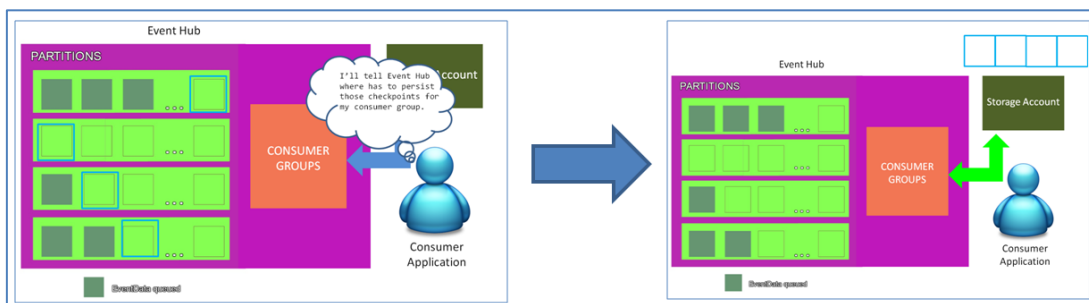


Ilustración 17. Esquema para realizar el *checkpointing*

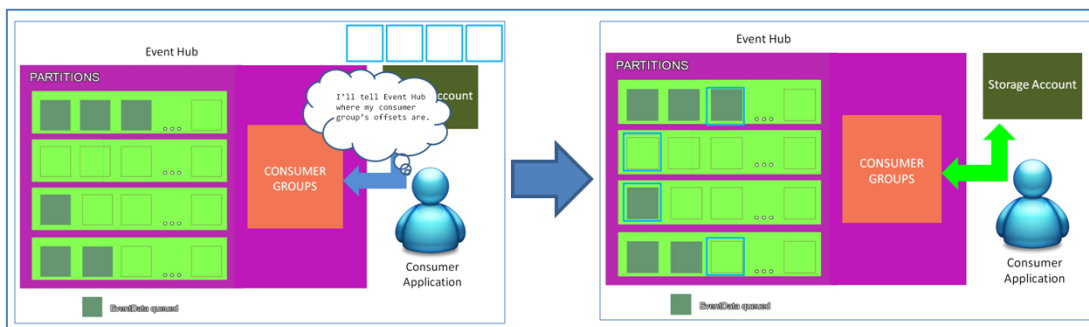


Ilustración 18. Esquema de recuperación de *offset*

Esto es, en términos generales, como trabaja internamente esta tecnología para la ingesta de datos y las posibilidades que ofrece.

### 5.1.3. Azure Stream Analytics

**Azure Stream Analytics** (o análisis de transmisiones de Azure, **ASA**) es un servicio de plataforma administrada y motor de procesamiento de eventos que se encarga de realizar un potente análisis de la información entrante para obtener conocimiento de datos muy profundos.

Este servicio ofrece la capacidad de análisis en tiempo real de flujos de datos procedentes de distintas fuentes (dispositivos, sensores, aplicaciones, sitios web...).

En este servicio, se trabaja con instancias denominadas **job** (o trabajos), donde se define uno o varios **input** (fuentes de datos), uno o varios **output** (receptores de resultados) y una consulta de transformación que indica cuales son los cálculos de análisis a realizar, expresada a través de un lenguaje similar a SQL. El funcionamiento convencional de un **job** se muestra en la **ilustración 19**:

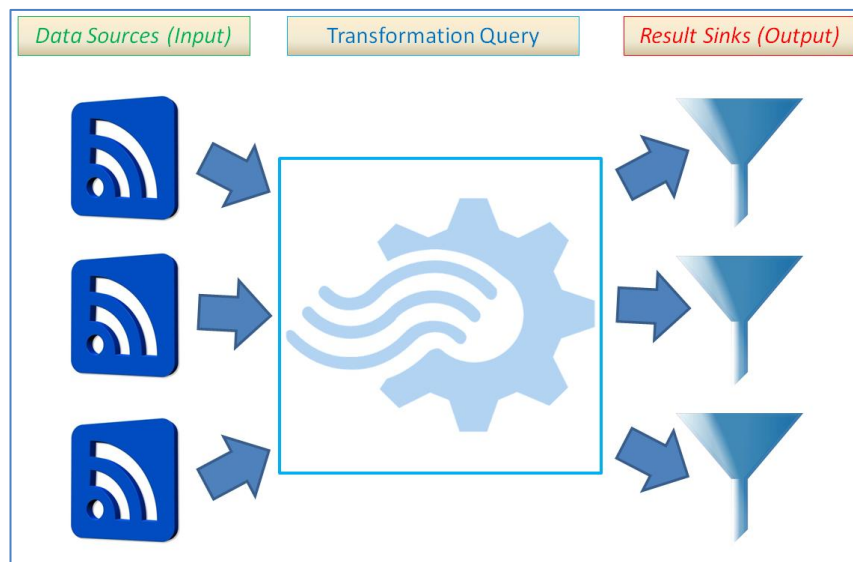


Ilustración 19. Esquema de funcionamiento habitual de un **job**

Además, se puede escalar la potencia de procesamiento incrementando las **unidades de streaming** que utiliza el mismo. Estas unidades son muy parecidas a las unidades de procesamiento, las cuales se introducen en el [apartado 5.1.2.2. \(Unidades de procesamiento\)](#). La diferencia entre ellas es que las unidades de procesamiento se definen a través de un servicio gestor (*namespace*, todas las instancias de los servicios tienen el mismo valor definido), mientras que las unidades de *streaming* se definen por instancia de servicio, es decir, cada **job** define el número de unidades de *streaming* que utiliza.

A la hora de crear un **job** hay que tener varios aspectos en cuenta. Uno de ellos es la **cuenta de supervisión regional**, que se crea como una cuenta de almacenamiento (instancia de *Azure Blob Storage*, servicio que se explica en el [apartado 5.1.5.](#)), donde persisten varios datos relacionados con esa cuenta. Esta cuenta se encarga de gestionar los datos de control que necesitan los **jobs** que

pertenezcan a una región concreta. En estos casos, por cuestiones de procesamiento y velocidad de respuesta para la solución aportada en el [apartado 7.1.](#), se utiliza la región *West Europe* (Oeste de Europa).

Otros de los aspectos a tener en cuenta, se definen en los siguientes apartados.

### 5.1.3.1. Fuentes de datos (*Input*)

Las fuentes de datos o datos de entrada son los orígenes de los cuales proviene la información a tratar por un *job*. Existen dos tipos de datos de entrada:

- Flujos de datos: Consiste en una secuencia continua de datos o **eventos** que un *job* consumirá y transformará.
- Datos de referencia: Son datos auxiliares usados para correlacionar otros datos o realizar búsquedas. Son datos estáticos o no cambian con demasiada frecuencia.

Actualmente (2016), existen tres tipos de flujos de datos definibles como entrada de un *job*:

- Centro de eventos (*Event Hub*): Proporciona flujos de trabajo en tiempo real, a gran escala, con baja latencia y alta fiabilidad. Algunas de los datos necesarios para enlazarlo como entrada de datos son:
  - Espacio de nombres del *Event Hub*
  - Nombre del *Event Hub*
  - Política SAS del *Event Hub* (debe tener derechos de **gestión**)
  - Nombre del grupo de consumidores

Este último dato, además, se utiliza para gestionar los datos referentes al *checkpointing* de particiones (se introduce en el [apartado 5.1.2.6. \(Puntos de control\)](#)), utilizando la cuenta de supervisión regional. Es decir, *Azure Stream Analytics* gestiona automáticamente el almacenamiento de datos de control como los *checkpoint*, que habría que definir según se indica en el [apartado 5.1.2.5. \(Desplazamiento\)](#).

Por lo tanto, también hace posible definir a partir de donde el *job* comienza a leer, a través de un *timestamp* proporcionado. Hay tres formas de definir dónde comienza la consumición de datos, y son las siguientes:

- Hora de inicio de trabajo (predeterminado): Indica que comienza a leer los eventos que están identificados con un *timestamp* igual o posterior a la hora en la que el *job* (o trabajo) se ha iniciado.
- Hora personalizada: Da la posibilidad de elegir un *timestamp*, donde comienza a leer los eventos que están identificados con ese *timestamp* o posterior.
- Hora de última detención (uso de *checkpoints*): El *job* comienza a leer los eventos que tienen un *timestamp* igual o

posterior al *timestamp* de la última parada. Este *timestamp*, además de otros datos de control, se encuentra almacenado en la cuenta de supervisión regional. Esta opción, además, **hace posible que no se pierdan datos cuando el *job* debe pararse** por alguna razón.

- Almacenamiento de *blobs* (*Blob Storage*): Consumo de grandes cantidades de datos no estructurados, sin orden temporal, a no ser que dichos datos dispongan de una marca (*timestamp*). Algunos de los datos necesarios para enlazarlo como entrada de datos son:
  - Nombre de la cuenta de almacenamiento
  - Nombre del contenedor

Este servicio se explica en el [apartado 5.1.5.](#)

- Centro de *IoT* (*IoT Hub*, en pruebas): Similar a *Event Hub*, aplicado a dispositivos *IoT*. Algunas de los datos necesarios para enlazarlo como entrada de datos son:
  - Nombre del *IoT Hub*
  - Política SAS del *IoT Hub* (debe tener derechos de **gestión**)
  - Nombre del grupo de consumidores

Por el contrario, los datos de referencia se obtienen directamente de una cuenta de almacenamiento de *blobs*. Las características de este tipo de entrada de datos son idénticas a las definidas en los tipos de flujos de datos (Almacenamiento de *blobs* (*Blob Storage*)).

### 5.1.3.2. Receptores de resultados (*Output*)

Los receptores de resultados son los que se encargan de consumir los resultados obtenidos tras modificar la información inicial con la consulta de transformación de un *job*. Existen varias posibles salidas para los resultados proporcionados por el *job*, las cuales se muestran en la **ilustración 20**.

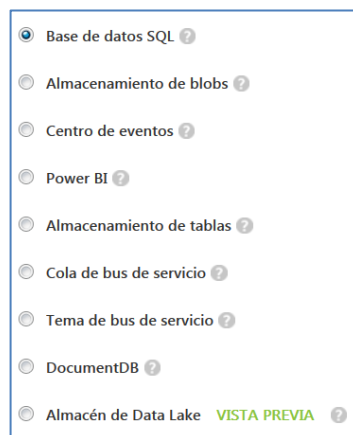


Ilustración 20. Posibles receptores de resultados definibles para un *job* (2016)



No se explica cada una de las tecnologías, ya que muchas de ellas no están incluidas en el alcance de este proyecto. Sin embargo, una de ellas sí que se utiliza, y se define a continuación:

- **Power BI:** Elemento para visualizar resultados en gráficos y otros elementos (también denominados **visualizaciones**) que trabajan en tiempo real. Además de autorizar su uso con una cuenta organizativa de *Microsoft (Office 365*, ver [apartado 5.3.](#)), algunos de los datos necesarios para enlazarlo como salida de datos son:
  - Nombre del *dataset*
  - Nombre de la tabla
  - Grupo de trabajo

Este servicio se explica en el [apartado 5.2.2.](#)

### 5.1.3.3. Consulta de transformación

La consulta de transformación es una consulta, escrita en un lenguaje similar a SQL, que se encarga de obtener datos más profundos a partir de información provista por fuentes de datos (*Input*) y devolviendo los resultados obtenidos a los receptores de resultados (*Output*) de un *job*.

Un *job*, permite la **multiconsulta**, es decir, puede realizar varias consultas en paralelo. Gracias a esto, es posible utilizar varias fuentes de datos y receptores de resultados al mismo tiempo, en un mismo *job*.

Si que soporta **subconsultas**, pero con varias limitaciones. Por ejemplo, al utilizar subconsultas, no permite funciones agregadas en ellas.

En el caso de los flujos de datos como entrada de información, se puede definir ciertos tipos de **ventanas de tiempo** para tratar la información entrante con un mayor control. Los tipos de ventanas de tiempo existentes son las siguientes:

- ***Tumbling Window***: Indica que los datos deben leerse en series de intervalos de tiempo de tamaño fijo, no superpuestos y contiguos.
- ***Hopping Window***: Similar a *Tumbling Window*, pero con intervalos de tiempo superpuestos.
- ***Sliding Window***: Considera todas las posibles ventanas de tiempo de un tamaño concreto y puede obtener nuevos eventos cuando el contenido de la ventana cambia.

Se muestran a continuación, de forma más gráfica, cómo funciona cada una de estas tres ventanas de tiempo, con las siguientes ilustraciones:



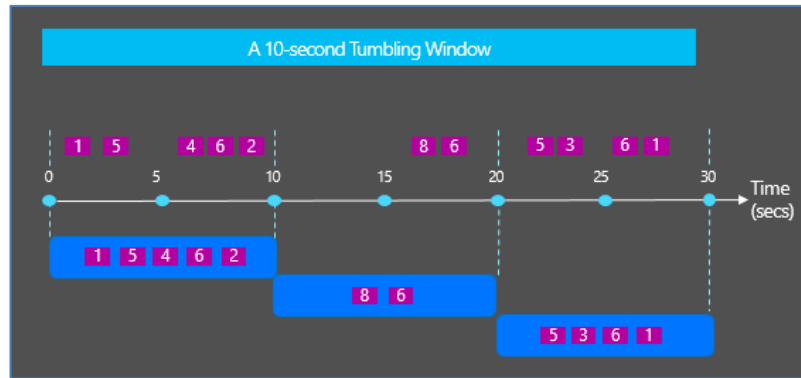


Ilustración 21. Ejemplo de *Tumbling Window* (extraído de *blogs de Microsoft*)

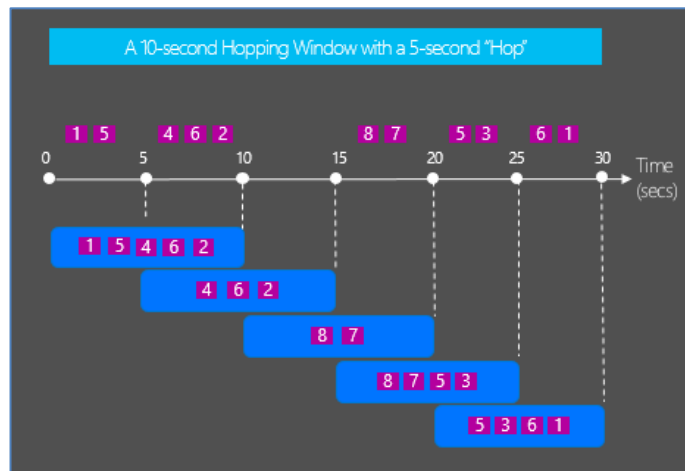


Ilustración 22. Ejemplo de *Hopping Window* (extraído de *blogs de Microsoft*)

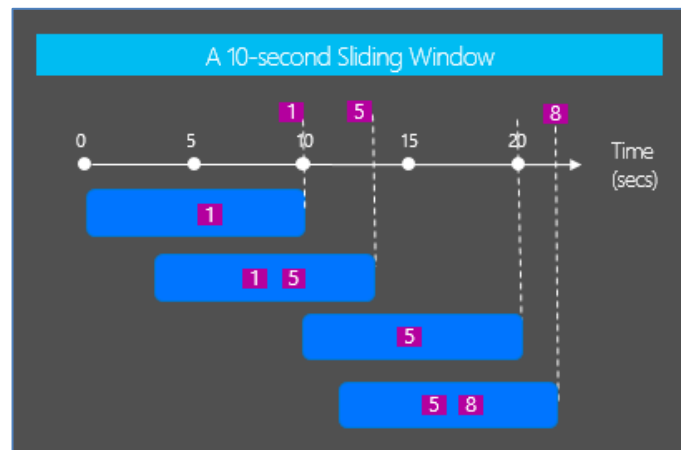


Ilustración 23. Ejemplo de *Sliding Window* (extraído de *blogs de Microsoft*)

Otra característica que soporta *Azure Stream Analytics*, la cual está en fase de prueba (2016), son las **funciones**.

Las funciones se pueden definir como referencias a servicios web desplegados a través del entorno de *Machine Learning*, servicio el cual se explica en

el siguiente apartado ([apartado 5.1.4.](#)). Algunos de los datos que se deben definir para utilizar un servicio web en la consulta de transformación son:

- Área de trabajo
- Servicio web
- Punto de conexión

#### 5.1.4. Machine Learning

**Machine Learning** (o aprendizaje automático de *Azure*, *Azure ML*) es un servicio de plataforma administrada que hace posible el diseño de modelos predictivos que aprenden de los datos que existen y obtienen, para prever los resultados y comportamientos futuros del sistema en el que se integran.

Es capaz de hacer que tanto el propio sistema como los dispositivos que utilizan dicho sistema tengan un comportamiento más inteligente. Un ejemplo de aprendizaje automático en la nube es [la tienda online de Amazon](#). Esta tienda es capaz de recomendar productos a los clientes, en función de lo que ellos compran.

**Machine Learning** proporciona unas herramientas gráficas para crear, probar, desarrollar y gestionar soluciones de análisis predictivo en la nube. Estas soluciones se desarrollan a través un **área de trabajo**, en la cual se crean y editan los experimentos.

Un **área de trabajo** o *workspace*, es una instancia de *Machine Learning* donde se desarrollan y almacenan experimentos, conjuntos de datos (o *datasets*), servicios web desplegados, modelos entrenados y otro tipo de recursos relacionados con este servicio.

Un **experimento** es un grupo de módulos relacionados entre sí, los cuales en su conjunto definen una lógica de procesamiento de los datos que este grupo obtiene o genera.

Un modelo predictivo se desarrolla dentro de un experimento. Se debe tener en cuenta que no es requisito imprescindible que un experimento genere, pruebe o use un modelo predictivo. Además, en un experimento no puede generarse más de un modelo predictivo simultáneamente. Nótese que se distingue entre generación (o **entrenamiento**) del modelo predictivo y **uso** del mismo (ver [ilustraciones 24 y 25](#)).

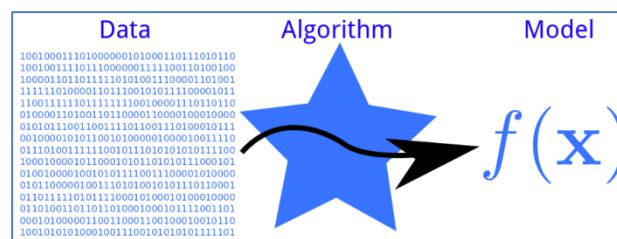


Ilustración 24. Entrenamiento de un modelo (imagen realizada por Philippe Desjardins-Proulx)

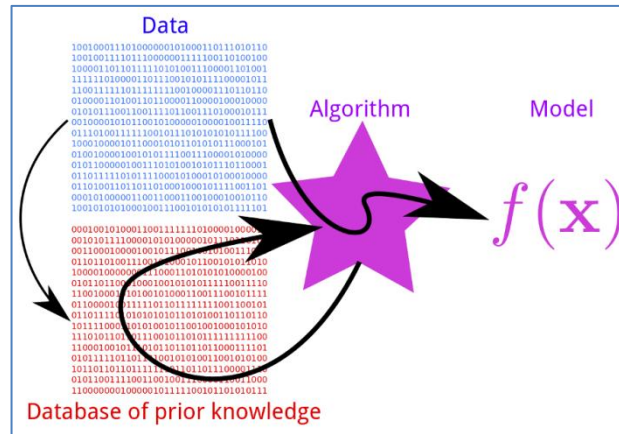


Ilustración 25. Uso y consecuente entrenamiento progresivo de un modelo (imagen realizada por Philippe Desjardins-Proulx)

Un **módulo** es un componente que realiza una acción con uno conjunto de datos determinado (*dataset*), que a continuación, devuelve el resultado de la acción realizada. El módulo no tiene por qué tener datos de entrada. Este caso se da si dicho módulo se encarga de generar los datos o de obtenerlos, y por tanto, su acción sería definir como los genera o de donde los obtiene respectivamente.

Un *dataset* es un conjunto de datos estructurado, generalmente originario de algún proveedor de datos, que tienen organizada la información en formato tabular y actúa como fuente de datos. Las filas son registros, y las columnas, los distintos campos existentes por cada registro; similar a las tablas de una base de datos. Un *dataset* se puede crear en el portal, cargando un fichero fuente, a través de la opción “+NEW → Dataset → From Local File”, situada en la parte inferior izquierda de la interfaz de desarrollo (ver **ilustración 26**). Los formatos de un fichero fuente puede ser de varias extensiones (.csv, .txt, .nh.csv, .zip...).

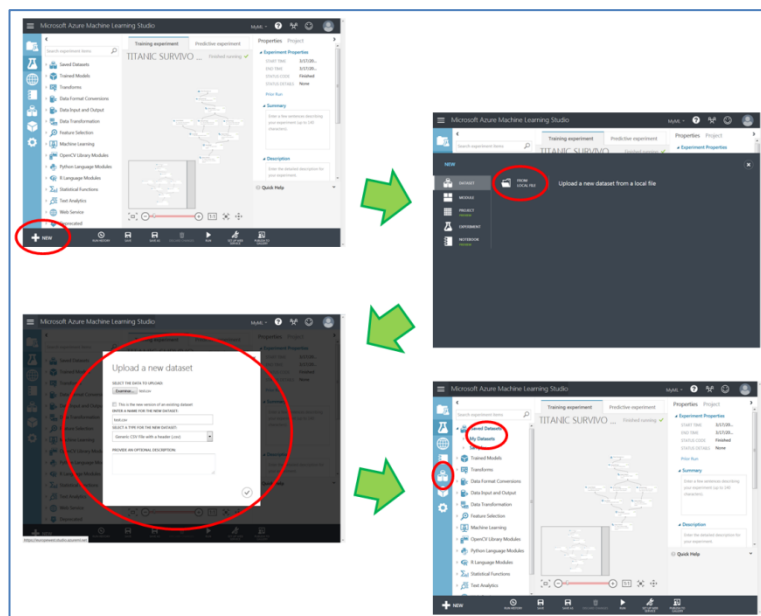
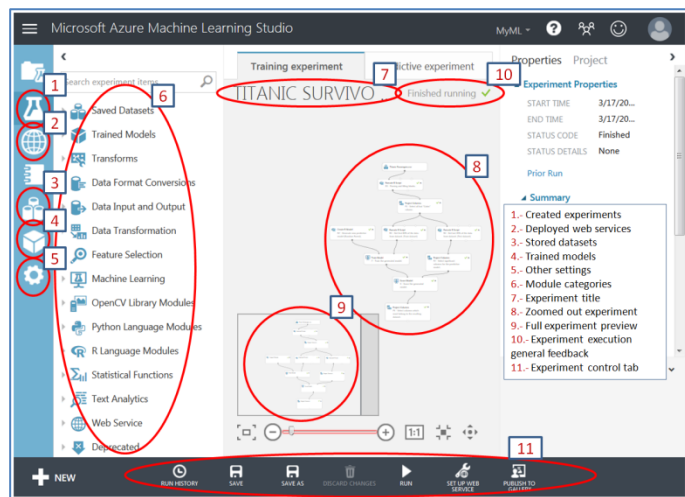


Ilustración 26. Proceso para crear un *dataset* de pruebas a partir de datos en un fichero local

Posteriormente, al visitar el apartado de “*Saved Datasets* → *My Datasets*” en la interfaz de desarrollo del experimento, se encuentra disponible para su uso un nuevo *dataset*, con la información del fichero fuente subido recientemente.

Una **acción** es la actividad que realiza un módulo para procesar el *dataset* entrante o relacionar varios (si los tiene; en caso de que no tenga, su acción consiste en generar la información u obtenerla de alguna manera). Cada módulo tiene una acción diferente, la cual se puede configurar.

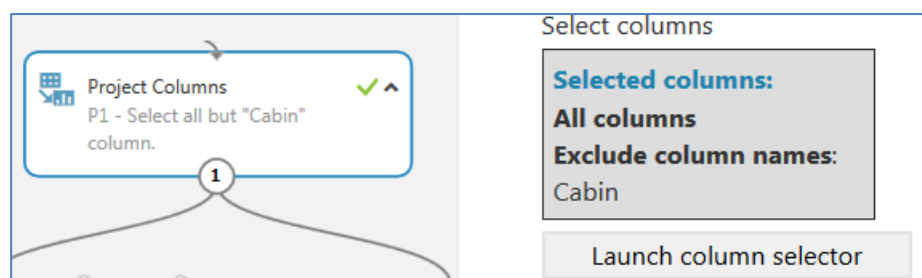
Las herramientas gráficas pueden utilizarse a través del portal de [Microsoft Azure Machine Learning Studio](https://azureml.microsoft.com). En este portal se pueden crear experimentos, utilizando una interfaz gráfica orientada a un uso “*drag and drop*”, y así, poder desarrollar los modelos predictivos, de una forma fácil e intuitiva (ver **ilustración 27**).



**Ilustración 27. Elementos destacables de la interfaz de desarrollo de un experimento**

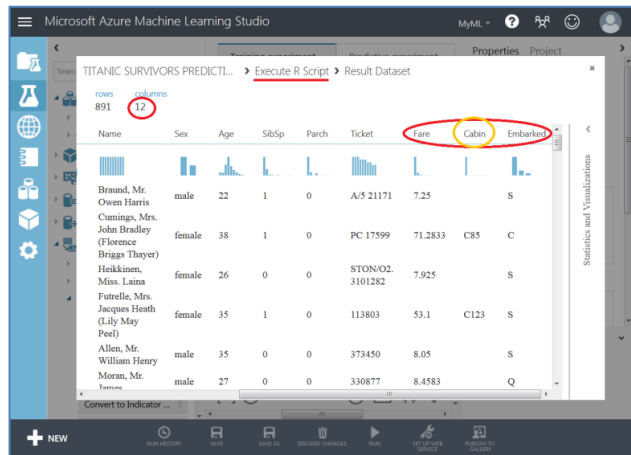
Para ello, se eligen los módulos deseados a través de las categorías disponibles (**etiqueta 6** de la **ilustración 27**), se arrastran dichos módulos al panel central y se conectan entre ellos.

Una vez realizadas las relaciones entre módulos y se configuren las acciones de estos (ver **ilustración 28**), en el apartado de control del experimento (**etiqueta 11** de la **ilustración 27**), comenzar la ejecución del experimento desarrollado con el botón denominado “**RUN**”.

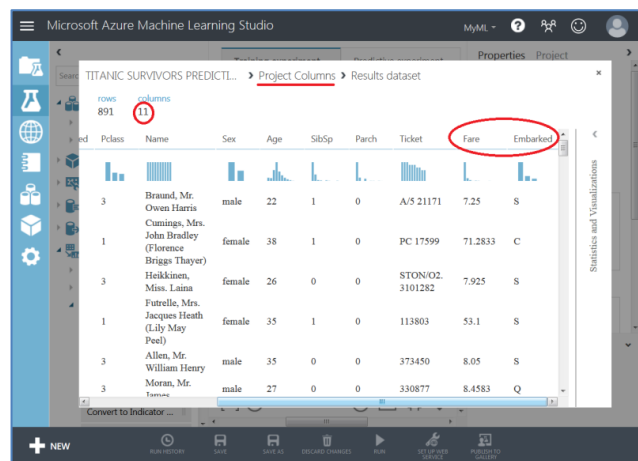


**Ilustración 28. Módulo "Project Columns" (izquierda) con su acción definida (derecha)**

Como ejemplo de funcionamiento de un módulo, en el caso de la **ilustración 28**, obtiene el *dataset* resultante de otro módulo (ver **ilustración 29**), excluye la columna “Cabin” del mismo y devuelve su *dataset* resultante (ver **ilustración 30**). La mayoría de módulos trabajan de esta forma; obteniendo un *dataset* y devuelven otro.



**Ilustración 29.** Dataset entrante del módulo "Project Columns"



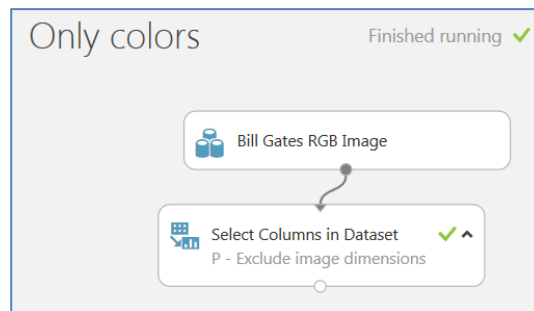
**Ilustración 30.** Dataset resultado del módulo "Project Columns"

En la ejecución del experimento, pueden acaecer tres situaciones distintas (descritas de menor a mayor gravedad):

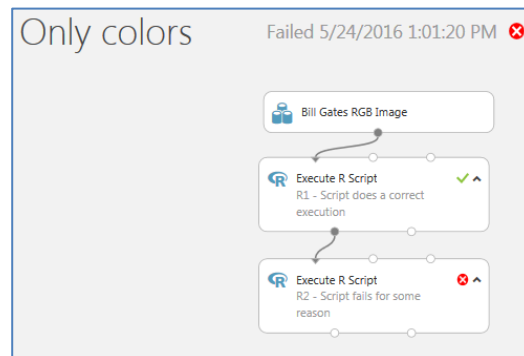
- El experimento se ejecuta correctamente (el *feedback* general del experimento [etiqueta 10 de la **ilustración 27**] y los propios módulos del experimento [etiqueta 8 de la **ilustración 27**], lo hacen ver con una visible marca de verificación verde; ver **ilustración 31**) y los resultados que se obtienen son los esperados (caso ideal).
- El experimento falla a causa de algún módulo (el *feedback* general del experimento [etiqueta 10 de la **ilustración 27**] y los propios módulos del experimento [etiqueta 8 de la **ilustración 27**], lo hacen ver con una visible marca de error roja. Los módulos que se ejecutan

sin errores, muestran una marca de verificación verde; ver **ilustración 32**). Esto puede deberse a varias razones:

- Los módulos requieren una definición de su acción.
  - Algún módulo requiere de más entradas de datos.
  - En el caso de módulos cuya acción es ejecutar un script, indica que la **sintaxis** de dicho código es incorrecta.
- El experimento se ejecuta correctamente (el *feedback* general del experimento [**etiqueta 10** de la **ilustración 27**] y los propios módulos del experimento [**etiqueta 8** de la **ilustración 27**], lo hacen ver con una visible marca de verificación verde; ver **ilustración 31**), sin embargo, los resultados que se obtienen no son correctos. Esto puede deberse a varias razones:
    - La relación entre módulos no es correcta.
    - En el caso de módulos cuya acción es ejecutar un script, indica que la **semántica** de dicho código es incorrecta.

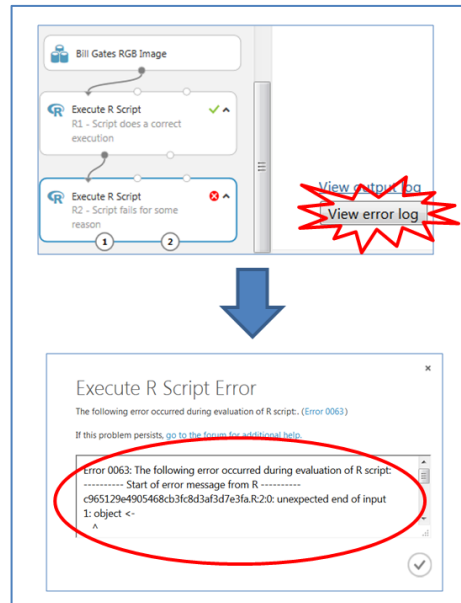


**Ilustración 31.** *Feedback* de un experimento de ejecución satisfactoria



**Ilustración 32.** *Feedback* de un experimento de ejecución fallida

En el caso de la **ilustración 32**, puede comprobarse una aproximación de cuál ha sido la causa de dicho error, accediendo a las propiedades del módulo seleccionado, donde puede verse una breve descripción del error (ver **ilustración 33**).



**Ilustración 33. Breve descripción del error que ha ocasionado la parada del módulo**

En este caso, se ha dado un error sintáctico, ya que el código ejecutado por el módulo de ejecución de código R ha determinado que parte del código descrito como acción tiene errores de sintaxis. Este tipo especial de módulos se utilizan para desarrollar la solución aportada en este proyecto ([apartado 7.1.](#)).

Inicialmente, a un experimento se le denomina **experimento de entrenamiento**. En un experimento de entrenamiento, se diseña el entrenamiento de un modelo predictivo u otro tipo de procesamiento el cual no tiene por qué estar relacionado con modelos predictivos. Si el experimento se ejecuta correctamente, se ofrece la posibilidad de transformarlo a un experimento predictivo.

Un **experimento predictivo** consiste en casi la misma configuración de módulos que un experimento de entrenamiento. Lo que cambian de uno a otro es que, en el predictivo, se sustituyen los componentes de entrenamiento por un **modelo entrenado** (creado en el proceso de transformación del experimento) e inclusión de un *input* y *output* de servicio web para poder desplegar este experimento como tal.

Este experimento, a pesar de que algunos módulos aparecen y desaparecen automáticamente para adaptar el experimento, es moldeable de igual forma que el experimento de entrenamiento.

Una vez adaptado y se ejecute correctamente este experimento, ofrece la posibilidad de desplegarlo como un servicio web, de tal forma que se pueda aprovechar en otros servicios o aplicaciones.

El proceso para desplegar un experimento inicial como un servicio web se explica de manera más gráfica en la **ilustración 34**.

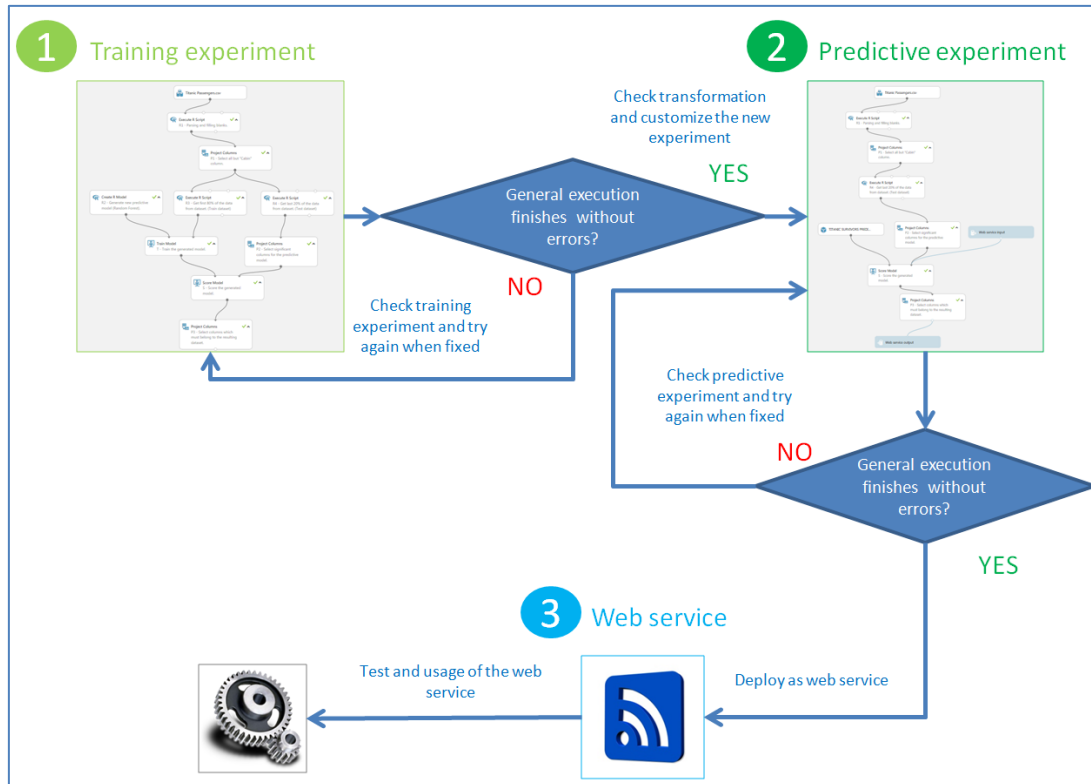


Ilustración 34. Proceso para desplegar un experimento como un servicio web.

Al desplegarlo como servicio web, genera una página de ayuda y documentación expresamente para ese servicio web, donde se puede encontrar, entre otras cosas:

- La estructura de datos que acepta el servicio web
- La estructura de datos que devuelve el servicio web
- Código de ejemplo para probar el servicio web (en **C#**, **Python** y **R**)

Otro elemento que se puede definir en un servicio web son los *endpoints*. Un *endpoint* sería algo similar a los grupos de consumidores en *Azure Event Hubs* ([apartado 5.1.2.4. \(Consumidores de eventos\)](#)). Se considera un consumidor del servicio web.

Cada *endpoint* puede tener sus propios modelos entrenados, mientras siguen enlazados con el experimento que ha creado el servicio web. Además, permite aplicar actualizaciones al experimento del mismo, sin alterar las configuraciones personalizadas de este.

Otra característica que ofrece *Machine Learning* es la **capacidad para poder reentrenar un modelo predictivo**, con el objetivo de volver a un estado de aprendizaje anterior, el cual se considere más estable y preciso. Este proceso se puede poner en funcionamiento a través de una API, de la cual se indica sus limitaciones en el [apartado 7.2.](#).

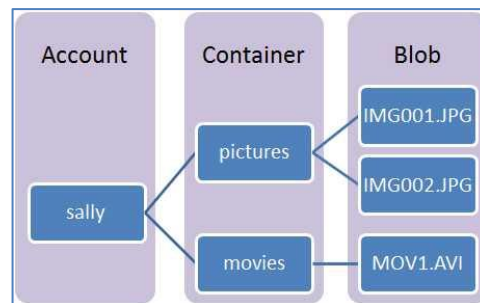
En términos generales, estos son algunos de los elementos a conocer sobre *Machine Learning*, para poder desarrollar experimentos a través del portal.



### 5.1.5. Azure Blob Storage

**Azure Blob Storage** es un servicio de almacenamiento en la nube para gran cantidad de datos no estructurados, para objetos, texto o datos binarios. Se puede utilizar para almacenar datos de aplicaciones que tienen un carácter estático o que cambian con poca intensidad, o para realizar un almacenamiento de archivos para acceso distribuido, entre otras cosas.

Una instancia de este servicio es una cuenta de almacenamiento, la cual simula una cuenta para almacenar información, como los servicios de *Dropbox* o *Google Drive*. En ella, se crean **contenedores** (similar a las carpetas en una cuenta de almacenamiento convencional), y dentro de estos se almacenan los **blobs** (elementos de gran tamaño almacenados en un contenedor). En la **ilustración 35**, se puede ver esta estructura de una forma más gráfica.



**Ilustración 35.** Cuenta de almacenamiento (izquierda), que tiene contenedores (centro), cada uno con sus blobs almacenados (derecha) [extraído de *blogs de Microsoft*]

Se puede definir la privacidad de un contenedor. Los niveles de privacidad que se pueden definir son los siguientes:

- Privado (predeterminada): Sólo puede acceder el propietario de la cuenta.
- Blob público: Permite el acceso de lectura público a los blobs del contenedor. Sin embargo, no permite el acceso a las propiedades ni a los metadatos del mismo.
- Contenedor público: Permite el acceso de lectura público completo al contenedor y a los blobs que este contiene.

Algunas de las características que se pueden definir sobre las cuentas de almacenamiento son:

- Replicación: Replicar los datos almacenados con el objetivo de protegerlos en el caso de que se produzca un error o interrupción. Los tipos de replicación existentes son los siguientes:
  - Replicación localmente redundante: Replicación de los datos en la misma región.
  - Replicación geográficamente redundante (predeterminada): Replicación de los datos en una región de almacenamiento secundaria.

- Replicación geográficamente redundante con acceso de lectura: Replicación de los datos en una región de almacenamiento secundaria, con acceso de sólo lectura.
- Supervisión de almacenamiento: Estadísticas acumuladas de las transacciones y datos de capacidad de un servicio de almacenamiento. Definible una retención de la supervisión en días junto con el nivel de supervisión: desactivado, mínimo o detallado.
- Registro de almacenamiento o logs: Registra información sobre las solicitudes realizadas a un servicio de almacenamiento. Puede supervisar una solicitud para diagnosticar problemas. Estos registros se almacenan en un contenedor denominado **\$logs**.

Esto son, en términos generales, cómo los aspectos generales de esta tecnología para persistencia de datos y las posibilidades que ofrece.

### 5.1.6. Otras tecnologías

En este apartado, se definen algunas tecnologías extra que se han tratado. Algunas se trabajan de forma puntual, pero que aportan mejoras para el proyecto, mientras que otras se han estudiado y trabajado menos debido a un insuficiente tiempo para su dedicación.

#### 5.1.6.1. Azure Service Bus

*Azure Service Bus* (o bus de servicio) es un servicio en la nube multiinquilino, en el cual a través de un **espacio de nombres** (o *namespace*, instancia de este servicio) se pueden definir varios mecanismos de comunicación dentro del mismo. Multiinquilino, en este caso, significa que varios usuarios pueden compartir el mismo servicio. Según la **ilustración 10**, este servicio gestiona cuatro mecanismos de comunicación diferentes, los cuales también se consideran servicios. Son los siguientes:

- Queues (o colas): Comunicación unidireccional que funciona como un intermediario que almacena los mensajes hasta que alguien los recibe. Cada mensaje puede tener un único destinatario.
- Topics (o temas): Comunicación unidireccional mediante suscripciones, donde cada tema puede tener varias suscripciones. De igual manera que en las colas, son intermediarios que almacenan los mensajes hasta que alguien los recibe. Cada suscripción tiene la posibilidad de aplicar filtros en la recepción de mensajes, para recibir únicamente los que cumplan ciertos criterios. Tecnología que cumple con el patrón publicador/suscriptor.
- Relays (o retransmisiones): Comunicación bidireccional que, a diferencia de las colas y los temas, no son intermediarios, es decir, no almacenan los mensajes que reciben. Se encargan de transmitirlos a la aplicación de destino.

- Event Hubs o centros de eventos: Proporcionan ingresos de telemetría y **eventos** a escala masiva. Es un intermediario que almacena estos durante un periodo de tiempo, los cuales pueden recibirse por varios consumidores del servicio. Ofrece baja latencia y alta confiabilidad.

Además, desde el espacio de nombres es desde donde se gestionan varios aspectos de seguridad, como las **firma de acceso compartido** de cada mecanismo de comunicación, o las firma de acceso compartido del propio espacio de nombres. Estas últimas se utilizan para ofrecer la posibilidad de gestionar el espacio de nombres programáticamente, con el fin de hacer que las configuraciones realizadas sean portables.

Otro elemento de seguridad que se puede gestionar sobre un espacio de nombres son las comunicaciones punto a punto entre un emisor de datos y la plataforma. Este tipo de comunicaciones puede protegerse mediante certificados autofirmados. Este aspecto se trata en el siguiente apartado (con *OpenSSL*).

#### 5.1.6.2. *OpenSSL*

*OpenSSL* es un proyecto de software libre, que consiste en un conjunto de herramientas y bibliotecas relacionadas con la criptografía, que suministra funciones criptográficas a otros paquetes y navegadores web. En otras palabras, API que ayuda a implementar protocolos relacionados con la seguridad (SSL/TLS) y posibilita el acceso a sitios web de forma segura a través de HTTPS.

Permite crear **certificados**, que pueden aplicarse a un servidor para proteger las comunicaciones punto a punto con sus clientes.

Estos certificados, permiten autenticar a los participantes de la comunicación y permiten cifrar los datos que viajan entre ellos, para evitar que, a través de un *sniffer*, se pueda extraer y entender la información intercambiada entre ambos.

Existen dos tipos de certificados:

- Certificados provistos por una autoridad certificadora: Provee y valida la identidad del servidor. Este aplica su firma sobre la petición de certificación. A continuación, el navegador puede comprobarlo a través de su base de datos de **certificados raíz**. De este modo, el navegador confía en que el sitio es quien dice ser, ya que según la autoridad emisora del certificado, la firma utilizada por el servidor es válida.
- Certificado autofirmado: Similar al anterior, con la diferencia de que no existe un tercero (autoridad certificadora) que pueda validarlo.

No obstante, no es necesaria la validación de un tercero para asegurar las comunicaciones. Para ello se utiliza un certificado autofirmado, con el que no se delega la verificación de la identidad mediante un tercero, y por consiguiente, hay que confiar en el emisor del certificado.

Para ello, se puede generar este tipo de certificados, haciendo uso de la API anteriormente mencionada.

*OpenSSL*, al ser un proyecto de software libre (que, por definición, puede ser usado, copiado, estudiado, modificado y redistribuido), se puede descargar su herramienta y usarla libremente para cualquier propósito. La herramienta es accesible a través del dominio oficial, [openssl.org](https://openssl.org).

A continuación, se desarrolla un ejemplo de cómo asociar un certificado autofirmado a un espacio de nombres de *Azure Service Bus*.

Los requisitos necesarios para desarrollar este ejemplo son los siguientes:

- Disponer de la herramienta a través de la [página oficial](#).
- Disponer de un fichero de configuración, como el que proporciona el [MIT](#), e incluirlo en la misma carpeta donde se extrae la herramienta. En este caso, el fichero de configuración se llama “openssl.cnf”.
- Disponer de un espacio de nombres creado. En este ejemplo se utiliza uno de prueba llamado “certificatetest”.
- A la hora de crear el certificado, ajustarse a los requisitos que indica *Microsoft* en este enlace: [Configuring SSL for an application in Azure](#) >> *Step 1: Get an SSL certificate*.

En este caso, el desarrollo que se hace de este ejemplo, se realiza en un sistema operativo *Windows*. Sin embargo, el desarrollo no es muy diferente en el resto de sistemas.

Se comienza “[ejecutando la herramienta como administrador](#)”. Se abre una consola, donde se realizan unos pasos descritos a continuación para generar el certificado autofirmado.

1. Generar una clave privada de 2048 bits de cifrado en el fichero “mykey.key”, introduciendo a continuación una contraseña para protegerla:

```
genrsa -des3 -out mykey.key 2048
```

2. Generar una petición del certificado con la clave recientemente creada y el fichero de configuración. Solicita la contraseña que protege la clave privada:

```
-req -config openssl.cnf -new -key mykey.key -out request.csr
```

3. Se piden introducir varios valores, los cuales no son obligatorios introducir, y muchos tienen valores por defecto. Hay que centrarse en el campo llamado “*Common Name (eg, YOUR name)*”, ya que en este campo hay que introducir el espacio de nombres completo. En este caso, *Common Name* toma el valor

“**certificatetest.servicebus.windows.net**”. Una vez hecho esto, la petición se genera en el fichero “request.csr”.

4. Firmar la petición del certificado, indicando su caducidad, y generando el certificado en el fichero “certificate.cer”. Solicita la contraseña que protege la clave privada:

```
x509 -req -days 365 -in request.csr -signkey mykey.key -out certificate.cer
```

5. Generar un fichero de exportación del certificado para poder instalarlo en el navegador, indicando un nombre descriptivo del mismo. Solicita la contraseña que protege la clave privada y otra para proteger la exportación :

```
pkcs12 -export -in certificate.cer -inkey mykey.key -name "INTRODUCIR_NOMBRE_DESCRIPTIVO" -out export.p12
```

6. Una vez generado el certificado y el fichero de exportación, se procede a instalarlo en el servidor y en el cliente. Para instalarlo en el servidor (en este caso, es una suscripción de [Microsoft Azure](#)), se accede a “Settings > Management Certificates > Upload” y elegir el archivo “certificate.cer”. La siguiente ilustración muestra este proceso.

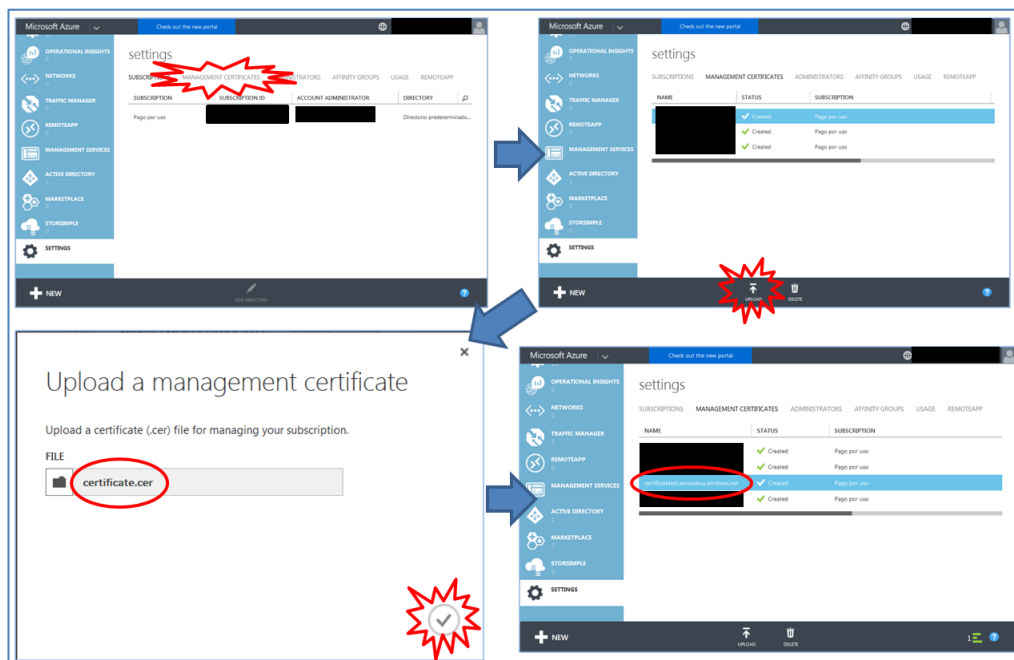


Ilustración 36. Proceso para subir el certificado de un namespace a Microsoft Azure

7. Instalar el certificado en el cliente (máquina local), ejecutando el fichero de exportación (*export.p12*). Solicita la contraseña que protege la exportación y varios parámetros más. En este caso, se define que el certificado se guarde en el almacén denominado “Personal”. Se puede comprobar que la importación se ha realizado

correctamente, ejecutando la herramienta para gestión de certificados local denominada *certmgr.msc*, y accediendo al almacén de certificados “Personal”.

De este modo, se consigue una protección extra en el acceso a mecanismos de mensajería como *Azure Event Hubs* ([apartado 5.1.2.](#)), ya que, para utilizar este servicio programáticamente, el cliente requiere de un certificado compatible con el *namespace* del recurso que busca, además de disponer de un *token* de firma de acceso compartido.

### 5.1.6.3. API de Azure Service Bus

La API de *Azure Service Bus*, se utiliza para gestionar e interactuar con espacios de nombres creados en la plataforma programáticamente.

De esta forma, permite una portabilidad de la configuración realizada en el [portal de Azure](#) sobre este servicio.

Entre otras cosas, permite crear espacios de nombres, gestionar sus reglas de acceso, crear instancias de los servicios gestionados, gestionar las reglas de acceso de los propios servicios, etc.

En este proyecto, se utiliza para que, una aplicación de consola, pueda interactuar con un espacio de nombres y poder enviar datos a un *Event Hub*.

Para poder utilizar esta API en un proyecto, se puede importar fácilmente las librerías necesarias utilizando la consola de **NuGet** (gestor de paquetes) en Visual Studio (*Tools > NuGet Package Manager > Package Manager Console*), ejecutando el siguiente comando:

```
Install-Package WindowsAzure.ServiceBus
```

A partir de aquí, es posible gestionar programáticamente tanto los espacios de nombres creados en la plataforma, como las instancias de los servicios que estos sostienen.

### 5.1.6.4. API de Azure Stream Analytics

La API de *Azure Stream Analytics*, se utiliza para gestionar *jobs* creados en la plataforma y alterar sus propiedades programáticamente.

De esta forma, permite una portabilidad de la configuración realizada en el [portal de Azure](#) sobre este servicio.

Entre otras cosas, permite crear *jobs*, añadirles *inputs* y *outputs*, testear las conexiones, definir o modificar la consulta de transformación, iniciar y parar los *jobs*, etc.

Esta API no es necesaria para desarrollar la solución aportada, ya que el *job* se crea una vez y a modo de prueba. Además, esta API tiene una limitación, y es que, actualmente (2016), programáticamente no se puede definir como *output* una cuenta de *Power BI*. Es un problema, ya que para la solución adoptada es necesario definir dicho *output*.

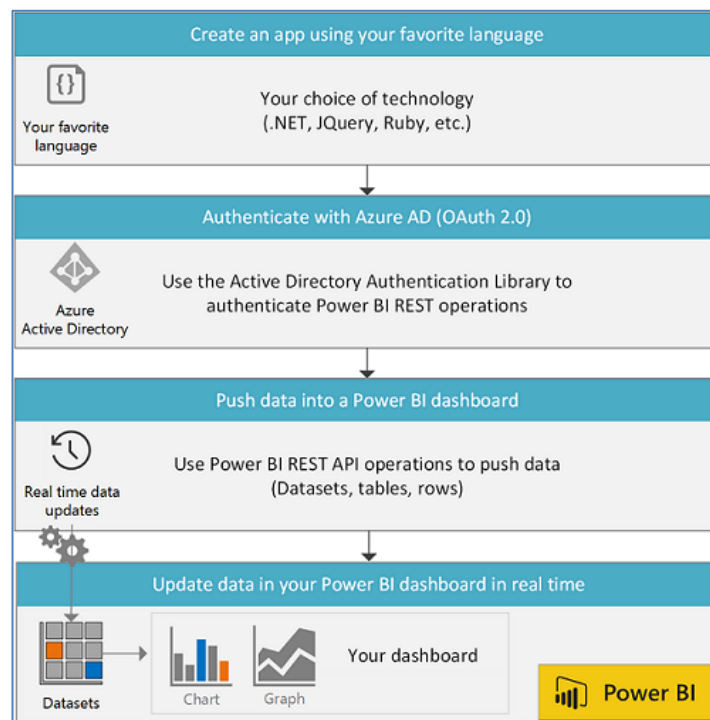
Para poder utilizar esta API en un proyecto, se puede importar fácilmente las librerías necesarias utilizando la consola de **NuGet** (gestor de paquetes) en Visual Studio (*Tools > NuGet Package Manager > Package Manager Console*), ejecutando el siguiente comando:

**Install-Package Microsoft.Azure.Management.StreamAnalytics**

A partir de aquí, es posible gestionar programáticamente los *jobs* creados en la plataforma.

### 5.1.6.5. API de Power BI

La API de *Power BI* se basa en el uso programático de *Azure Active Directory*, autenticándose mediante **OAuth 2.0** con la cuenta corporativa de *Power BI*, para así poder gestionar los recursos asociados a dicha cuenta mediante REST. Es decir, que no hay librerías asociadas a esta API. Sin embargo, requiere de librerías de *Azure Active Directory*. En la siguiente ilustración se muestra un esquema sobre cómo se consume esta API, a través de una aplicación cliente.



**Ilustración 37.** Flujo para gestionar recursos de *Power BI* programáticamente (imagen extraída de *blogs de Microsoft*)



Entre otras cosas, permite crear *datasets*, crear tablas, crear grupos, añadir filas a las tablas, etc.

Esta API, no es necesaria para desarrollar la solución adoptada, ya que el *dataset* que contiene los datos se crea automáticamente. Además, esta API tiene muchas limitaciones, ya que actualmente (2016) le faltan aspectos como tener la capacidad de elegir qué borrar de una tabla. Únicamente es posible borrar todo su contenido. Es un problema, ya que en la ejecución de la solución adoptada, si no se borra periódicamente el contenido de las tablas, los datos se irán acumulando en las tablas hasta que llegue al límite de memoria que permite *Power BI*. Posteriormente, se rechaza cualquier nueva información entrante.

Para poder utilizar esta API en un proyecto, se puede importar fácilmente las librerías necesarias utilizando la consola de **NuGet** (gestor de paquetes) en Visual Studio (*Tools > NuGet Package Manager > Package Manager Console*), ejecutando el siguiente comando:

```
Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory
```

A partir de aquí, se puede gestionar programáticamente, en la medida de lo posible, los recursos asociados a la cuenta de *Power BI*.

#### 5.1.6.6. DocumentDB y Azure SQL

A modo de resumen, se define a continuación cada uno de estos servicios, ya que han sido una pequeña parte de la dedicación, pero que no se han incluido en la solución adoptada.

**DocumentDB** es un servicio ofrecido por la plataforma de *Azure*, el cual consiste en un servicio de bases de datos NoSQL basados en documentos JSON. Consiste en un conjunto de colecciones de objetos que se almacenan en formato JSON. Permite que estos objetos puedan evolucionar con el paso del tiempo, sin tener problemas de retrocompatibilidad con las aplicaciones que se desarrollen utilizando este servicio.

**Azure SQL** es un servicio ofrecido por la plataforma de *Azure*, el cual consiste en un servicio de bases de datos relacionales, con una tecnología basada en *SQL Server*. Este servicio es altamente escalable y disponible, lo que facilita el despliegue de bases de datos en la nube. Ofrece bases de datos seguras y accesibles a través de Internet para cualquier aplicación o servicio que la requiera.



## 5.2. Office 365

---

### 5.2.1. ¿Qué es Office 365?

*Office 365* es un servicio de suscripción, a través de una cuenta de *Microsoft*, que ofrece distintas herramientas web, accesibles a través de cualquier dispositivo y desde casi cualquier lugar. Entre otras cosas, es posible editar documentos, revisar el correo electrónico, realizar reuniones en línea...

Consiste en un servicio basado en la nube, útil para gestionar una organización, el cual es compatible con las herramientas más populares de *Microsoft*, como *Microsoft Word* o *Microsoft Excel*, y posibilita la creación de sitios web fácilmente. Además, da la posibilidad de colaborar con socios y clientes a través de dichos sitios.

Dispone de una gran cantidad de servicios (ver **ilustración 38**) con los que se puede extender, incorporando más funcionalidades de gestión y productividad; entre las cuales se encuentra *Power BI*. Este servicio se define a continuación.



Ilustración 38. Servicios de Office 365 (2016)

### 5.2.2. Power BI

**Power BI** (*BI, Business Intelligence*) es un conjunto de aplicaciones de análisis de negocio que permite analizar datos y compartir información. El [dashboard web de Power BI](#), es un tablero de herramientas desde el cual se puede gestionar la información que recibe o tiene almacenada, y así, dotarla de varias interpretaciones visuales.

El *dashboard* está dividido en grupos de trabajo, los cuales pueden ser, por ejemplo, distintos departamentos de una organización, donde en cada uno de ellos, la información que se gestiona y visualiza es de distintos ámbitos. Por defecto, se puede encontrar un grupo de trabajo denominado “Mi área de trabajo”. La creación de grupos sólo está soportada en el caso de actualizarse a la versión “*Power BI Pro*”. Existe la posibilidad de probar esta versión de forma gratuita durante 60 días. Después de que expire esta, hay dos opciones:

- Solicitar una extensión de 60 días (ver **ilustración 39**). La extensión será posible si aprueban la solicitud.
- Comprar la versión completa por pago mensual (por usuario u organizacional).

▲ Su prueba gratuita expiró. Solicite una extensión de 60 días para continuar usando las características de Power BI Pro.

#### Ilustración 39. Posibilidad de extender la prueba de 60 días de *Power BI Pro*

Cada grupo de trabajo puede contener múltiples paneles y *datasets*.

La definición de “*dataset*”, en este apartado, no varía de la que se utiliza en el resto del documento, salvo por un detalle: en lugar de ser una única tabla donde se organizan los datos, es un contenedor de tablas (como mínimo una).

La creación de un *dataset* puede realizarse de diversas maneras:

- Examinando paquetes de contenido que otros socios de la organización hayan publicado.
- Paquetes de contenido de servicios en línea que se utilizan.
- Importar datos como informes, libros, datos en *Excel* o archivos con extensión ‘.csv’.
- Datos activos, como por ejemplo, bases de datos SQL de *Azure*.
- Como se ha indicado en el apartado [apartado 5.1.3.2. \(Receptores de resultados\)](#), se puede hacer que *Azure Stream Analytics* devuelva los resultados de datos analizados por un *job* a la tabla de un *dataset*, dentro de un grupo de trabajo concreto. El *dataset* y la tabla se generan automáticamente cuando *Azure Stream Analytics* comienza a enviar resultados.

Un **panel** es una página donde se pueden incluir múltiples informes y visualizaciones.

Un **informe** es un componente visual, que se puede incluir dentro de un panel, desde el cual, se puede crear, editar, actualizar y borrar visualizaciones alimentadas por una o varias tablas de un *dataset*.

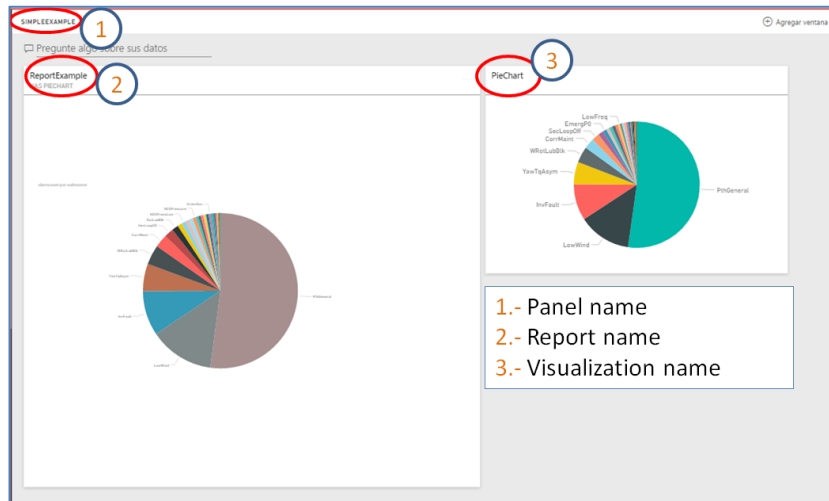
Una **visualización** es un componente visual, que muestra de una forma determinada la información almacenada en una o varias tablas de un *dataset*. Es capaz de actualizarse en tiempo real, a causa de la actualización de los datos almacenados en las tablas que lo alimentan. Esto es posible siempre y cuando dicha visualización se incluya (o se ancle) en un panel.

A continuación, se expone un ejemplo para entender esta estructura de visualización.

Suponiendo que se está trabajando en el grupo denominado “*Mi área de trabajo*”, en él se añade un nuevo panel llamado “*SIMPLEEXAMPLE*”. En ese panel

se crea un informe llamado “*ReportExample*”, donde se utiliza una tabla de un *dataset* (el cual se ha generado previamente), para crear una visualización denominada “*PieChart*”.

Por último se ancla al panel tanto el informe como la visualización. El resultado de este pequeño laboratorio se muestra en la **ilustración 40**.

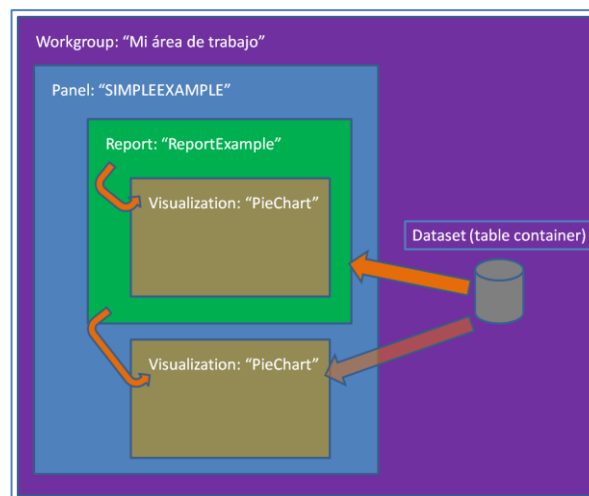


**Ilustración 40. Resultado del ejemplo de visualización descrito**

En este caso, la visualización que se encuentra dentro del informe y la que se encuentra fuera es la misma.

Sin embargo, hay una pequeña diferencia entre ambas. Como se ha descrito anteriormente, la visualización que está anclada directamente al panel (la que está fuera del informe), tiene la capacidad de actualizarse en tiempo real, mientras que la otra (que está anclada al panel a través de un informe), no podrá actualizarse en tiempo real, ya que los informes no tienen esa capacidad.

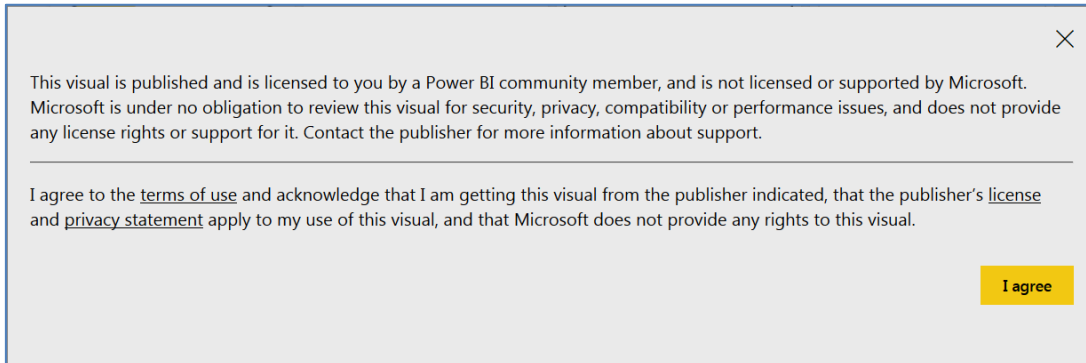
En la **ilustración 41**, se muestra un esquema de la estructura definida en este ejemplo.



**Ilustración 41. Estructura del ejemplo de visualización descrito**

Además, *Power BI* permite añadir visualizaciones creadas por la [comunidad](#), importando los modelos de estos (ficheros con extensión *.pbviz*).

A la hora de descargarlos, *Microsoft* avisa de que no está obligado a revisar la seguridad y compatibilidad de estas visualizaciones.



**Ilustración 42. Aviso en la descarga de una visualización desarrollada por la comunidad**

Una vez descargados, se pueden importar desde el portal de *Power BI*, cuando van a ser utilizados en algún informe.

Esto es, en términos generales, cómo trabaja internamente esta tecnología para la visualización de datos y las posibilidades que ofrece.

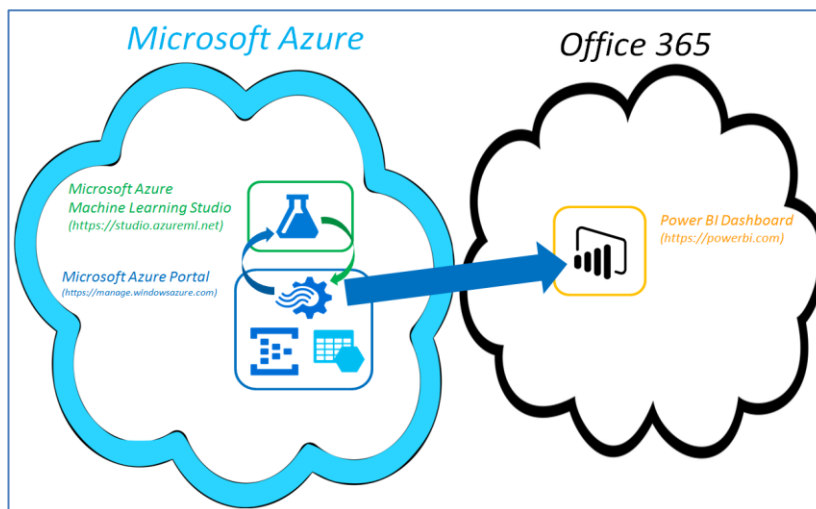
### 5.3. Relación entre *Microsoft Azure* y *Office 365*

En los apartados anteriores se describen *Microsoft Azure* ([apartado 5.1.](#)) y *Office 365* ([apartado 5.2.](#)), los cuales pueden conectarse entre sí.

Como se muestra en la **ilustración 43**, *Azure Stream Analytics* puede devolver información procesada a *Power BI*, según se indica en el [apartado 5.1.3.2. \(Receptores de resultados\)](#).

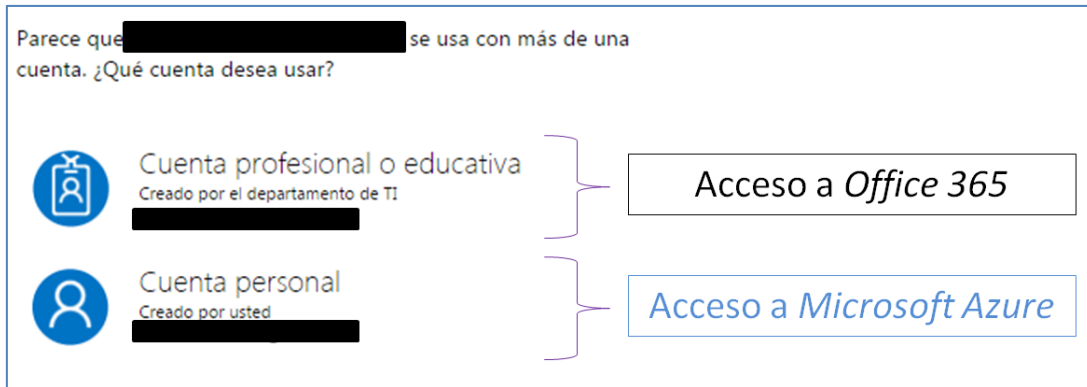
Además, *Azure Stream Analytics* puede intercambiar información con otro servicio de *Microsoft Azure* que no es accesible directamente desde el mismo *dashboard* que este; el servicio *Machine Learning*. Puede hacerlo a través de una característica llamada **función**, la cual se introduce en el [apartado 5.1.3.3. \(Consulta de transformación\)](#).

La siguiente ilustración muestra de manera más gráfica la relación que hay entre las distintas herramientas y tecnologías descritas.



**Ilustración 43.** Relación entre servicios de *Microsoft Azure* y *Office 365*

Otra característica importante son las cuentas de acceso a los distintos servicios. Para acceder a los servicios de *Microsoft Azure*, se debe utilizar una **cuenta personal** de *Microsoft*. En cambio, para acceder al servicio de *Office 365*, requiere de una **cuenta profesional o educativa** de *Microsoft* (ver **ilustración 44**).



**Ilustración 44.** Cuentas *Microsoft* necesarias para acceder a los servicios descritos

El aspecto de tener que acceder a dos cuentas de *Microsoft* trae consigo un inconveniente, y es que para realizar ambos accesos simultáneamente (por ejemplo, cuando se está desarrollando una solución en la plataforma), requiere utilizar dos navegadores; uno por cada cuenta. Esto se debe a que, en este caso, ambas cuentas están asociadas al mismo correo, y no se puede acceder con la misma cuenta a distintos conjuntos tecnológicos, debido a que estas tienen propósitos distintos.

Entonces, según la **ilustración 43**, a *Microsoft Azure Portal* y *Microsoft Azure Machine Learning Studio*, a pesar de que estén en dominios distintos, puede accederse simultáneamente a través de un navegador. Sin embargo, para acceder al *dashboard* de *Power BI*, será necesario utilizar otro navegador.

En resumen, *Azure Stream Analytics* (servicio accesible a través de *Microsoft Azure Portal*) es capaz de comunicarse con el servicio de desarrollo y despliegue de experimentos predictivos denominado *Machine Learning*, y devolver datos a un servicio de visualización de información denominado *Power BI*, el cual no se encuentra dentro de la misma “nube” que el resto de tecnologías y servicios.

---

## 6. Solución utilizando tecnologías de *Microsoft*

---

Este apartado describe cuál es la solución adoptada para el problema inicial utilizando tecnologías de *Microsoft*, ofreciendo diagramas y descripción de otras características propias de la mencionada solución.

### 6.1. Descripción de una solución al problema inicial

---

Con el fin de adaptar la solución realizada previamente por el equipo de IK4-Ikerlan sobre la monitorización predictiva de los aerogeneradores, se describe en la **ilustración 45** el esquema general sobre la solución propuesta utilizando servicios y tecnologías provistas por la nube de *Microsoft*. Las tecnologías y servicios se han estudiado y seleccionado en función de las necesidades detectadas, en referencia a la solución original.

La mayoría de tecnologías que se utilizan en dicho esquema son muy similares a las tecnologías de software libre utilizadas en la solución original, por lo que pueden ser una sustitución directa dentro de la plataforma de *Microsoft Azure*. Por ejemplo, *Azure Event Hubs* es un equivalente a [Apache Kafka](#).

Sin embargo, otros aspectos de la plataforma son dispares a los de la solución original, y de alguna manera, ha sido necesario adaptarlo al contexto, a través de la solución propuesta. Por ejemplo, el preprocesado para el sistema predictivo se ha desarrollado de distinta forma en ambos casos, a pesar de que los pasos que siguen son los mismos (ver **ilustración 48**).

Por ello, la solución dispone de un sistema para la ingesta de la información, procesamiento de la información recibida, preprocesamiento de información para la puesta en marcha de un sistema predictivo y visualización de los resultados. En conjunto, es un sistema que trabaja en tiempo real.

### 6.2. Objetivo de las tecnologías utilizadas

---

En este apartado se describe el papel que desempeñan los servicios más importantes, utilizados para desarrollar la solución adoptada.

#### 6.2.1. Aplicación emisora de datos (simulador de un aerogenerador)

Esta solución, dispone de un **cliente de aplicación de consola** (escrita en el lenguaje de programación *C#* y desarrollada con el IDE gratuito [Visual Studio 2015](#)

*Community*), el cual utiliza varias API (algunas de las descritas en el [apartado 5.1.6.](#)) para acceder a servicios de *Microsoft*. Su objetivo principal es simular **un aerogenerador** (consultar el [apartado 7.2.](#)), que envía sus datos operacionales cada cierto tiempo a la plataforma, con el objetivo de que este sea monitorizado (según se describe en el [apartado 2.1.](#)).

### 6.2.2. Azure Event Hubs (gestión de la información)

Servicio descrito en el [apartado 5.1.2.](#) En esta solución, se encarga de **recibir la información que envía el cliente desarrollado**, y actuar como fuente de datos para el servicio *Azure Stream Analytics* con el objetivo de que dichos datos sean procesados.

### 6.2.3. Azure Stream Analytics (análisis y gestión de la información)

Servicio descrito en el [apartado 5.1.3.](#) En esta solución, es uno de los servicios fundamentales, ya que se encarga de **comunicar el servicio de Machine Learning** (servicio donde se genera y despliega como servicio web el sistema predictivo) y **el servicio Power BI** (servicio para visualizar los datos procesados) con los **datos recibidos de Azure Event Hubs**. Trabaja en conjunto con *Azure Blob Storage* para almacenar información de control relacionada con *Azure Event Hubs*.

### 6.2.4. Machine Learning (creación y uso del sistema predictivo)

Servicio descrito en el [apartado 5.1.4.](#) En esta solución, es el encargado de **generar y hacer disponible a través de un servicio web el sistema predictivo desarrollado**. Se desarrollan experimentos, tanto para la ejecución del preprocesado como para la puesta en marcha del sistema predictivo. Trabaja en conjunto con *Azure Blob Storage* para almacenar información procesada.

### 6.2.5. Azure Blob Storage (persistencia de información de control y preprocesado)

Servicio descrito en el [apartado 5.1.5.](#) En esta solución, se utiliza para **almacenar los resultados obtenidos de un procesamiento intermedio**, y poder hacer uso posterior de los mismos. Además, se utiliza de una forma pasiva para la gestión de datos de control relacionados con el espacio de nombres utilizado en la solución (ver [apartado 5.1.3.](#)).

### 6.2.6. Power BI (visualización de los datos procesados)

Servicio descrito en el [apartado 5.2.2.](#) En esta solución, se encarga de **visualizar la información procesada** que devuelve *Azure Stream Analytics*. Consiste en la puesta en marcha de visualizaciones que trabajan en tiempo real.





### 6.4. Diagrama de componentes

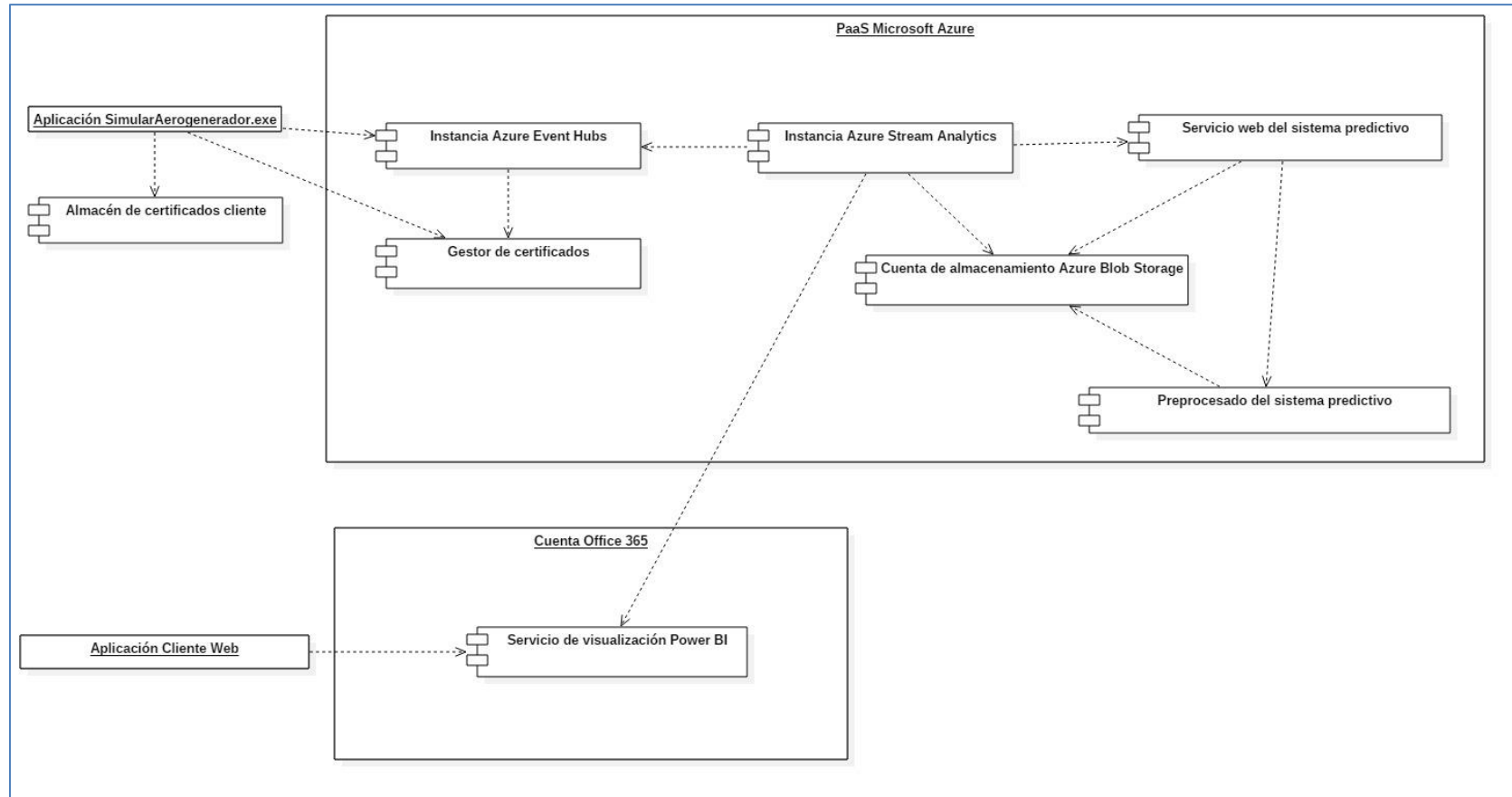


Ilustración 46. Diagrama de componentes

## 6.5. Diagrama de secuencia

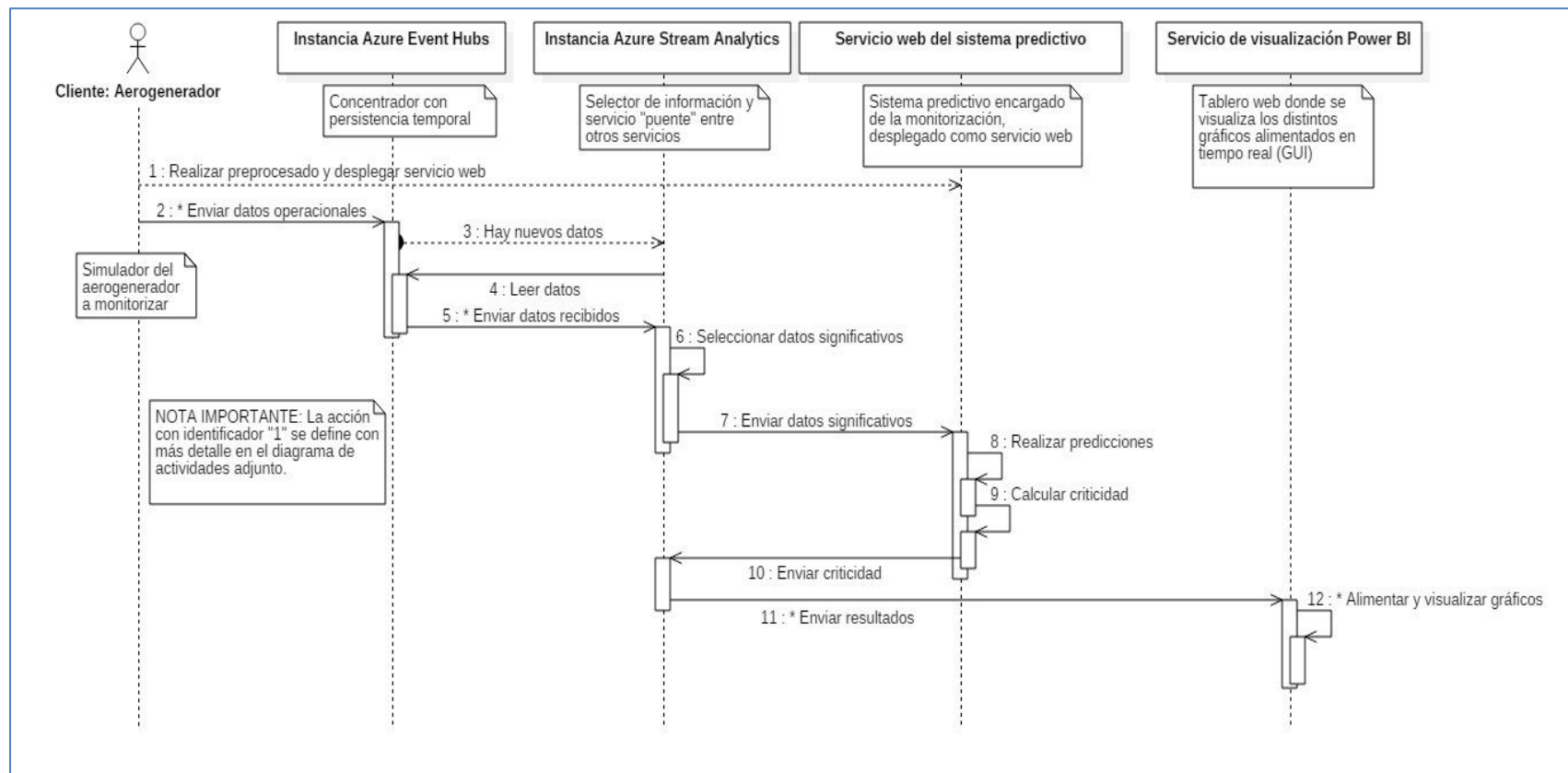


Ilustración 47. Diagrama de secuencia

## 6.6. Diagrama de actividades

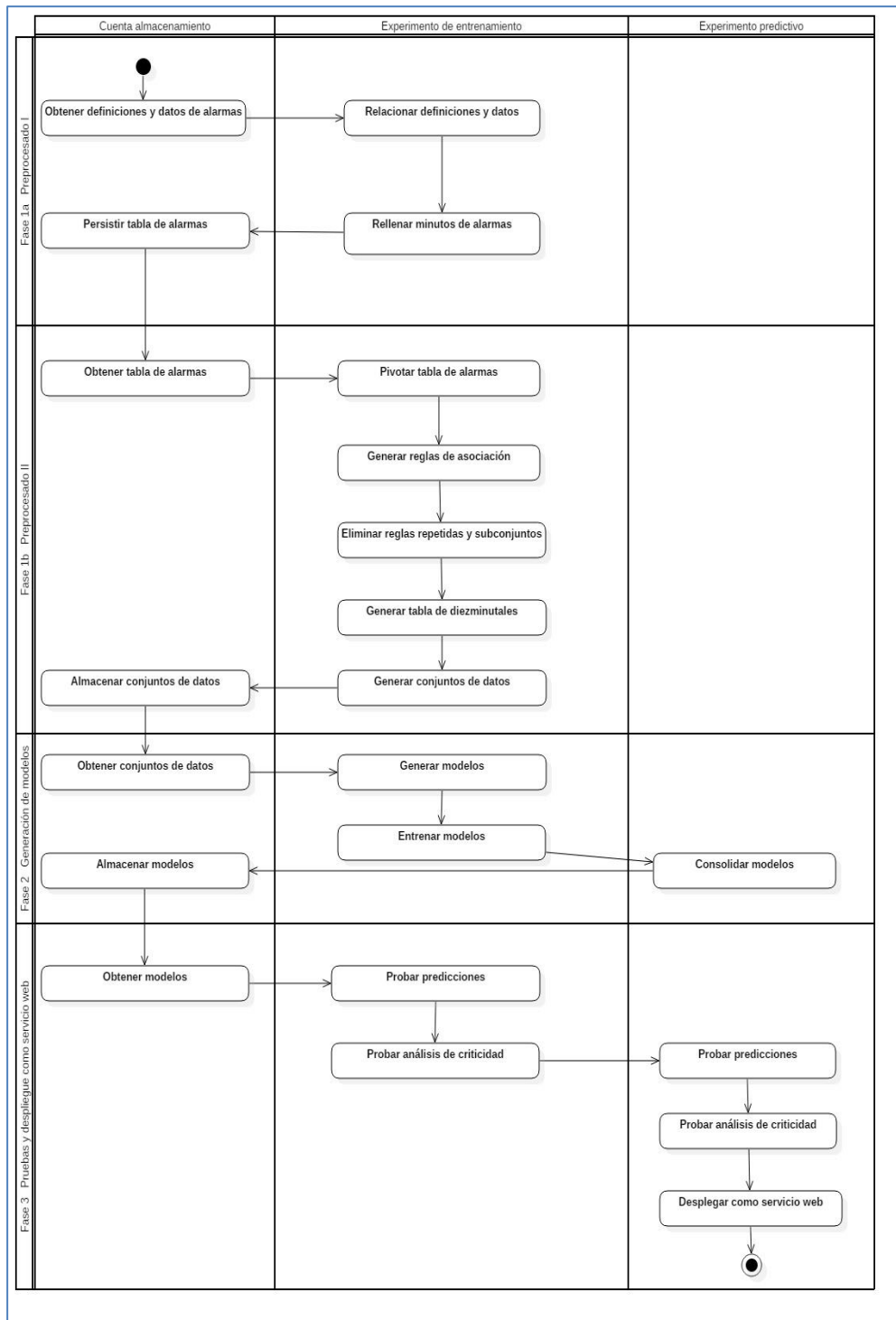


Ilustración 48. Diagrama de actividades (preprocesado y despliegue del sistema predictivo)

## 6.7. Requisitos no funcionales

---

Algunos de los requisitos no funcionales que más se tienen en cuenta en el desarrollo de la solución adoptada son:

- **Rendimiento** de las tecnologías y servicios.
- **Disponibilidad** de las tecnologías y servicios.
- **Portabilidad** de las configuraciones realizadas en la plataforma.
- **Escalabilidad** de las tecnologías y servicios.
- **Costo** en la implantación y uso de las tecnologías y servicios.
- **Seguridad** en las comunicaciones entre emisores y la plataforma.

## 7. Implementación de la solución

En este apartado se describen los pasos realizados para implementar la solución adoptada y otros aspectos relacionados con esta.

### 7.1. Desarrollo de la solución

A continuación, se explica con más detalle el desarrollo de la solución adoptada. Hay que tener en cuenta que lo descrito a continuación siempre hace referencia al esquema general de la solución (**ilustración 45**).

#### 7.1.1. Azure Event Hubs (gestión de la información)

Lo primordial en esta parte de la implementación consiste en la conexión entre el emisor de datos y el receptor de los mismos en la nube, con el resto de componentes.

Por lo tanto, lo primero es crear el espacio de nombres donde debe residir y utilizarse el *Event Hub* receptor. Para hacerlo, se accede al [portal de Azure](#) con una “cuenta personal” (es posible crear una y usarla de forma gratuita para realizar pruebas durante un corto periodo de tiempo), y se crea un espacio de nombres siguiendo los pasos de la **ilustración 49**.

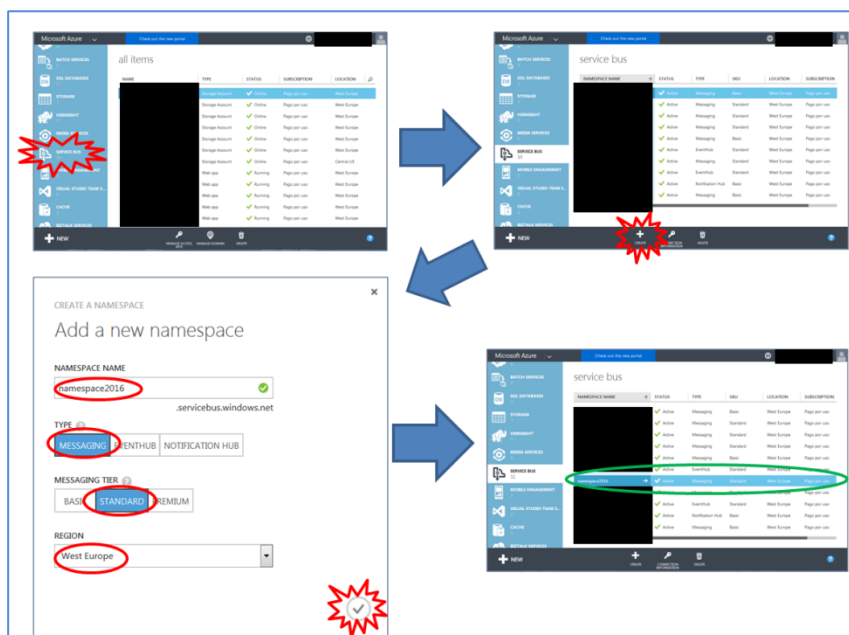


Ilustración 49. Proceso para crear un espacio de nombres

Una vez hecho esto, se procede a crear el *Event Hub*, dentro de dicho espacio de nombres.

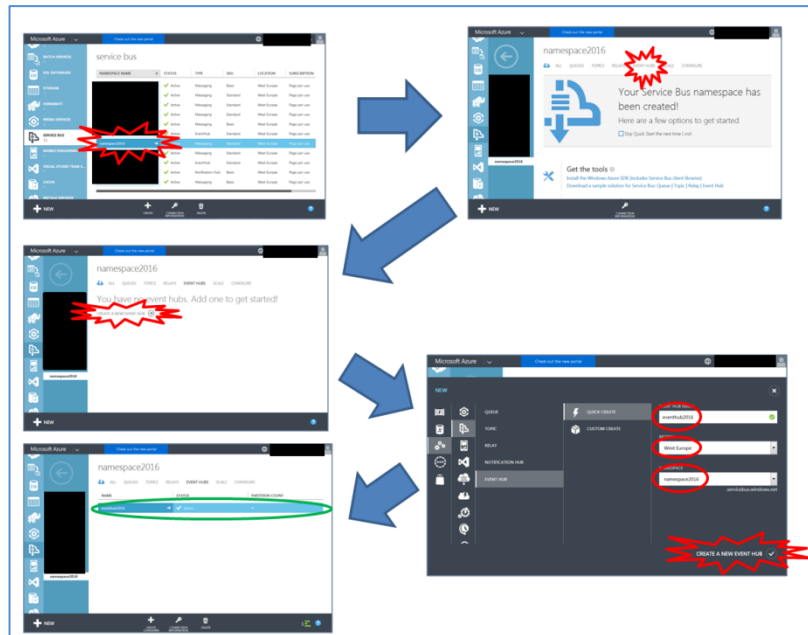


Ilustración 50. Proceso para crear un *Event Hub*

Una vez creado, se definen las políticas SAS para poder hacer uso del mismo. En este caso, se deben definir por lo menos dos: una con derechos de envío y otra con derechos de gestión. La primera para que, el publicador que se encarga de enviar los datos, sea capaz de interactuar con el *Event Hub*, y el segundo, para que un *job* de *Azure Stream Analytics* sea capaz de obtener la información que el *Event Hub* retiene. **Es un requisito que la política de acceso que use un *job* tenga derechos de gestión.** Al crear estas políticas, se generan las claves SAS o claves de acceso compartido automáticamente.

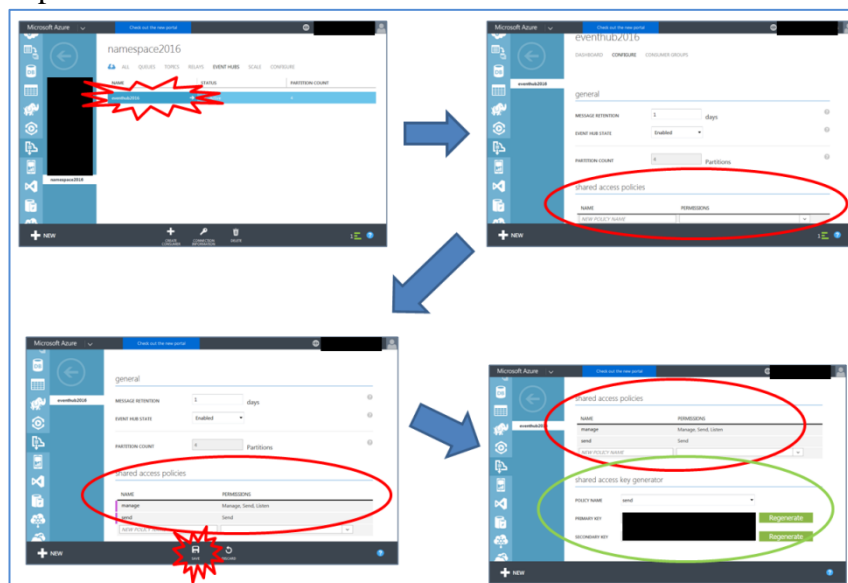
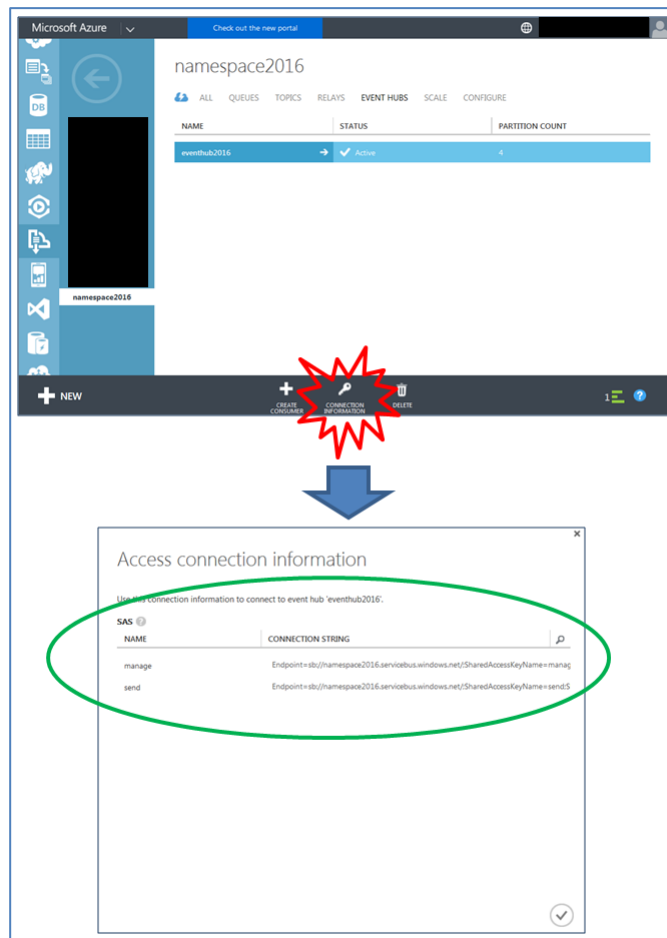


Ilustración 51. Proceso para añadir políticas de acceso al *Event Hub*

Una vez creadas las políticas de acceso, se puede acceder a las cadenas de conexión generadas por las políticas. Estas cadenas son las que se utilizan para poder realizar la conexión con el *Event Hub*.



**Ilustración 52. Consultar cadenas de conexión de un *Event Hub***

Una cadena de conexión tiene el siguiente formato:

```

“
Endpoint=sb://<NOMBRE_ESPACIO_NOMBRES>.servicebus.windows.net/;
SharedAccessKeyName=<NOMBRE_POLÍTICA_ACCESO>;
SharedAccessKey=<CLAVE_ACCESO_COMPARTIDO_PRIMARIA>;
EntityPath=<NOMBRE_EVENT_HUB>
”
    
```

Una vez realizadas estas configuraciones, el *Event Hub* creado es accesible.

### 7.1.2. Machine Learning (creación y uso del sistema predictivo)

Cabe decir que para la realización de esta parte del proyecto, se ha necesitado aprender y enriquecerse en el lenguaje de programación R, ya que tanto el preprocesado como la creación de los modelos predictivos se desarrollan en



experimentos, **relacionando módulos de ejecución de código en R**. La característica principal de estos módulos es su acción: ejecuta un script en R sobre el o los *dataset* que obtiene de entrada, devolviendo el procesamiento resultante. No se introduce el código desarrollado, pero se explica cómo funcionan los experimentos realizados para esta parte de la implementación. Además, la mayor parte del código desarrollado, previamente se ha elaborado en un entorno de desarrollo local, con el objetivo de realizar pruebas controladas de su funcionamiento, utilizando la herramienta [RGui](#).

Previo al comienzo del desarrollo de los experimentos, es necesario obtener unos *dataset* de prueba, obteniendo algunos datos como las alarmas de un aerogenerador, las definiciones de dichas alarmas y los datos operacionales producidos por el aerogenerador mencionado. En este caso, se extraen los datos mencionados de una base de datos de IK4-Ikerlan, la cual tiene almacenados los datos monitorizados de distintos aerogeneradores. Estos datos se deben extraer haciendo que los datos extraídos no sean los auténticos. Este aspecto es importante por dos razones:

Los datos que se utilizan en la plataforma pública no sean iguales a los originales, ya que no se dispone del permiso de la empresa propietaria de los datos para publicarlos.

Debido al punto anterior, no se puede esperar que las predicciones realizadas tengan coherencia, ya que las modificaciones sobre los datos no se hacen de una forma uniforme o reversible. Es decir, la alteración es aleatoria.

Otro aspecto a tener en cuenta es que, los pasos que se describen a continuación, son los pasos descritos en la solución ya existente, para realizar el preprocesado y puesta en marcha de los modelos predictivos.

Por último, los *dataset* que se muestren en las imágenes, no son los resultados completos de las operaciones, ya que los *dataset* (tanto resultados como fuentes) son demasiado grandes como para mostrarse en una sola imagen. Por eso, se muestra una fracción de los mismos.

Teniendo en cuenta estos aspectos, es posible continuar con la explicación de los experimentos realizados.

Consta de las siguientes fases:

[Fase 1a: Rellenar los minutos de las alarmas generadas \(experimento no predictivo\)](#)

[Fase 1b: Generación de los conjuntos de datos \(experimento no predictivo\)](#)

[Fase 2: Entrenamiento de los modelos predictivos](#)

[Fase 3: Experimento de monitorización](#)

### 7.1.1.1. Fase 1a: Rellenar los minutos de las alarmas generadas (experimento no predictivo)

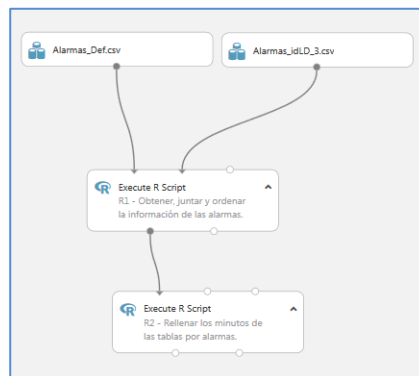


Ilustración 53. Experimento de desarrollo de la fase 1a

En la **ilustración 53** se muestran dos *dataset* de entrada: definición de las alarmas y alarmas generadas por un aerogenerador concreto, ambos *dataset* originarios de ficheros *.csv*. Estos *dataset* son la entrada de un módulo denominado **R1** (**ilustración 53**), el cual se encarga de relacionar las definiciones de las alarmas con las alarmas generadas por el aerogenerador. Al hacer esto, obtiene un *dataset* donde se sustituyen los códigos de alarmas, por su nombre original.

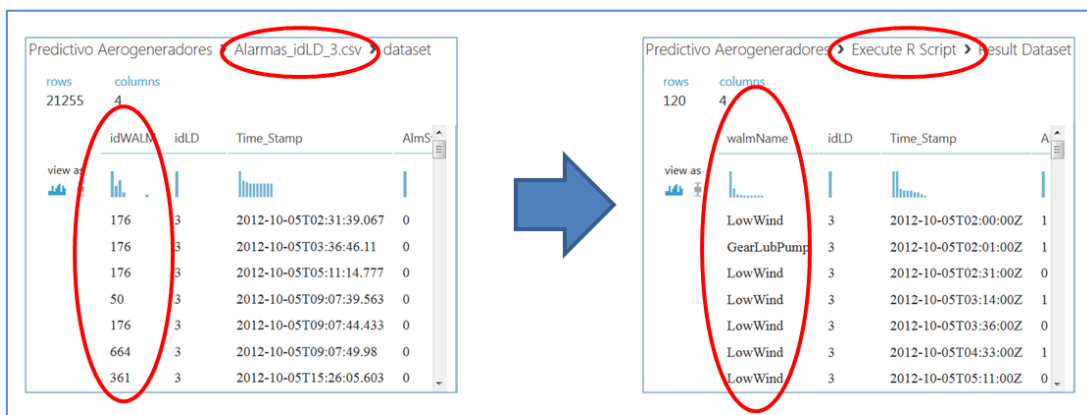
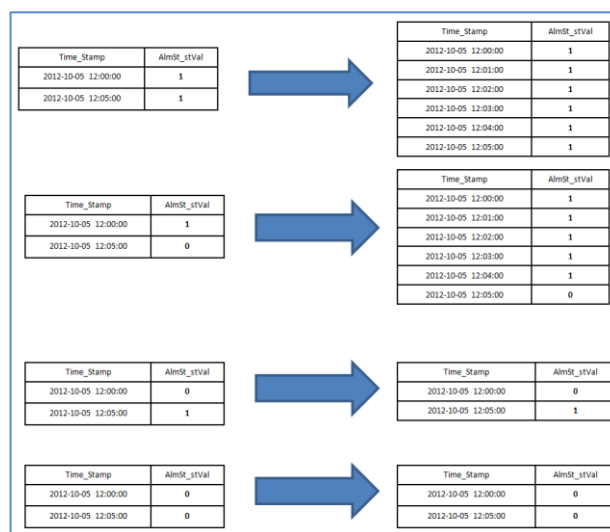


Ilustración 54. Datos originales (izquierda) y la sustitución realizada tras ejecutar el *script* (derecha)

Cabe destacar que en esta fase se hace una notable reducción de datos a utilizar, ya que el procesamiento descrito a continuación, sin un clúster para procesar toda esa información, el tiempo de ejecución sería muy elevado, y por lo visto, la máquina que ejecuta estos experimentos dispone de únicamente dos núcleos (verificable ejecutando en un módulo de R el comando “*detectCores()*” y comprobar el *log* de este. La función pertenece a la librería “*parallel*”). A pesar de esto, no es posible ejecutar código paralelizado en un experimento (ver apartado 7.2.3.). Por tanto, el procesamiento de prueba (para esta fase) se realiza con datos de **15 días** sobre ese aerogenerador. Son datos suficientes para ver que la ejecución se realiza correctamente y sin tardar demasiado.

A continuación, con el *dataset* obtenido en el paso anterior, se ejecuta el módulo **R2 (ilustración 53)** para rellenar los minutos restantes entre cada registro y por alarma, siguiendo las siguientes condiciones (explicadas también en la **ilustración 55**):

- Entre dos registros, si ambos tienen indicado que la alarma está activa, rellenar los minutos entre ambos registros.
- Entre dos registros, si el primero indica la alarma está activa y el segundo no, rellenar los minutos entre ambos registros.
- En cualquier otro caso, no rellenar los minutos.



**Ilustración 55. Comparaciones a realizar para el relleno de los minutos**

Este paso se realiza por cada una de las alarmas generadas. El *dataset* resultante tras este procesado tiene un aspecto similar al de la ilustración siguiente:

walmName	idLD	Time_Stamp	AlmSt_stVal
	3	2012-10-05T02:00:00	1
LowWind	3	2012-10-05T02:01:00	1
GearLubPump	3	2012-10-05T02:01:00	1
LowWind	3	2012-10-05T02:02:00	1
GearLubPump	3	2012-10-05T02:02:00	1
LowWind	3	2012-10-05T02:03:00	1
GearLubPump	3	2012-10-05T02:03:00	1

**Ilustración 56. Resultado de rellenar los minutos de todas las alarmas generadas**

Este es el resultado de la fase 1a, en la cual se obtiene un *dataset* donde se sabe, por cada minuto, que alarmas se han activado para un aerogenerador concreto.

### 7.1.1.2. Fase 1b: Generación de los conjuntos de datos (experimento no predictivo)

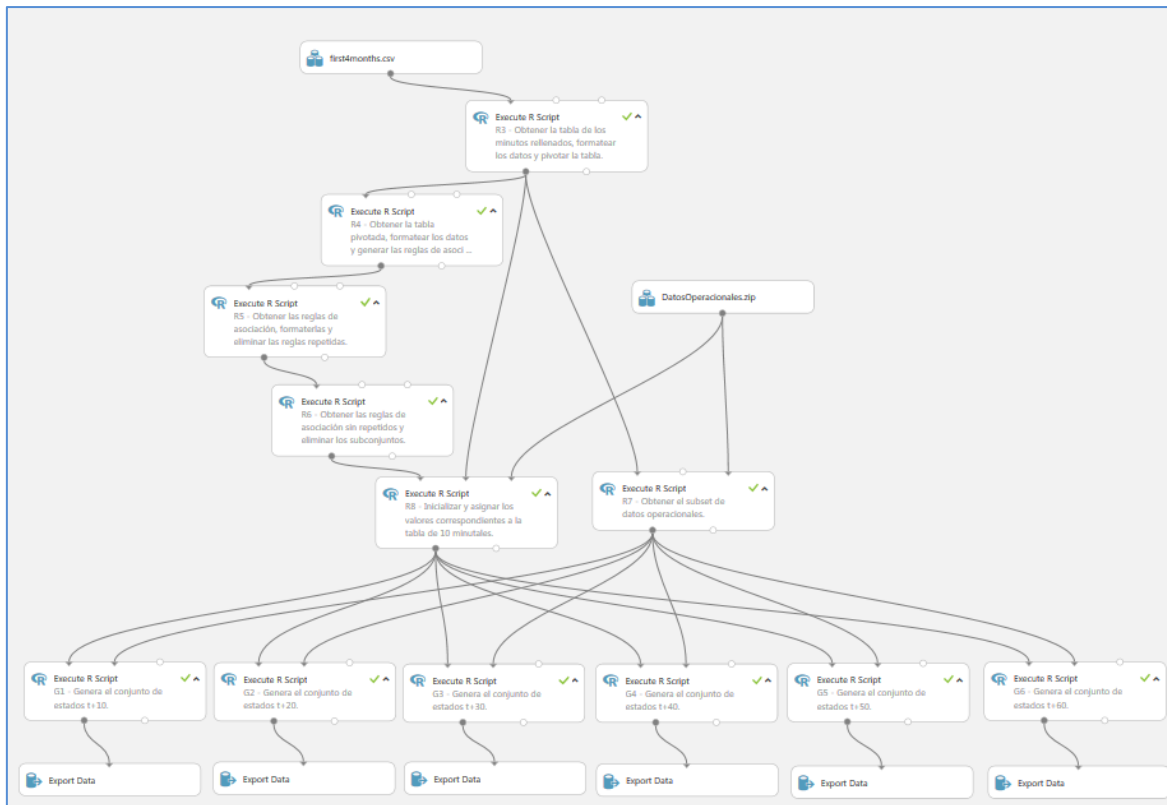


Ilustración 57. Experimento de desarrollo de la fase 1b

En teoría, esta fase y la anterior deberían estar unidas. La razón de separarlas es debido a que la ejecución de la primera fase es muy lenta.

Por eso, en esta fase, se ha optado por subir al portal un *dataset* procesado, según las especificaciones de la fase anterior. Este *dataset* se ha generado utilizando código paralelizado en la máquina local.

De esta forma, con un tiempo de procesamiento similar, se obtiene una mayor cantidad de datos procesados. En este caso, el *dataset* de entrada es de **4 meses** de datos procesados.

El módulo **R3** (ilustración 57) se encarga de pivotar la tabla procesada en la fase 1a, de tal forma que cada una de las filas muestra un *timestamp* donde se sabe que alarmas se han activado en dicho momento.

walmName	Time_Stamp	AlmSt_stVal
LowWind	2012-10-05 12:00:00	1
LowWind	2012-10-05 12:01:00	1
GearLubPump	2012-10-05 12:01:00	1
LowWind	2012-10-05 12:02:00	1

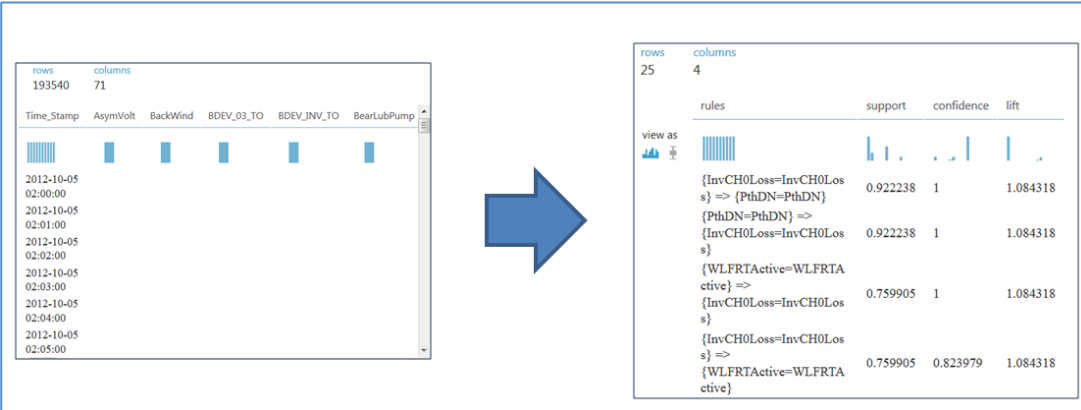
Time_Stamp	GearLubPump	LowWind
2012-10-05 12:00:00		LowWind
2012-10-05 12:01:00	GearLubPump	LowWind
2012-10-05 12:02:00		LowWind

**Ilustración 58. Pivotar la tabla por timestamp**

Para pivotar la tabla, se utiliza una función denominada “dcast()” (librería “*reshape2*”), con el que se indica en función de qué parámetros de la tabla se va a pivotar y que valores van a tomar.

El módulo **R4 (ilustración 57)**, se encarga de generar las reglas de asociación a partir de la tabla pivotada, obtenida en el módulo R3 (ver **ilustración 59**).

Para generar estas reglas, se utiliza una función denominada “apriori()” (librería “*arules*”), con la que, aportando unos parámetros (mínimo de dos alarmas, soporte mínimo de 0.5 y grado de confianza mínimo de 0.8), el algoritmo de reglas de asociación los tendrá en cuenta para encontrar ciertos hechos en común en el *dataset* obtenida en el módulo R3, es decir, combinaciones de alarmas que con más intensidad se repiten en el conjunto de datos aportado.



Time_Stamp	AsymVolt	BackWind	BDEV_03_TO	BDEV_INV_TO	BearLubPump
2012-10-05 02:00:00					
2012-10-05 02:01:00					
2012-10-05 02:02:00					
2012-10-05 02:03:00					
2012-10-05 02:04:00					
2012-10-05 02:05:00					

rules	support	confidence	lift
{InvCH0Loss=InvCH0Loss} => {PthDN=PthDN}	0.922238	1	1.084318
{PthDN=PthDN} => {InvCH0Loss=InvCH0Loss}	0.922238	1	1.084318
{WLFRTActive=WLFRTActive} => {InvCH0Loss=InvCH0Loss}	0.759905	1	1.084318
{InvCH0Loss=InvCH0Loss} => {WLFRTActive=WLFRTActive}	0.759905	0.823979	1.084318

**Ilustración 59. De la tabla pivotada (izquierda), obtener las reglas de asociación (derecha)**

No son necesarias todas las reglas de asociación generadas, por lo tanto, además de darles un formato, en el módulo **R5 (ver ilustración 57)**, se eliminan las reglas repetidas. Por ejemplo, una regla {A,B,C} y otra {B,C,A}, en términos de comparación, son la misma regla, ya que ambos tienen los mismos elementos. Esto ocurre, por ejemplo, con las dos primeras reglas de asociación generadas en la **ilustración 60 (izquierda)**.



rules	support	confidence	lift
{InvCH0Loss=InvCH0Loss} => {PthDN=PthDN}	0.922238	1	1.084318
{PthDN=PthDN} => {InvCH0Loss=InvCH0Loss}	0.922238	1	1.084318
{WLFRTActive=WLFRTActive} => {InvCH0Loss=InvCH0Loss}	0.759905	1	1.084318
{InvCH0Loss=InvCH0Loss} => {WLFRTActive=WLFRTActive}	0.759905	0.823979	1.084318

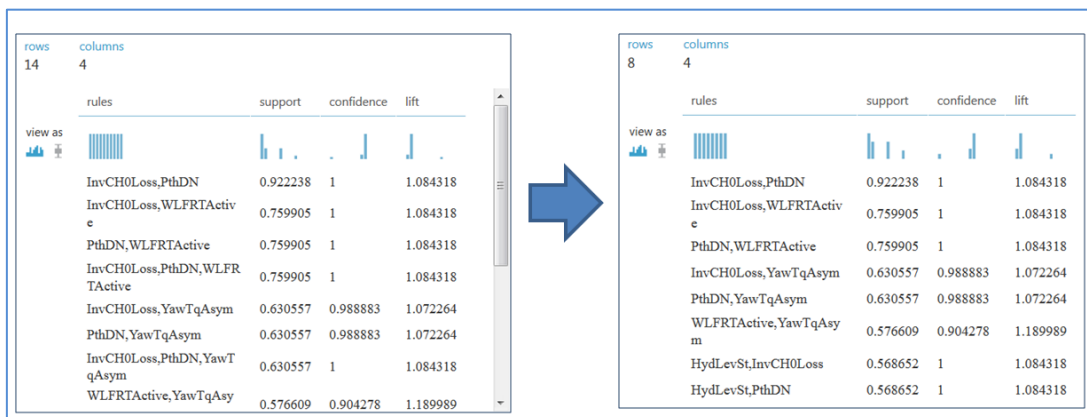
rules	support	confidence	lift
InvCH0Loss,PthDN	0.922238	1	1.084318
InvCH0Loss,WLFRTActive	0.759905	1	1.084318
PthDN,WLFRTActive	0.759905	1	1.084318
InvCH0Loss,PthDN,WLFRTActive	0.759905	1	1.084318
InvCH0Loss,YawTqAsym	0.630557	0.988883	1.072264
PthDN,YawTqAsym	0.630557	0.988883	1.072264
InvCH0Loss,PthDN,YawTqAsym	0.630557	1	1.084318
WLFRTActive,YawTqAsym	0.576609	0.904278	1.189989

**Ilustración 60. Eliminación de reglas de asociación repetidas**

Nótese que se hace la conversión entre el formato propio que utiliza R para generar reglas de asociación (“{A=A} => {B=B}”) y el formato que se busca (“A,B”), realizando a su vez la eliminación.

Además de esto, puede que una regla sea el subconjunto de otra. Por tanto, el módulo **R6 (ilustración 57)** se encarga de eliminar estos subconjuntos, según las siguientes condiciones:

- Si una regla es subconjunto de otra y el porcentaje de confianza entre ellas (parámetro “confidence”) es menor al 5%, se elimina la regla más grande (es decir, la que más alarmas involucre).
- En cualquier otro caso, ambas reglas se mantienen.



rules	support	confidence	lift
InvCH0Loss,PthDN	0.922238	1	1.084318
InvCH0Loss,WLFRTActive	0.759905	1	1.084318
PthDN,WLFRTActive	0.759905	1	1.084318
InvCH0Loss,PthDN,WLFRTActive	0.759905	1	1.084318
InvCH0Loss,YawTqAsym	0.630557	0.988883	1.072264
PthDN,YawTqAsym	0.630557	0.988883	1.072264
InvCH0Loss,PthDN,YawTqAsym	0.630557	1	1.084318
WLFRTActive,YawTqAsym	0.576609	0.904278	1.189989

rules	support	confidence	lift
InvCH0Loss,PthDN	0.922238	1	1.084318
InvCH0Loss,WLFRTActive	0.759905	1	1.084318
PthDN,WLFRTActive	0.759905	1	1.084318
InvCH0Loss,YawTqAsym	0.630557	0.988883	1.072264
PthDN,YawTqAsym	0.630557	0.988883	1.072264
WLFRTActive,YawTqAsym	0.576609	0.904278	1.189989
HydLevSt,InvCH0Loss	0.568652	1	1.084318
HydLevSt,PthDN	0.568652	1	1.084318

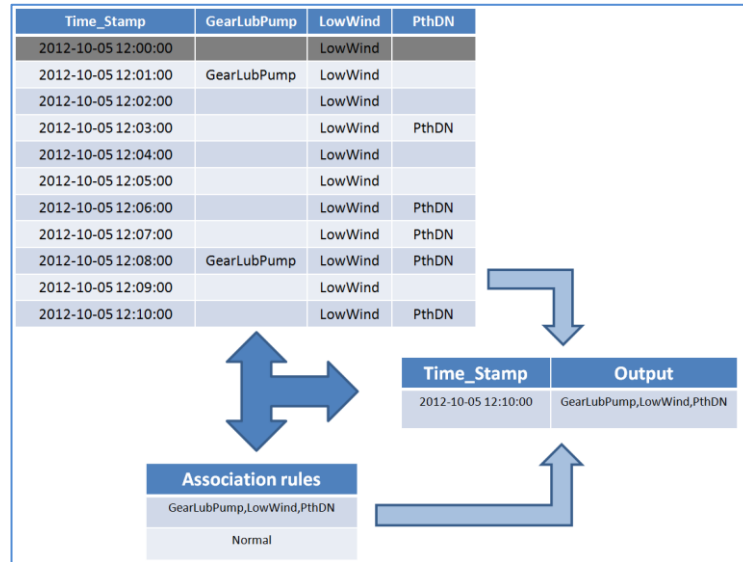
**Ilustración 61. Eliminación de subconjuntos de reglas de asociación**

Esto ocurre, por ejemplo, con la primera y cuarta regla de asociación, generadas en la **ilustración 61** (izquierda, la cuarta acaba desapareciendo).

En este punto, se ha terminado de filtrar las reglas de asociación redundantes e innecesarias. En la **ilustración 61**, la imagen de la derecha muestra las reglas de asociación que se utilizan en el resto del preprocesado.

A partir de aquí, son necesarios dos conjuntos de datos (además de las reglas de asociación): los datos operacionales del aerogenerador y los *output* generados por la búsqueda de reglas de asociación en la tabla pivotada.

Esta búsqueda se realiza en la tabla pivotada, comparando los registros de cada diez minutos con las reglas de asociación. En el caso de que alguna coincida, al diezminutal que corresponda ese registro se le asocia como *output* la regla de asociación coincidente. Si ninguna coincide, al *output* se le atribuye el valor “Normal”. Esto se repite para todos los rangos de diez minutos existentes en el conjunto de datos.



**Ilustración 62. Ejemplo de comparación y relación entre tabla pivotada y reglas de asociación**

Esta comparativa se realiza en el módulo **R8 (ilustración 57)**, donde además, se utiliza los datos operacionales como referencia para crear la tabla de diezminutales (se genera un *output* con su diezminutal en la **ilustración 62**).

Aquí surge un problema, y es que un módulo de ejecución R **sólo admite dos dataset** como entrada, cuando en este caso, se necesitan tres. Esto reduce la escalabilidad de la solución desarrollada, puesto que para solucionar esto, se procede a aprovechar la entrada extra de la que dispone, donde sólo admite comprimidos con datos subidos con anterioridad. Por lo tanto, se importa un archivo comprimido (.zip en este caso) con el *dataset* necesario, para incluirlo en el módulo de forma manual. El comando a añadir en el código para importar los datos en el entorno es el siguiente (importar previamente la librería “*data.table*” usando `library(data.table)`):

```
datosOperacionales <- fread("src/DatosOperacionales_E100_idLD_3.csv", na.strings=c("", "NA"))
```

Dado que el contenido del comprimido es el siguiente:

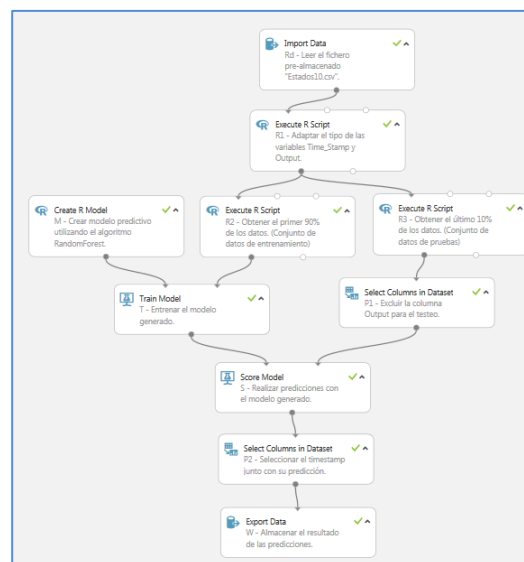
rows	columns		
1	4		
Contents	Full Path	Compressed Size	Uncompressed Size
DatosOperacionales_E100_idLD_3.csv	DatosOperacionales_E100_idLD_3.csv	5.7 MB	17.1 MB

**Ilustración 63. Contenido de "DatosOperacionales.zip"**

Aquí debe indicarse que, los datos operacionales que se incluyen en este fichero, no son todos los que un aerogenerador envía, sino los parámetros más significativos. Es decir, los parámetros cuya variabilidad es significativa para el resultado de su estudio. Para detectar qué parámetros son significativos, existe un estudio que se realizó con anterioridad por el equipo de IK4-Ikerlan para identificar estos parámetros. Por lo tanto, en este proyecto se toma como identificados estos parámetros y se dan por estudiados para el desarrollo del sistema predictivo.

El módulo **R7 (ilustración 57)** se encarga de obtener los datos operacionales para que puedan relacionarse con los *output* obtenidos en el módulo R8 posteriormente. En los módulos **G1 a G6 (ilustración 57)**, se relacionan los *output* y los datos operacionales, para generar los conjuntos de datos t+10 a t+60 respectivamente (el proceso es el indicado en la **ilustración 6**). Además, cada uno de estos módulos se encarga de almacenar los conjuntos de datos generados en un contenedor de *Azure Blob Storage*, con el formato “Estados10.csv” a “Estados60.csv” respectivamente, para su posterior uso.

### 7.1.1.3. Fase 2: Entrenamiento de los modelos predictivos



**Ilustración 64. Entrenamiento del modelo predictivo t+10**

En la **ilustración 64**, se muestra el desarrollo de un experimento de entrenamiento, que puede ser convertido en experimento predictivo, y que dispone de un componente para entrenar un modelo (el módulo T).

El módulo **Rd (ilustración 64)** se encarga de obtener el conjunto de datos de entrenamiento para, posteriormente, alimentar el modelo.

El módulo **R1 (ilustración 64)** se encarga de adaptar algunas variables para evitar errores en el proceso de entrenamiento del modelo. Por ejemplo, la variable que se pretende tomar como base a la hora de hacer las predicciones debe ser de tipo “factor”. En R, las variables de tipo factor son variables que conforman categorías.



Un ejemplo sencillo para entender este tipo de parámetros es la clase de un billete de avión (turista o *business*). Los valores asociados a la clase siempre van a tomar un valor u otro (no confundir con valores binarios, ya que los valores de tipo binario siempre se limitan a dos valores, mientras que los de tipo factor pueden ser más de dos).

A continuación, se definen tres módulos especiales (que se encuentran en la **ilustración 64**), los cuales sirven para entrenar un modelo “libremente”. Esto significa que, mediante *scripts* escritos en R, se pueden entrenar y alimentar modelos sin tener que utilizar modelos prediseñados.

- El módulo **T**, mencionado anteriormente, es el encargado de entrenar un modelo. Requiere de un conjunto de datos y un módulo de creación de modelos (**M**) para proceder al entrenamiento. Además, en este se define sobre que parámetro se debe predecir. En este caso, el parámetro “*Output*”.
- El módulo **S** es el encargado de hacer las predicciones y ofrecer al modelo un entrenamiento progresivo. Requiere de un conjunto de datos de prueba y un modelo entrenado para proceder a la predicción.
- El módulo **M** se encarga de la propia creación del modelo, asociando los *scripts* necesarios para el módulo T y el módulo S.

El módulo **R2 (ilustración 64)** se encarga de obtener un conjunto de datos de entrenamiento del conjunto de datos generado en la fase anterior. El conjunto de datos para entrenamiento es el primer 90% de este.

El módulo **R3 (ilustración 64)** se encarga de obtener un conjunto de datos de pruebas del conjunto de datos generado en la fase anterior. El conjunto de datos para pruebas es el último 10% de este.

Las implementaciones de los *scripts* del módulo M son importantes. El primero, para el módulo de entrenamiento, se define como se entrena el modelo, en dos líneas:

```
# Input: dataset
# Output: model
library(randomForest)
# Entrenar modelo
model <- randomForest(Output ~ W + WdSpd1 + WdSpd2 + Spd + Dir + H2OInTmp + StaTmpA +
StaTmpB + StaTmpC + BearDe + BrgHSSGGStmp + BrgHSSGRStmp + BrgIMSGGStmp + BrgIMSRStmp +
PtAngSpB11, dataset, ntree=30, nodesize=20)
```

Es decir, entrena el modelo utilizando el algoritmo de *Random Forest*, tomando como referencia de predicción el parámetro “*Output*”, y utilizando como parámetros significativos los estudiados por el equipo de IK4-Ikerlan. Además se especifica que el número de árboles sea de 30 y una profundidad de 20, debido a cuestiones de eficiencia computacional, también estudiadas por dicho equipo.

Mientras que el módulo de predicción y aprendizaje progresivo contiene el siguiente código:

```

# Input: model, dataset
# Output: scores
library(randomForest)
# Realizar la predicción
scores <- data.frame(predict(model,dataset,type="class"))
# Nombrar a la columna generada
names(scores) <- c("Prediction")

# Comando necesario para corregir el bug de que "el dataset se queda bloqueado"
# al utilizarse el modelo cuando está desplegado como un servicio web.
unlockBinding("dataset",environment())

# Comando necesario para corregir el bug de "todas las conexiones en uso"
# al utilizarse el modelo cuando está desplegado como un servicio web.
closeAllConnections()

```

Nótese que, además, hay que añadir cierto código para evitar que se ejecuten algunos errores. Estos errores no tendrían que ocurrir, según los desarrolladores de estas tecnologías (consultar [apartado 7.2.3.](#)).

El módulo **P1 (ilustración 64)** se encarga de excluir el parámetro *Output* para realizar predicciones de prueba sobre el modelo predictivo entrenado.

El módulo **P2 (ilustración 64)** se encarga de obtener únicamente la predicción junto con su *timestamp*.

El módulo **W (ilustración 64)** se utiliza para almacenar eventualmente un conjunto de predicciones para comprobar su correcto funcionamiento. No es obligatorio para el caso de uso.

Una vez el experimento de entrenamiento se ejecuta correctamente, se da la posibilidad de transformarlo en un experimento predictivo, con el objetivo de obtener un modelo entrenado.

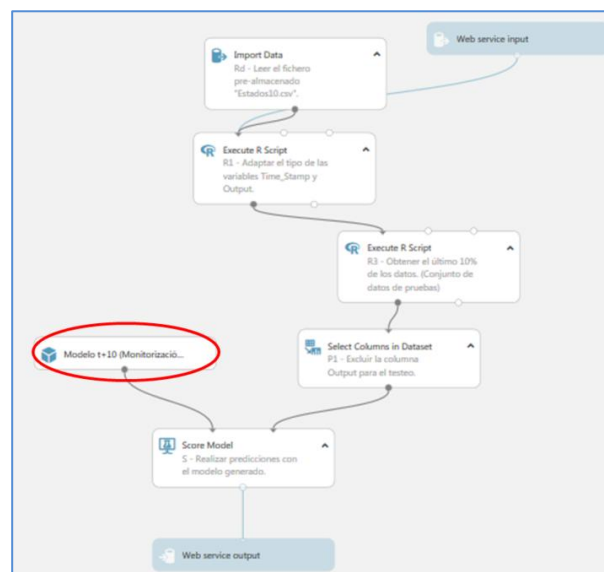
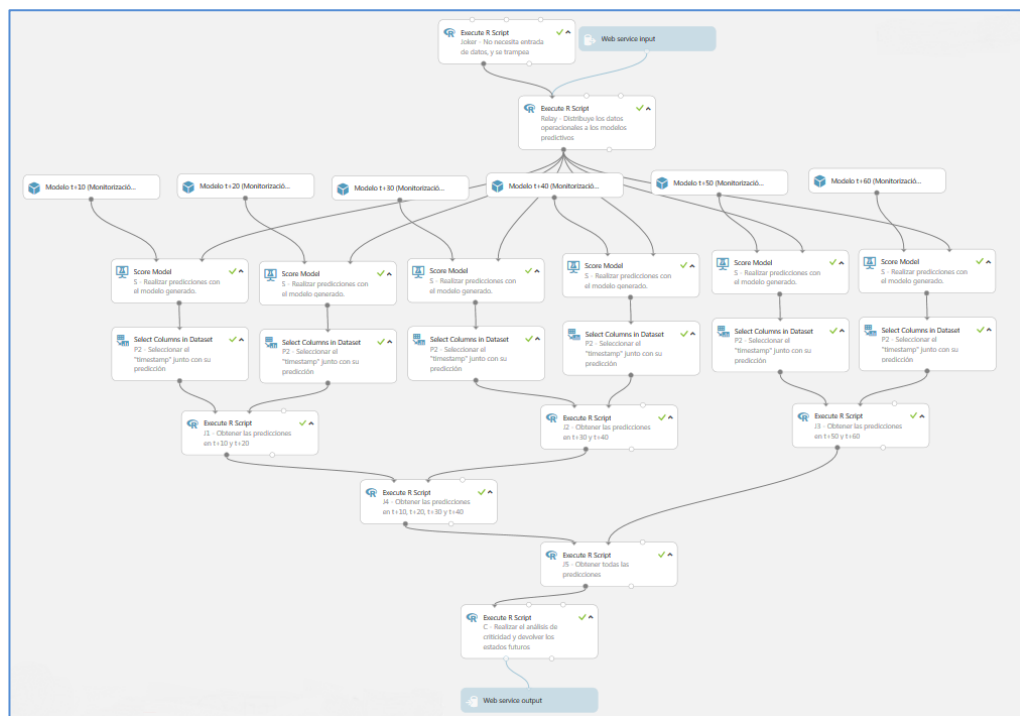


Ilustración 65. Generación del modelo t+10 entrenado

En este punto, el modelo entrenado en t+10 está disponible para su uso como un recurso más del portal, a través de un menú del portal (**etiqueta 4** de la **ilustración 27**). Es necesario realizar esto mismo con los modelos restantes (t+20 a t+60).

Una vez los modelos de t+10 a t+60 se hayan generado y entrenado, se procede a realizar la última fase del sistema predictivo.

### 7.1.1.4. Fase 3: Experimento de monitorización



**Ilustración 66. Experimento para monitorización de un aerogenerador**

Este experimento es **especial** y sirve para realizar la monitorización del aerogenerador. Se dice que es “especial”, ya que en la fase de experimento de entrenamiento y en la de experimento predictivo, este experimento es idéntico. Es decir, el experimento se crea desde un principio con el objetivo de que se despliegue como servicio web y sin entrenar modelos, ya que los modelos a utilizar ya están entrenados por los pasos realizados en la fase anterior.

En la **ilustración 66**, hay seis modelos entrenados con sus correspondientes módulos de predicción (**S**) y módulos **P2** para obtener las predicciones en formato “*timestamp*-predicción”, de igual forma que se describe en la fase anterior. Además, también está el módulo que se encarga de hacer el análisis de la criticidad (**C**, descrito en el [apartado 3.2.](#)). Estos son los módulos fundamentales del experimento.

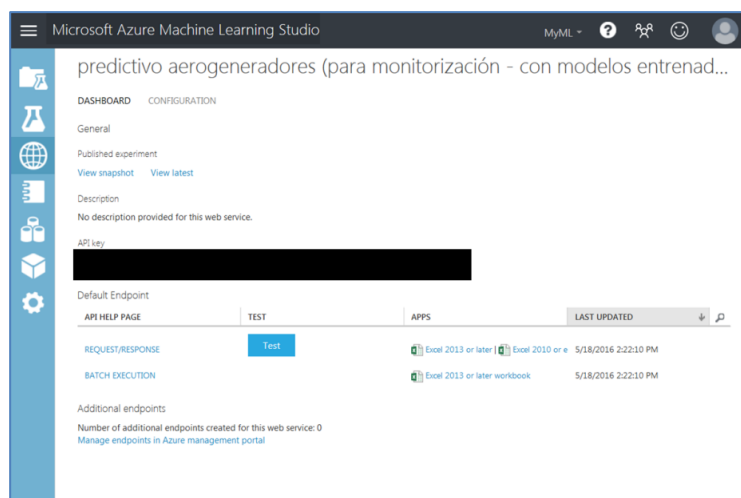
Los que se describen a continuación, son módulos encargados de adaptar la información.

El módulo denominado **Joker** (**ilustración 66**) es un módulo con cierta importancia. Se encarga de definir los datos para la entrada del servicio web (*web service input*), sin aportar datos que puedan afectar al entrenamiento progresivo de los modelos predictivos. Es decir, hace ver al sistema que hay datos cuando no los hay. Este módulo es necesario para evitar que, a la hora de ejecutar el experimento, de un error indicando que no se han podido obtener datos. Esto se debe a que, a pesar de que el experimento tiene una entrada del servicio web, en una ejecución de prueba el servicio web no aporta datos al módulo que requiere datos (el módulo *Relay*). Por lo tanto se incluye este módulo, para generar una estructura de datos vacía, que al fin y al cabo, son datos. Ya que **vacío no es lo mismo que nulo**. Como curiosidad extra, este módulo podría definirse como un programa o script, que genera un comportamiento diferente en el sistema, sin causar a este ningún daño. Definición del [virus joke](#), del cual se inspira su nombre.

El módulo denominado **Relay** (**ilustración 66**), mencionado anteriormente, es el encargado de distribuir la información recibida por el servicio web a los modelos predictivos, donde los cuales harán sus correspondientes predicciones.

Los módulos **J1** a **J5** (**ilustración 66**), se encargan de juntar, de forma ordenada, las predicciones realizadas por los modelos predictivos. Se utilizan tantos módulos, por culpa de las limitaciones que tienen los módulos de ejecución de R; máximo dos entradas de datos. De esta forma, el *dataset* resultante del módulo **J5** dispone de las predicciones de los seis modelos predictivos. Estas predicciones sirven para realizar el análisis de criticidad (con el módulo **C**, mencionado anteriormente), para determinar en qué nivel de criticidad se encuentra el aerogenerador. Esto se describe en el [apartado 3.2.](#). El resultado de este análisis se devuelve a la salida del servicio web (*web service output*).

Una vez el experimento de entrenamiento y el experimento predictivo se ejecuten correctamente, es posible hacer el despliegue del experimento como servicio web. Para ello, se pulsa el botón “DEPLOY WEB SERVICE” situado en el apartado de control del experimento (**etiqueta 11** de la **ilustración 27**). Una vez el servicio web se termina de desplegar, se abre automáticamente el manual autogenerado del mismo, donde se pueden ver ejemplos de uso y otras características de este.



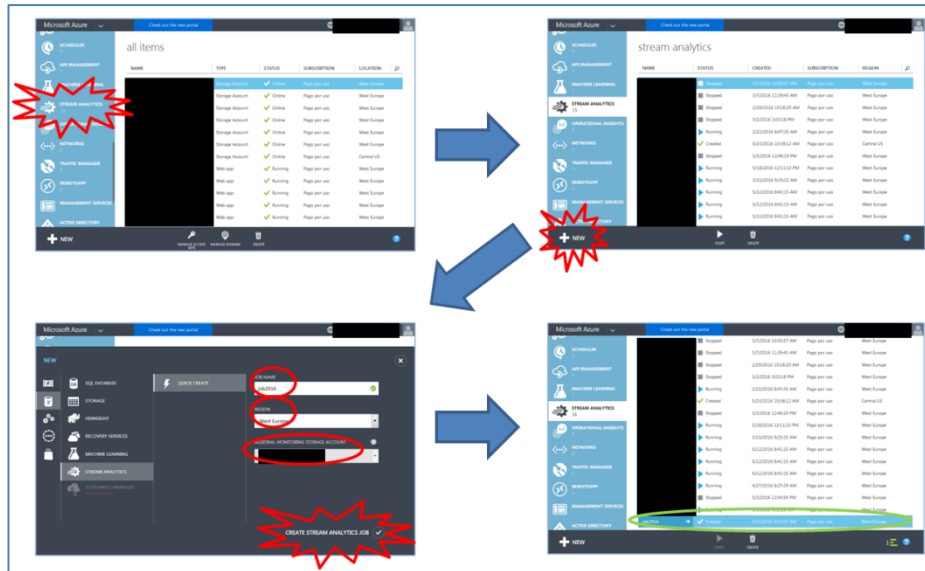
**Ilustración 67. Manual autogenerado del servicio web.**

A modo explicativo, y sin entrar en detalles sobre el desarrollo del código, esta sería la implementación realizado en esta parte.

### 7.1.3. Azure Stream Analytics (análisis y gestión de la información)

Esta parte se define como el “puente” de todo el sistema, ya que conecta todos los servicios de la nube entre ellos.

Para ello, se crea un *job*, utilizando el [portal de Azure](#).



**Ilustración 68. Proceso para crear un job**

Una vez el *job* esta creado, debe ponerse en funcionamiento. Para ello, hay que definir, por lo menos, una fuente de datos (*input*), un receptor de resultados (*output*) y una consulta de transformación.

La **fuente de datos** que se utiliza es el *Event Hub* creado anteriormente (**ilustración 69**).

Para ello se especifican los datos sobre el *Event Hub* que se trata como fuente de datos, entre otras cosas, el nombre del *Event Hub*, grupo de consumidores, política SAS, etc. Una vez hecho esto, la fuente de datos queda definida.

El **receptor de resultados** que se utiliza es el *dashboard* de *Power BI* (**ilustración 70**).

Para ello se especifican los datos sobre donde se almacenarán los datos enviados hacia la cuenta de *Power BI*, la cual se va a tratar como receptora de resultados. Además de autorizar el uso, se proporciona el nombre del grupo de trabajo, el *dataset* y la tabla donde se almacenarán los datos. Una vez hecho esto, el receptor de resultados queda definido.

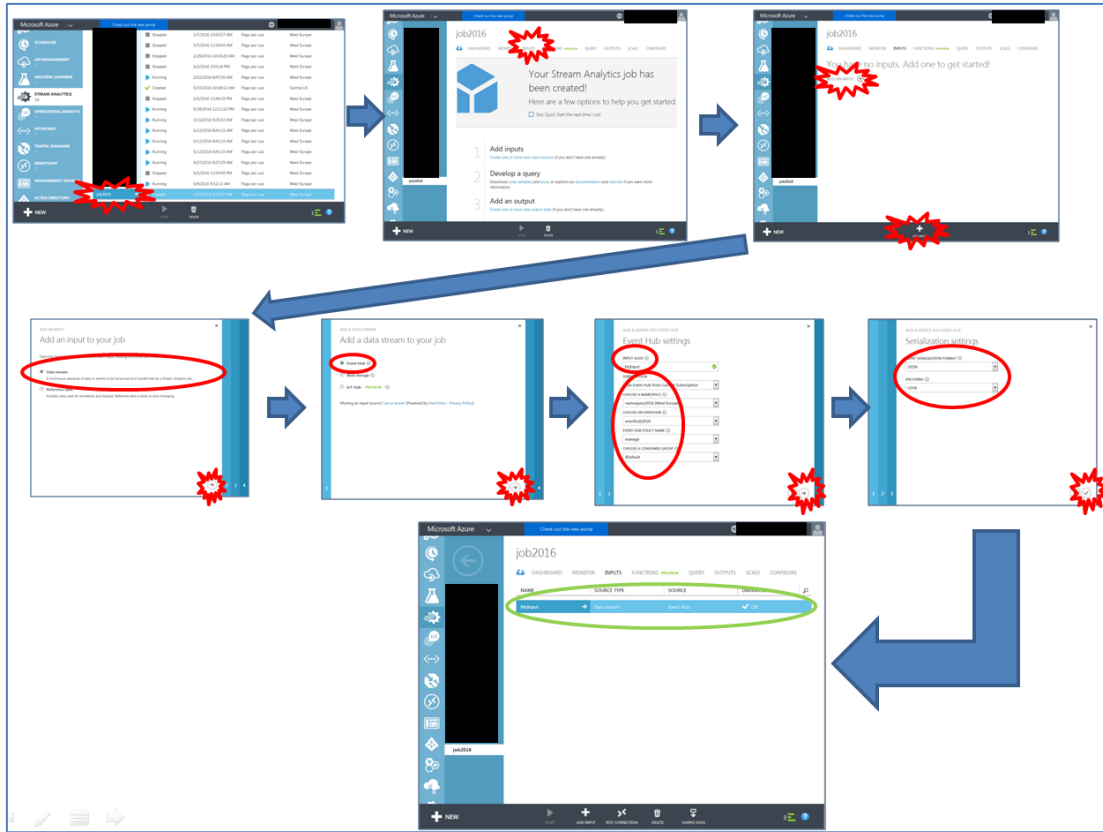


Ilustración 69. Proceso para añadir una fuente de datos a un *job*

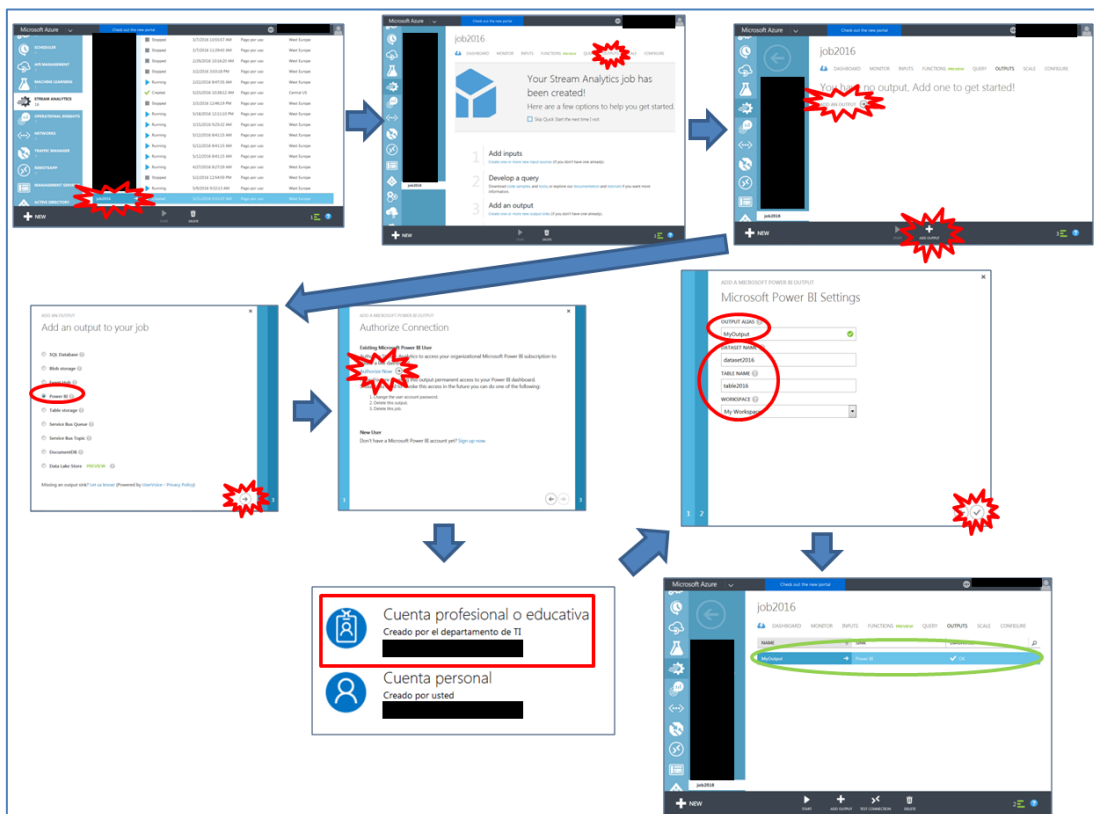


Ilustración 70. Proceso para añadir un receptor de resultados a un *job*



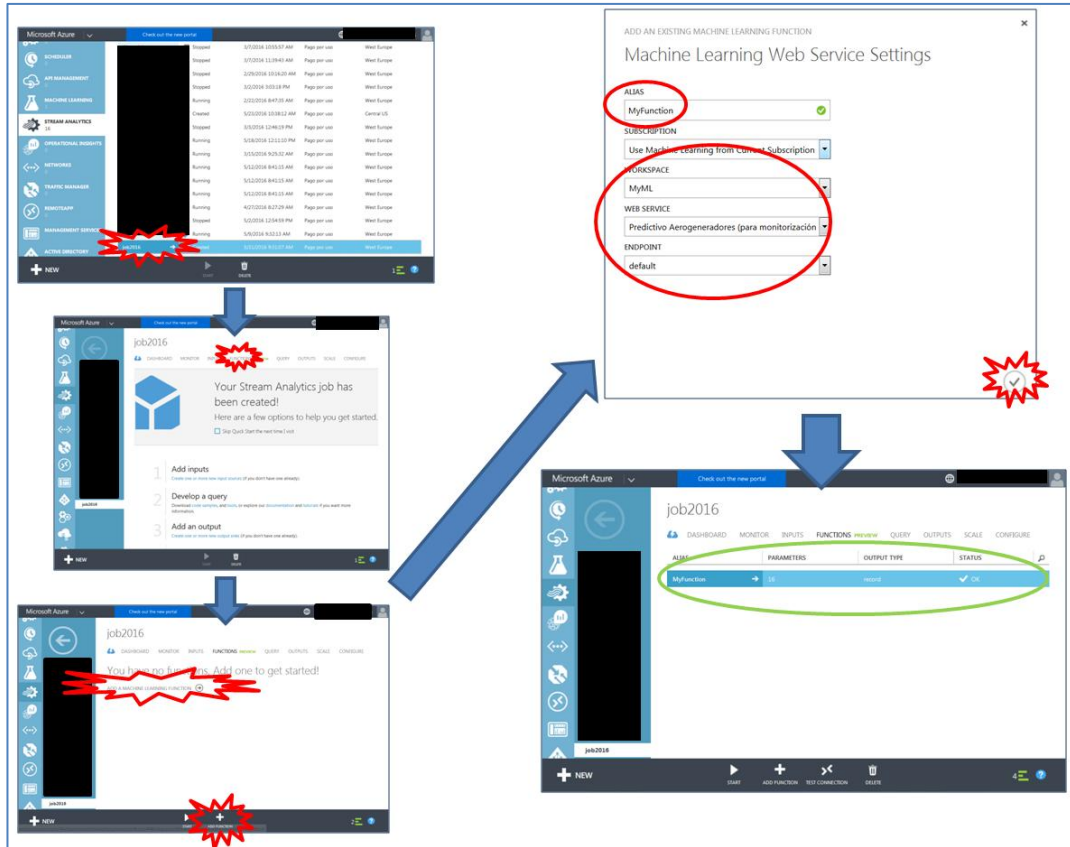


Ilustración 71. Proceso para añadir una función a un job

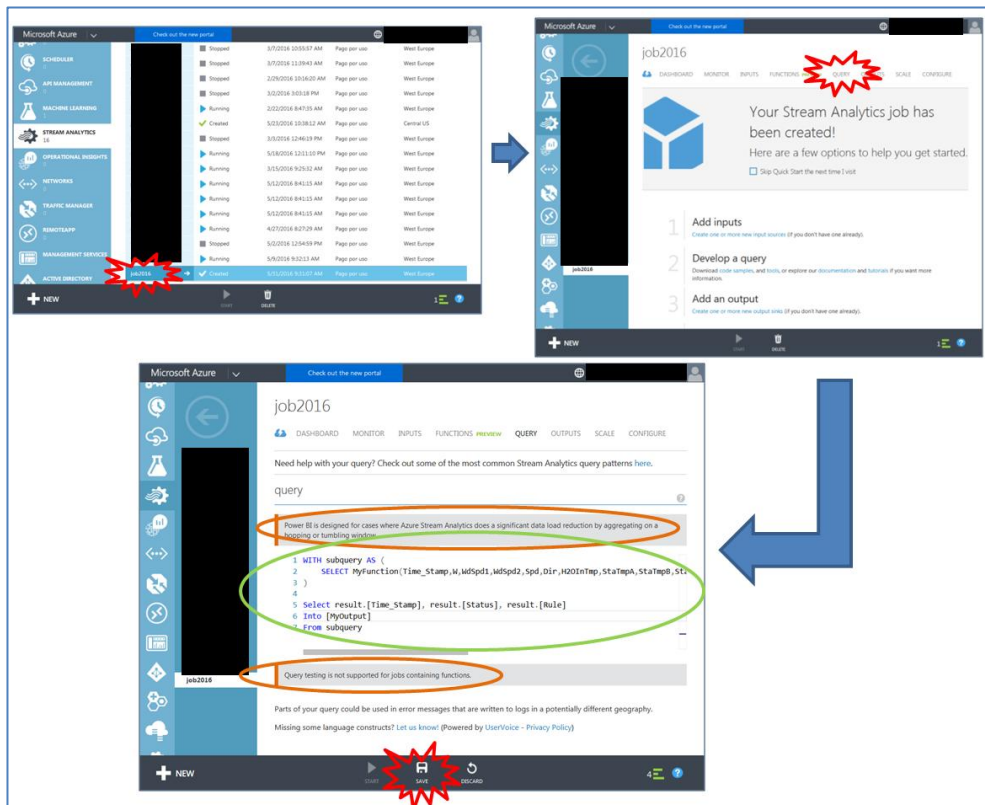


Ilustración 72. Proceso para aplicar una consulta de transformación a un job

Por último, queda definir la consulta de transformación. Para ello, primero hay que definir una **función**, la cual es la encargada de comunicarse con el servicio web desplegado anteriormente (**ilustración 71**).

Una vez creada la función, se puede definir la consulta de transformación (**ilustración 72**).

La consulta utilizada en este ejemplo (la cual sustituye a la generada por defecto) es la siguiente (nótese que los parámetros de la función son los parámetros significativos para las predicciones comentados anteriormente):

```
WITH subquery AS (
  SELECT
    MyFunction(Time_Stamp,W,WdSpd1,WdSpd2,Spd,Dir,H2OInTmp,StaTmpA,StaTmpB,StaTmpC,BearD
    e,BrgHSSGGStmp,BrgHSSGRStmp,BrgIMSGGStmp,BrgIMSRStmp,PtAngSpB11) as result from
    [MyInput]
)

Select result.[Time_Stamp], result.[Status], result.[Rule]
Into [MyOutput]
From subquery
```

Una vez definida la consulta de transformación, el *job* está listo para ser ejecutado. Al pulsar “START”, comienza su ejecución.

De esta forma, el *job* remite los datos obtenidos de *Event Hub* hacia el servicio web, cuyos resultados se envían a una tabla de *Power BI*. Todo esto acontece en tiempo real.

#### 7.1.4. Aplicación emisora de datos (simulador de un aerogenerador)

Dado que esta parte consiste en redacción de código de una aplicación de consola, a continuación se hace una síntesis sobre qué y cómo se utiliza para que el cliente realizado se comunique con el *Event Hub* previamente creado. Se introduce código únicamente para entender algunos casos importantes.

Para desarrollar la aplicación cliente se utiliza la API para gestión de espacios de nombres ([apartado 5.1.6.3. \(API de Azure Service Bus\)](#)). Para ello, en muchos casos de prueba realizados y vistos en ejemplos en la fase de investigación práctica, se ha visto que la cadena de conexión se “*hardcodea*” en la aplicación, tanto para las instancias de los servicios (por ejemplo, *Event Hub*) como para la gestión del espacio de nombres. A modo de ejemplo:

```
EventHubClient.CreateFromConnectionString(“SENDER_CONNECTION_STRING_GOES_HERE”);

o

NamespaceManager.CreateFromConnectionString(“ADMIN_CONNECTION_STRING_GOES_HERE”);
```

Para evitar estas configuraciones, se ha optado por una configuración más **segura**. Consiste en configurar un certificado autofirmado para el espacio de nombres que gestiona el *Event Hub* destino, con el objetivo de evitar introducir cualquier



credencial en el código. El proceso para crear este tipo de certificados y su instalación se explica en el [apartado 5.1.6.2. \(OpenSSL\)](#). De esta forma, únicamente los dispositivos que tengan instalado este certificado pueden acceder al espacio de nombres y, por lo tanto, a los recursos que gestiona este. Además, cifra las comunicaciones entre la aplicación emisora y la plataforma.

Para poder hacer uso del certificado en la aplicación, primero hay que definir unos parámetros a nivel de aplicación; en “*App.config*” (**ilustración 73**).

```
<appSettings>
  <add key="subscriptionID" value=" " />
  <add key="managementCertSN" value="namespace2016.servicebus.windows.net" />
  <add key="serviceNamespace" value="namespace2016" />
</appSettings>
```

**Ilustración 73.** Parámetros definidos en *App.config*

La definición de estos parámetros:

- subscriptionID: Identificador único de la suscripción. Se genera al crear la cuenta de *Azure*, y es el que identifica una unidad de pago (unidad donde se hace el cargo de los gastos mensuales de los servicios de *Azure*, no confundir con la cuenta de *Azure*, ya que cada cuenta puede tener más de una suscripción).
- managementCertSN: “*Common Name*” asociado al certificado en el momento de su creación (consultar el [apartado 5.1.6.2. \(OpenSSL\)](#)).
- serviceNamespace: Espacio de nombres asociado al certificado.

Gracias a estos parámetros, se puede gestionar el espacio de nombres directamente, sin necesidad de aportar credenciales.

Estos parámetros se utilizan en un método llamado “*ReadSASRuleOnNamespaceRoot*” (**ilustración 74**).

```

public static List<SharedAccessAuthorizationRule> ReadSASRulesOnNamespaceRoot(string subscriptionID, string managementCertSN, string serviceNamespace)
{
    // The endpoint for retrieving the SAS rules on the namespace is:
    // https://management.core.windows.net/{subscriptionId}/services/ServiceBus/namespaces/{namespace}/AuthorizationRules/
    string baseAddress = @"https://" + managementEndpoint + subscriptionID + @"/services/ServiceBus/namespaces/" +
        serviceNamespace + @"/AuthorizationRules/";

    // Operations on the Service Bus namespace root require certificate authentication.

    WebRequestHandler handler = new WebRequestHandler
    {
        ClientCertificateOptions = ClientCertificateOption.Manual
    };
    try
    {
        handler.ClientCertificates.Add(GetCertificate(managementCertSN));
    }
    catch (Exception e)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("No certificate associated to this service was found in client machine.");
        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("Check if the certificate associated to this service is installed in your local store.");
        Console.WriteLine("Check if the 'managementCertSN' setting is typed correctly.");
        Console.ResetColor();
        Console.WriteLine("Press 'Enter' to terminate the client.");
        Console.ReadLine();
        Environment.Exit(1);
    }
}

HttpClient httpClient = new HttpClient(handler)
{
    BaseAddress = new Uri(baseAddress)
};
httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
httpClient.DefaultRequestHeaders.Add("x-ms-version", "2012-03-01");

// Doing a GET on the base address above returns all the auth rules configured.
var getResult = httpClient.GetAsync("").Result;
List<SharedAccessAuthorizationRule> rules = new List<SharedAccessAuthorizationRule>();
try {
    rules = getResult.Content.ReadAsAsync<List<SharedAccessAuthorizationRule>>().Result;
}
catch (Exception e) {
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine("No authorization rules found.");
    Console.ForegroundColor = ConsoleColor.Cyan;
    Console.WriteLine("Check in Azure if the certificate associated to this service is installed.");
    Console.WriteLine("Check if the 'subscriptionID' setting is typed correctly.");
    Console.WriteLine("Check if the 'serviceNamespace' setting is typed correctly.");
    Console.ResetColor();
    Console.WriteLine("Press 'Enter' to terminate client.");
    Console.ReadLine();
    Environment.Exit(1);
}
return rules;
}

```

**Ilustración 74. Implementación del método *ReadSASRulesOnNamespaceRoot* (control de acceso mediante certificados)**

Este método es el encargado de verificar si en el almacén de certificados local se encuentra un certificado válido para algún espacio de nombres. En el caso de que así sea, obtiene las reglas de autorización del espacio de nombres. En caso contrario avisa de que no existe un certificado válido o que algún parámetro se ha descrito incorrectamente.

Una vez se obtienen las reglas del espacio de nombres, se selecciona una regla gestora, para poder manejar los recursos del espacio de nombres (en este caso sólo hay una regla y es gestora; la regla por defecto denominada *RootManageSharedAccessKey*. Se obtiene con el siguiente método:

```

private static void obtainRootManageRule(string subscriptionID, string managementCertSN, string serviceNamespace) {
    List<SharedAccessAuthorizationRule> nsSasRules = NamespaceAuthRulesCRUD.ReadSASRulesOnNamespaceRoot(subscriptionID, managementCertSN, serviceNamespace);
    nsSasRules.ForEach(delegate (SharedAccessAuthorizationRule rule)
    {
        if (rule.KeyName == "RootManageSharedAccessKey")
        {
            rootManageRule = rule;
        }
    });
}

```

**Ilustración 75. Implementación del método *obtainRootManageRule* (obtener la regla gestora por defecto)**

La clave y el nombre de esta regla gestora se utilizan en el método *SendMessageToEH*, para poder realizar el envío de datos al *Event Hub* objetivo.

```
public static void SendMessageToEH(string serviceNamespace, string ehPath, string keyName, string key, string message)
{
    // Create the event hub service URI
    Uri runtimeUri = ServiceBusEnvironment.CreateServiceUri("sb", serviceNamespace, string.Empty);

    // Create an instance of NamespaceManager for the operation.
    Uri managementUri = ServiceBusEnvironment.CreateServiceUri("https", serviceNamespace, string.Empty);

    // Ask for tokens related to namespace.
    TokenProvider sasTP = TokenProvider.CreateSharedAccessSignatureTokenProvider(keyName, key);

    // Obtain session to manage the namespace.
    NamespaceManager nsm = new NamespaceManager(managementUri, sasTP);

    // Get the event hub description from the namespace.
    EventHubDescription ehd = nsm.GetEventHub(ehPath);

    // Obtain the sending rule, providing the rule name
    SharedAccessAuthorizationRule myRule = obtainRuleByName(ehd, "send");

    // Create connection with event hub
    var sendClient = EventHubClient.CreateFromConnectionString("Endpoint=" + runtimeUri + ";SharedAccessKeyName=" + myRule.KeyName + ";SharedAccessKey=" + myRule.PrimaryKey, ehPath);

    // Sending message to event hub.
    EventData sendMessage = CreateMessage(message);
    sendClient.Send(sendMessage);
}
```

Ilustración 76. Implementación de *SendMessageToEH* (envío de datos)

En este último método se define, entonces, que regla se utiliza para enviar los datos al *Event Hub* descrito en el programa principal (regla “*send*”, definida anteriormente).

Este método se utiliza en el programa principal, el cual es el que simula el aerogenerador.

Nótese que el método de envío que se utiliza en el ejemplo es el método por defecto, ya que como se va a enviar información de un aerogenerador, no es necesario organizar la información por particiones o publicadores.

De todos modos, si se quisiera enviar información en función de claves de partición o publicadores, se puede hacer modificando las propiedades de un *EventData* antes de enviarlo.

```
ed.SystemProperties[EventDataSystemPropertyNames.PartitionKey] = partitionKey;
```

O

```
ed.SystemProperties[EventDataSystemPropertyNames.Publisher] = publisherId;
```

Una vez el cliente de ha desarrollado por completo, se compila y ejecuta. Si todos los pasos se han desarrollado correctamente, una vez el cliente comience a enviar datos, al abrir el portal de [Power BI](#) debe aparecer un nuevo *dataset* denominado “*dataset2016*”, como se ha descrito como salida del *job* anteriormente.

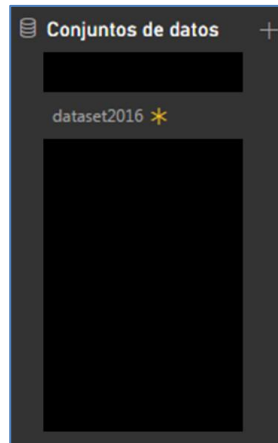


Ilustración 77. Dataset generado automáticamente en el portal de Power BI

Esto indica que la información esta fluyendo como se esperaba. Sólo queda desarrollar la parte visual del sistema.

### 7.1.5. Power BI (visualizar los datos procesados)

Con el objetivo de que no se extienda demasiado la explicación del desarrollo sobre la parte visual, se describe a continuación como se desarrolla una visualización con filtros, ya que el resto de visualizaciones siguen el mismo proceso de desarrollo.

Una vez exista el *dataset* (dentro del portal de *Power BI*), se puede proceder a crear visualizaciones a través de un informe, utilizando los datos de la tabla generada (ilustración 78).

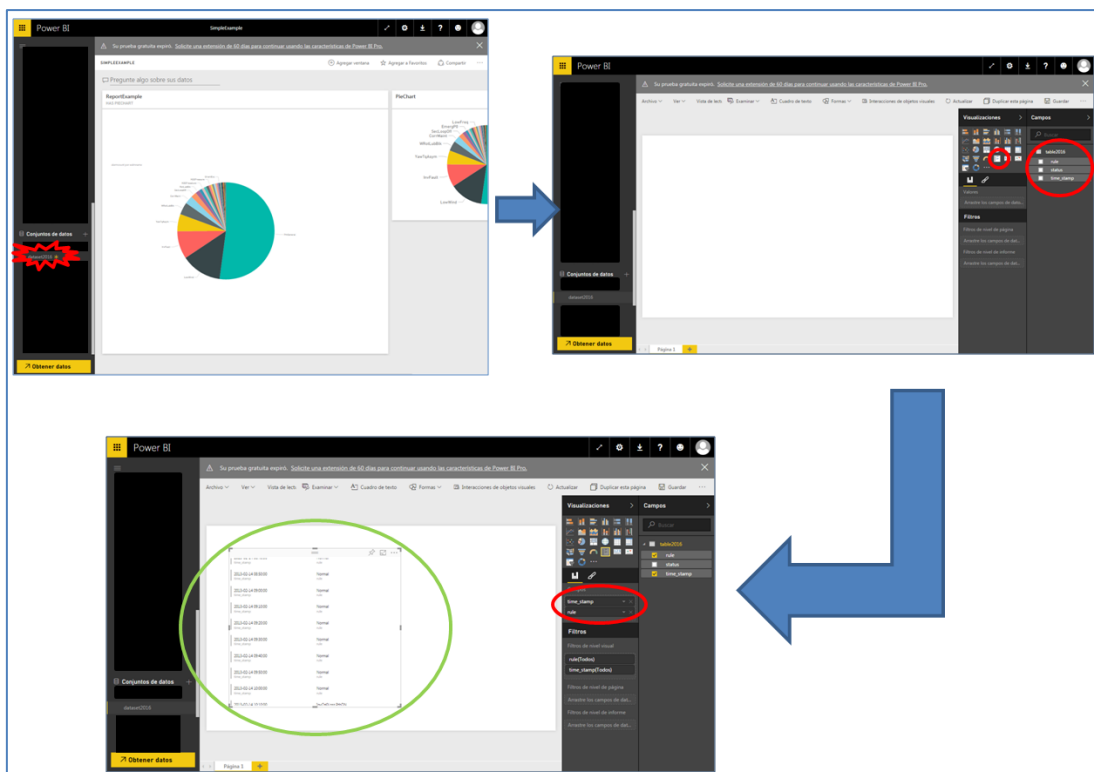


Ilustración 78. Crear una visualización a partir de un dataset (1/3)

En este caso, se crea una lista donde se muestra el *timestamp* junto a la regla que ha ocasionado el estado (campos *time\_stamp* y *rule*).

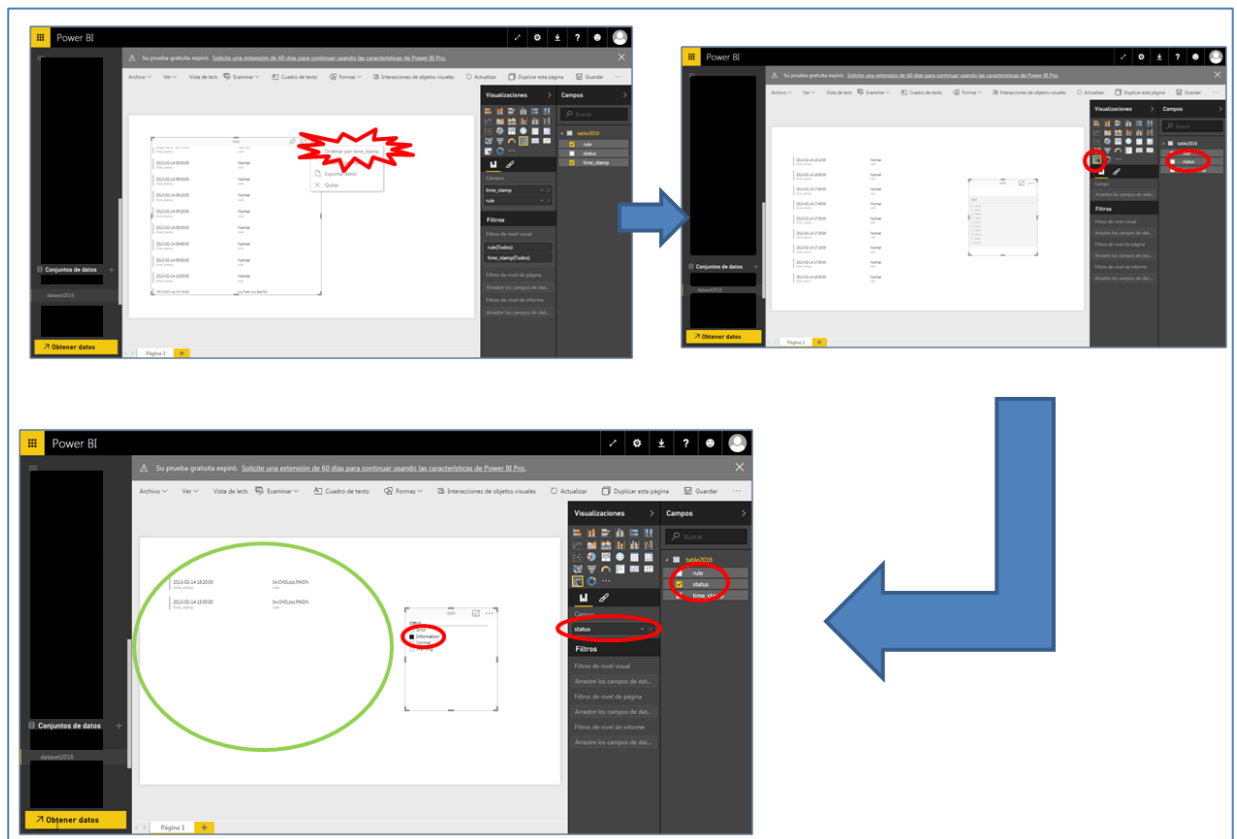
Una vez creada la base de la visualización, se aplican los filtros correspondientes para visualizar la información requerida (**ilustración 79**).

En este caso, se ordena la información por *timestamp* y se filtra para que se muestren las predicciones con un nivel de criticidad “*Information*”.

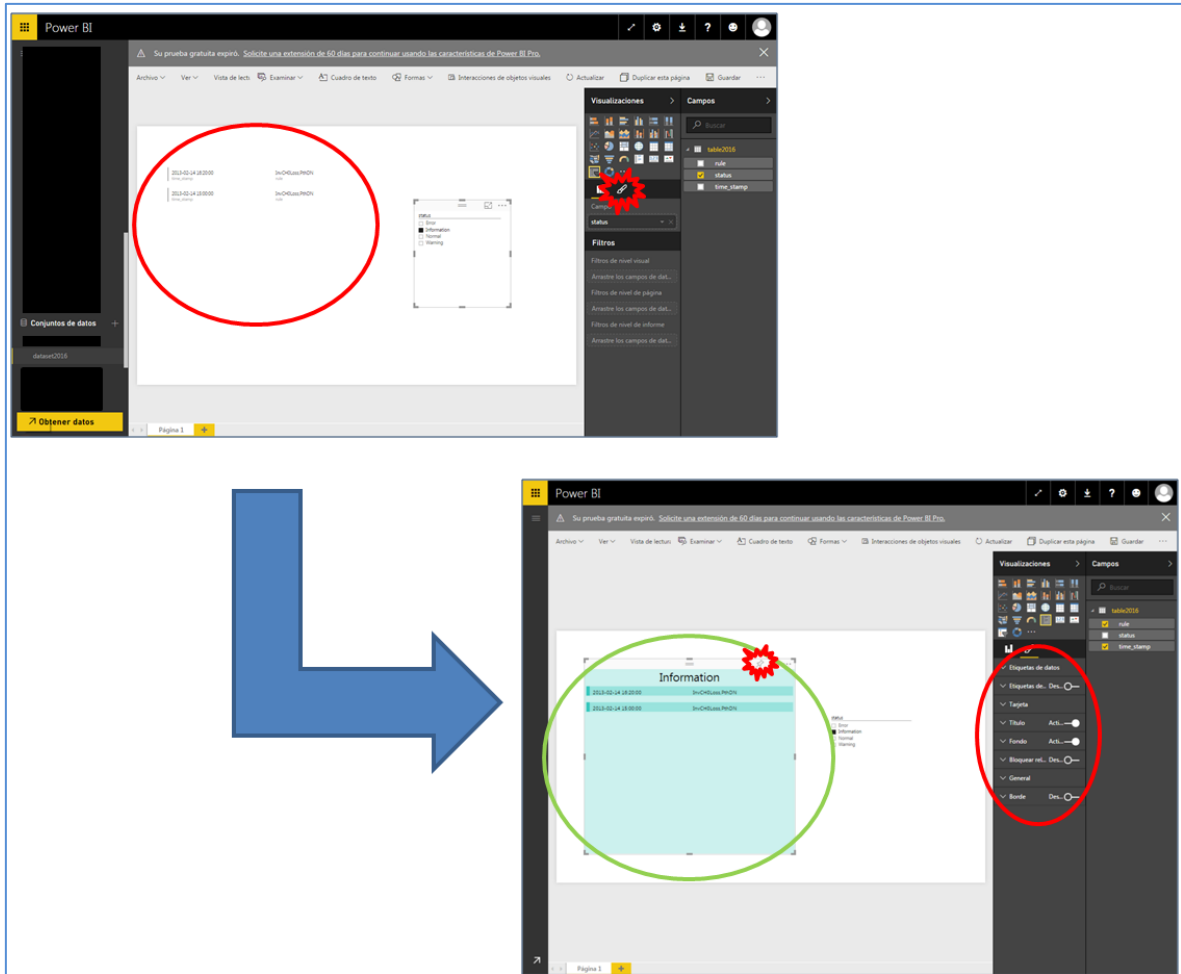
Una vez filtrada la información de la visualización, se procede a personalizar el aspecto del mismo y anclarlo al panel de visualización, para que sea posible ver la información actualizándose en tiempo real.

En este caso, se alteran varias propiedades de la visualización para que tenga un aspecto como el que se muestra en la **ilustración 80**. La visualización se ancla al panel para que se vea la información actualizándose en tiempo real. Nótese que si no existe ningún panel, ofrece la posibilidad de crear uno. En el caso de que no se haya guardado el informe, *Power BI* ofrece guardarlo primero.

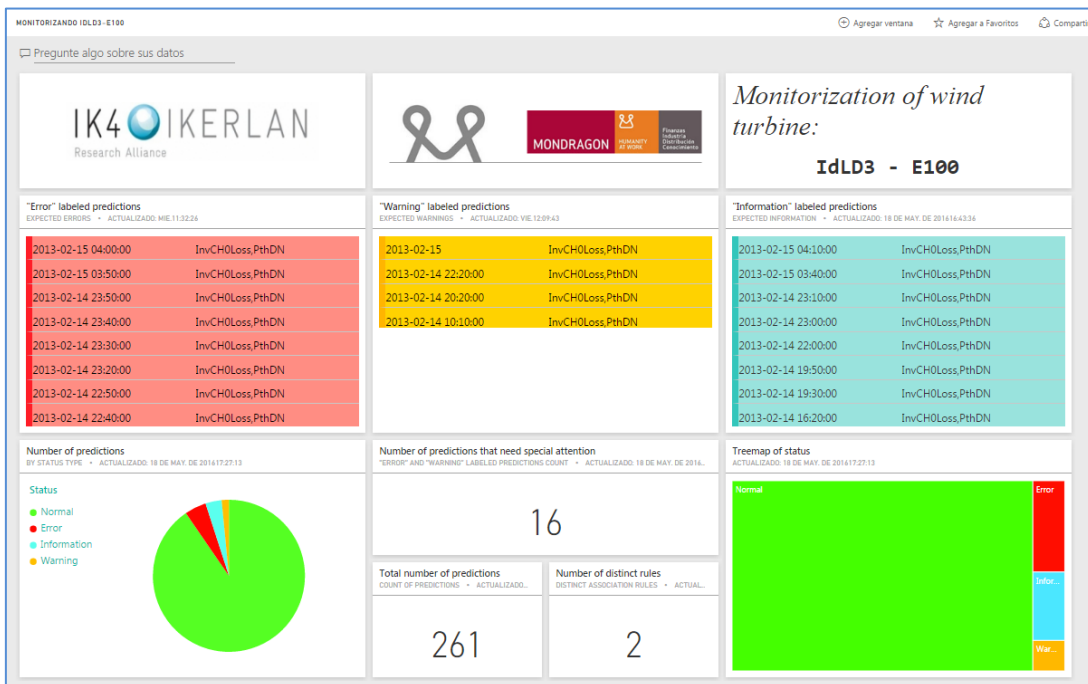
Una vez realizadas varias visualizaciones, y añadidos otros aspectos visuales, el *dashboard* de monitorización está listo para ser utilizado. El *dashboard* completo se muestra en la **ilustración 81**.



**Ilustración 79. Aplicar filtros a la visualización (2/3)**



**Ilustración 80. Modificar el aspecto de la visualización y anclarla a un panel (3/3)**



**Ilustración 81. Dashboard de monitorización final**

## 7.2. Problemas encontrados

---

A continuación, se define una recopilación de problemas encontrados (con soluciones, en el caso de que las tengan) y descripción de algunas limitaciones. Todos los problemas y limitaciones descritos están comprobados en **mayo de 2016**.

### 7.2.1. Azure Event Hubs (ingestión de la información)

- A la hora de definir las políticas de acceso, hay que tener en cuenta que, si el *Event Hub* se va a utilizar como fuente de datos de un *job*, la política de acceso que debe utilizar este tiene que tener derechos de **gestión**. Ya que, como se comenta en el [apartado 5.1.3.1. \(Fuentes de datos\)](#), se encarga de gestionar los *checkpoints* y manipular los metadatos de los eventos (<https://blogs.msdn.microsoft.com/kaevans/2015/02/26/using-stream-analytics-with-event-hubs/>).
- *Azure Event Hubs* no tiene la posibilidad de filtrar mensajes, sólo actúa de intermediario entre los publicadores y consumidores. Esto hace que los datos que se envíen tengan que estar bien definidos, o apoyarse en otra tecnología o servicio para filtrar los datos recibidos (<https://social.msdn.microsoft.com/Forums/azure/en-US/474b7eaf-a607-4e25-88b9-f7f1b034d36e/can-messages-from-azure-event-hub-be-filtered-to-this-level?forum=servbus>).

### 7.2.2. Azure Stream Analytics (análisis y gestión de la información)

- El lenguaje tipo SQL para definir las consultas de transformación es muy limitado, y hay poca información disponible sobre este (<https://azure.microsoft.com/en-us/documentation/articles/stream-analytics-stream-analytics-query-patterns/> y <https://msdn.microsoft.com/en-us/library/mt582049.aspx>). Además es muy costoso comprobar si la consulta de transformación es correcta cuando se utilizan funciones, ya que como se muestra en la **ilustración 72**, “la comprobación de consultas no está soportado para *jobs* que contengan funciones”. Esto hace que las pruebas sobre la consulta sean más lentas, ya que hay que iniciar el *job* y enviarle datos para comprobar su funcionamiento.
- Es capaz de leer los datos de las fuentes de datos definidas, incluso si están mal estructuradas (por ejemplo, en una estructura JSON mal definida). Sin embargo, al estar mal estructuradas, no las podrá interpretar, y por lo tanto, no devuelve nada a los receptores de resultados. Esto es un problema, ya que no informa de que “la estructura de los datos obtenida está mal definida”.
- A través de su API, se pueden definir distintos aspectos de este servicio (crear un *job*, añadirle fuentes de datos y receptores de resultados, añadir una consulta de transformación, etc), sin embargo, no permite añadir programáticamente una tabla de *Power BI* como receptor de resultados. Este



aspecto podría estar bien, ya que de este modo, la configuración que se hiciera sobre el *job* podría ser portable.

- No tiene la capacidad de acceder a las propiedades opcionales definibles por los publicadores. Una posible solución sería añadir estas como parte de los datos originales (<https://social.msdn.microsoft.com/Forums/es-ES/f40756e9-2f90-4c8a-ab3c-8665a81696ed/eventdata-properties-in-stream-analytics?forum=AzureStreamAnalytics>).

### 7.2.3. *Machine Learning* (creación y uso del sistema predictivo)

- Los módulos de ejecución de código R, ejecutan código R de una versión anterior a la desarrollada en local (se puede comprobar incluyendo en la acción de un módulo la línea de código `print(R.version.string)`). Esto devuelve que el portal soporta la versión 3.1.0, mientras que el código que se elaboró en local fue con la versión 3.2.3. Dado que el lenguaje evolucionó considerablemente entre estas dos versiones, se tuvo que adaptar el código desarrollado a la versión del portal, ya que no hay opción de definir la versión que se quiere utilizar.
- **Los experimentos** desarrollados en el portal de *Machine Learning* **tardan bastante en ejecutarse**, lo que hace que el proceso de prueba de estos sea más lento. Debido a esto, los datos que se utilizan son menos de los esperados, y por consiguiente, afecta negativamente al entrenamiento de los modelos predictivos, y consecuentemente, a sus predicciones. En este caso, de cuantos más datos dispongan los modelos predictivos, más certeras son sus predicciones.
- Los datos utilizados son datos modificados aleatoriamente. Esto se debe a que se están subiendo a una plataforma pública, y la empresa propietaria de estos datos no ha dado permiso para publicarlos. Esto **afecta negativamente** al entrenamiento de los modelos predictivos, y consecuentemente, **a sus predicciones**.
- *Machine Learning* es el servicio donde se desarrolla la parte nuclear del proyecto (el sistema predictivo) y, en cambio, es el menos escalable de todos. Explicación: como se comenta en el [apartado 3.1.](#), para cada aerogenerador deben crearse seis modelos predictivos y alimentarlos con seis conjuntos de datos distintos. Esto significa que habría que desarrollar las fases definidas en el [apartado 7.1.2. \(\*Machine Learning\*\)](#) por cada aerogenerador que se quiera monitorizar. Esto se podría solventar configurando programáticamente la creación y prueba de los experimentos; cosa que no permite la API de *Machine Learning* (<https://feedback.azure.com/forums/257792-machine-learning/suggestions/7575534-deployment-of-ml-experiments-programmatically>). Esta es la **causa principal** de que el caso de uso desarrollado consista en la monitorización de un único aerogenerador.
- El error descrito en el siguiente enlace puede ser consecuencia de un *input* del servicio web mal situado en el experimento predictivo (<http://stackoverflow.com/questions/28614199/r-error-in-data-frame-replacement-has-items-need>).
- No es posible ejecutar código R paralelizado en los módulos de ejecución R, ya que esta ejecución realiza una apertura de puertos por proceso iniciado (en



función del número de núcleos a utilizar), que posteriormente se sincronizan entre ellos para devolver un resultado. La máquina que ejecuta este programa no permite la apertura de puertos (un aspecto de seguridad que se ha tenido en cuenta), con lo que el módulo termina devolviendo un error (<https://social.msdn.microsoft.com/Forums/es-ES/00aa666c-8b60-481f-bfb3-4ec64fecdf9/cannot-open-the-connection-during-parallel-computing-with-r?forum=MachineLearning>).

- A la hora de probar un experimento desplegado como servicio web, puede ocurrir el error descrito en el siguiente enlace. La solución la aporta uno de los desarrolladores de la herramienta como solución temporal, y consiste en introducir una línea de código R para desbloquear el *dataset* de datos de prueba en el módulo de predicción. El código es `unlockBinding("dataset",environment())` (<https://social.msdn.microsoft.com/Forums/azure/en-US/e20b8cbf-2e89-4898-b602-d2138bf8901c/cannot-change-value-of-locked-binding-for-dataset?forum=MachineLearning>).
- A la hora de utilizar entre varios clientes el experimento predictivo desplegado como servicio web, puede ocurrir el error descrito en el siguiente enlace. La solución la aporta un usuario anónimo y es considerada solución temporal, y aceptada como respuesta por uno de los desarrolladores de la herramienta. Consiste en introducir una línea de código R para cerrar las conexiones existentes con el módulo de predicción. El código es `closeAllConnections()` (<https://social.msdn.microsoft.com/Forums/sqlserver/en-US/5b0c2185-4f79-4fcc-8225-0201a0c5b5d7/calling-an-r-model-via-a-web-service-error-in-raw-connection-all-connections-are-in-use?forum=MachineLearning>).

#### 7.2.4. Power BI (visualización de los datos procesados)

- No dispone de filtros avanzados dinámicos para mostrar datos en función de ciertas condiciones. Hay un tema abierto en los foros de *Power BI*, pero sin respuesta por parte de los desarrolladores (<https://ideas.powerbi.com/forums/265200-power-bi-ideas/suggestions/12879000-dynamic-filters> y <https://ideas.powerbi.com/forums/265200-power-bi-ideas/suggestions/6518993-set-colors-and-conditional-formatting-in-visuals>). Si que se han introducido algunas características nuevas que hacen más dinámica la herramienta, pero sólo se han introducido en la versión *Desktop* (<https://ideas.powerbi.com/forums/265200-power-bi-ideas/suggestions/9422493-conditional-formatting-in-tables>). Tampoco se pueden definir rangos de tiempo dinámicos para mostrar datos, lo que hace que todas las visualizaciones adopten un comportamiento acumulativo a la hora de mostrar los datos (<https://ideas.powerbi.com/forums/265200-power-bi-ideas/suggestions/7865136-timeline-slicer-for-date-month-quarter-year>).
- La API de *Power BI* permite añadir datos a una tabla y vaciar la misma, sin embargo, no permite eliminar datos de una forma selectiva (<https://ideas.powerbi.com/forums/265200-power-bi-ideas/suggestions/9322578-option-to-delete-a-subset-of-rows-from-a-table>).

- via y <http://docs.powerbi.apiary.io/#reference/datasets/table-rows/clear-the-rows-in-a-table>).
- Las visualizaciones ancladas a un panel, tienen la capacidad de actualizarse en tiempo real. Sin embargo, se ha probado que los informes, a pesar de que estén anclados a un panel, no tienen la capacidad para actualizar las visualizaciones que contienen (<https://powerbi.microsoft.com/en-us/documentation/powerbi-refresh-data/#what-can-be-refreshed>).
  - Se puede realizar una ordenación de los datos por campo, sin embargo, se limita a ordenación por orden alfabético o numérico (<https://ideas.powerbi.com/forums/265200-power-bi-ideas/suggestions/7073162-sorting-ability-for-reports-in-powerbi-com> y <https://ideas.powerbi.com/forums/265200-power-bi-ideas/suggestions/10660959-improve-advance-filtering-for-datetime-to-support>).
  - Al estar conectado al portal de *Power BI* a través de un servidor *proxy*, es posible que las actualizaciones en tiempo real de las visualizaciones se interrumpan intermitentemente. Para evitar esto, simplemente se debe evitar acceder al portal de *Power BI* conectado a un servidor *proxy*.
  - Los datos a representar sobre un informe sólo pueden obtenerse de un *dataset*. Con lo que no se puede relacionar datos que residan en distintos *dataset* (<http://community.powerbi.com/t5/Service/Can-a-report-be-created-using-multiple-datasets/td-p/14170>).
  - Otras limitaciones con respecto a la consistencia estructural de los *dataset* (<https://ideas.powerbi.com/forums/268152-developer-apis/suggestions/7111791-alter-datasets-by-adding-removing-individual-table>).

En resumen, estos son los problemas y limitaciones más destacadas y necesarias de tener en cuenta, para predecir y poder completar con éxito el desarrollo anteriormente descrito.

### 7.3. Evaluación de los requisitos no funcionales

---

A continuación, se evalúa el grado de cumplimiento de los requisitos no funcionales descritos anteriormente.

- Rendimiento: El rendimiento es **medio**, ya que, a pesar de que el funcionamiento de las tecnologías y servicios en su conjunto es en tiempo real, de cuantos más datos disponga la parte del preprocesado, más lento es el mismo.
- Disponibilidad: La disponibilidad es **alta**, ya que, a pesar de que lo hacen posible los sistemas implantados por *Microsoft* para este tipo de soluciones, es posible replicar la solución a distintos centros de datos y poder aumentar su disponibilidad desde distintas partes del mundo.
- Portabilidad: La portabilidad es **baja**, ya que es difícil de trasladar la solución realizada a otros tipos de plataformas.
- Escalabilidad: La escalabilidad es **baja**. Individualmente, la mayoría de los servicios utilizados en la solución son escalables. Sin embargo, las tecnologías en conjunto como solución no lo hacen posible, puesto que el desarrollo y despliegue del sistema predictivo no puede ser automatizado.
- Costo: El costo es **bajo**, ya que, a pesar de que no hay que comprar ni mantener la infraestructura que utiliza la solución desarrollada, las cuotas de “pago por uso” que se ofrecen son realmente asequibles.
- Seguridad: La seguridad es **alta**, ya que a pesar de que por definición la plataforma gestiona los recursos y servicios de forma segura, se pueden cambiar y añadir configuraciones de seguridad sobre la misma.

## 8. Extensión: Raspberry Pi como emisora de datos

Dado que se ha dispuesto de algo más de tiempo de lo esperado en el desarrollo del proyecto, se ha hecho posible realizar una extensión del mismo. Esta parte del proyecto no ha sido planificada.

### 8.1. Descripción de la extensión

Consiste en adaptar el cliente emisor de datos desarrollado en la solución adoptada para que una *Raspberry Pi* sea capaz de realizar esa tarea de la misma manera. Además, se sustituye el servicio encargado de la ingesta de datos (*Azure Event Hubs*) por otro servicio más apropiado para este tipo de dispositivos (*Azure IoT Hubs*).

Algunas de las diferencias fundamentales entre *Azure Event Hubs* y *Azure IoT Hubs* se muestran en la siguiente tabla (información extraída de la [documentación oficial de Microsoft Azure](#)):

Ámbito	<i>Azure IoT Hubs</i>	<i>Azure Event Hubs</i>
Patrones de comunicación	Bidireccional (dispositivo → nube ; nube → dispositivo)	Unidireccional (dispositivo → nube)
Compatibilidad de protocolos	AMQP, AMQP sobre WebSockets, MQTT y HTTP/1. Además dispone de una puerta de enlace de protocolos personalizable para admitir otro tipo de protocolos.	AMQP, AMQP sobre WebSockets y HTTP/1
Seguridad	Identidad por dispositivo y control de acceso revocable.	Soporta revocación limitada mediante políticas de publicadores. A veces este tipo de soluciones orientadas a dispositivos requieren medidas contra la suplantación de identidad.
Escala	Optimizado para admitir millones de dispositivos conectados al mismo tiempo.	Admiten hasta 5000 conexiones AMQP.
SDK de dispositivo	Ofrece SDK de dispositivo para una gran variedad de plataformas y lenguajes	Compatible con .Net, C. Ofrece interfaces de envío HTTP y AMQP.

De este modo, el esquema tras realizar la extensión descrita quedaría algo como lo que se describe en la **ilustración 82**:

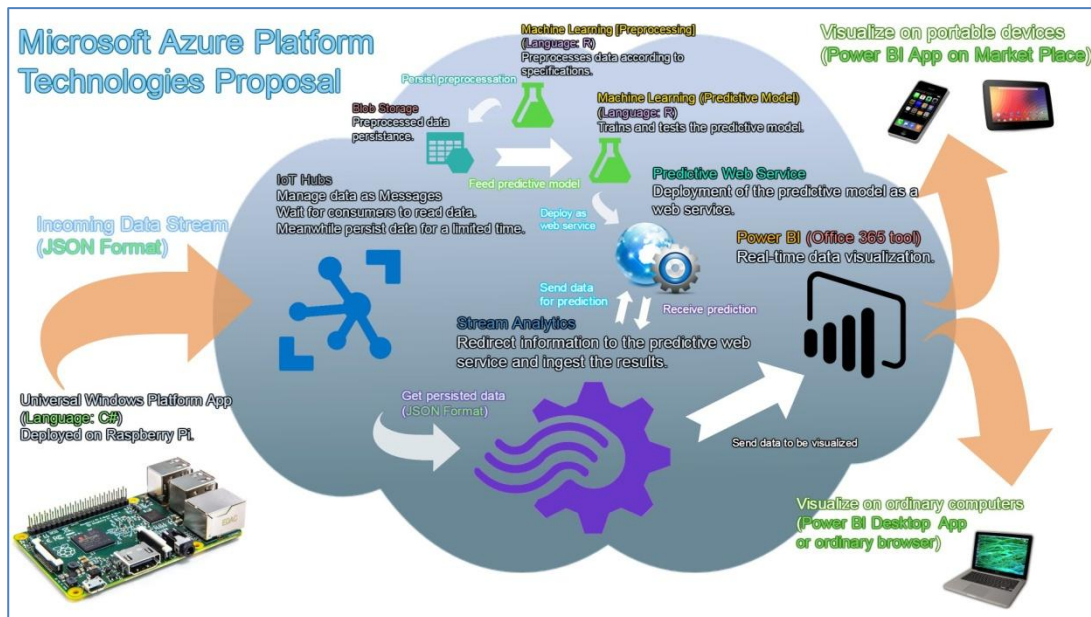


Ilustración 82. Esquema general de la solución adoptada (extendido)

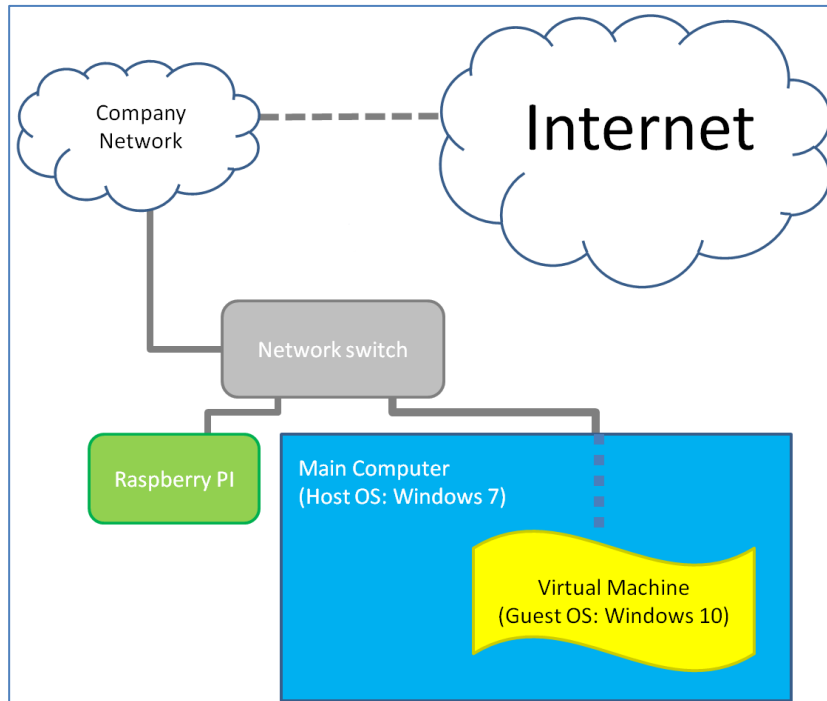
Para realizar esta extensión, han sido necesarios varios elementos (*hardware* y *software*) extra:

- **Microsoft Azure IoT Starter Kit**, el cual incluye una *Raspberry Pi 2* y una tarjeta microSD con el sistema operativo **Windows 10 IoT Core** preinstalado. Además dispone de otros componentes que también se les da un mínimo uso (entre otras cosas, cables, resistencias, módulos, diodos LED,...)
- **Máquina virtual de Windows 10.**
- **Conmutador** o *switch*.
- **Adaptador de MicroSD (USB).**

## 8.2. Implementación de la extensión

### 8.2.1. Disposición física de los componentes

Teniendo en cuenta que se dispone de una única toma RJ-45 que da acceso a Internet y a una red donde los componentes puedan comunicarse, se establece una configuración de conexiones, según se detalla en la siguiente ilustración:



**Ilustración 83. Arquitectura de red de la solución extendida**

De esta forma, la aplicación UWP (*Universal Windows Platform*, sólo compatible para dispositivos con *Windows 10*) desarrollada para adaptar el cliente de la solución adoptada, es desplegable a la *Raspberry Pi*, utilizando la máquina virtual de *Windows 10* que trabaja desde la máquina física habitual. En este caso, la máquina física se encarga de satisfacer a la máquina virtual con los recursos que ella necesite.

Dado que la *Raspberry Pi* está conectada a la red interna de la empresa, dispone de acceso a Internet para poder realizar envíos de datos (al igual que el cliente de la solución adoptada) hacia la plataforma de *Microsoft*.

Además, para comprobar que la *Raspberry Pi* está enviando datos, se ha realizado un sencillo circuito, con el cual hará saber, con el destello de un diodo LED, cuándo envía datos a la plataforma de *Microsoft*. El circuito se muestra en la siguiente ilustración:

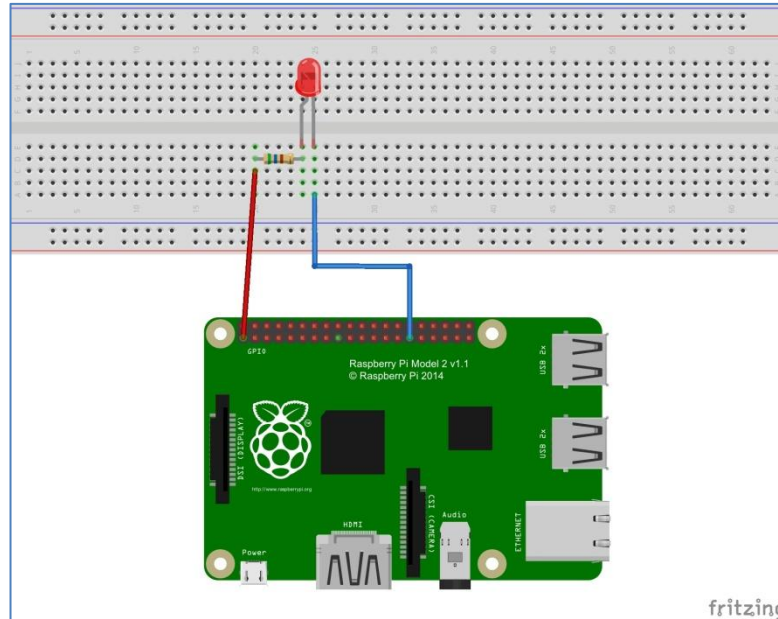


Ilustración 84. Circuito de verificación de envío desarrollado (extraído del [tutorial “Blinky”](#))

Por lo general, los datos se enviarán en un tiempo constante, igual que con el cliente desarrollado para la solución adoptada. Lo que significa que el parpadeo también lo será.

## 8.2.2. Configuración realizada en la Raspberry Pi

Dado que utiliza un sistema operativo *Windows* preinstalado en su microSD extraíble, una vez tenga conectada la alimentación y se inicie, es posible acceder a ella a través de una sesión de *PowerShell*. El nombre por defecto que utiliza el sistema es “minwinpc” y la contraseña del administrador por defecto es “p@ssw0rd”, por lo que es recomendable por lo menos cambiar estos parámetros. Para ello, una vez la placa sea visible en la red, ejecutar los siguientes comandos en una consola PowerShell (pasos extraídos del [manual de configuración con PS](#)):

- Comprobar/habilitar el servicio de conexiones remotas.

```
PS> net start WinRM
```

- Confiar en el dispositivo remoto definido (de forma temporal).

```
PS> Set-Item WSMAN:\localhost\Client\TrustedHosts -Value minwinpc
```

- Acceder por primera vez al dispositivo.

```
PS> Enter-PSSession -ComputerName minwinpc -Credential minwinpc\Administrator  
(introduciendo la contraseña por defecto)
```

- Una vez se acceda (tarda aproximadamente 30 segundos), cambiar la contraseña del administrador a una más segura.



```
[minwinpc] net user Administrator <new-password>
```

- Salir de esa sesión.

```
[minwinpc] Exit-PSSession
```

- Volver a acceder.

```
PS> Enter-PSSession -ComputerName minwinpc -Credential minwinpc\Administrator  
(esta vez con la nueva contraseña)
```

- Una vez dentro, cambiar el nombre del sistema.

```
[minwinpc] setcomputername <new-name>
```

- Reiniciar el dispositivo para guardar los cambios realizados.

```
[minwinpc] shutdown /r /t 0
```

- Mientras se reinicia, confiar en el dispositivo remoto, cambiando el nombre.

```
PS> Set-Item WSMAN:\localhost\Client\TrustedHosts -Value <new-name>
```

- De esta forma, una vez se reinicie se podrá acceder al dispositivo con los nuevos parámetros.

```
PS> Enter-PSSession -ComputerName <new-name> -Credential <new-name>\Administrator  
(utilizando la nueva contraseña)
```

### 8.2.3. Creación y enlace del IoT Hub

La creación y enlace de un **IoT Hub** (instancia de *Azure IoT Hubs*) es similar a la destacada en los apartados [7.1.1. \(Azure Event Hubs\)](#) y [7.1.3. \(Azure Stream Analytics\)](#) respectivamente. La sustitución es directa en esos apartados, ya que consiste en cambiar “*Event Hub*” por “*IoT Hub*”.

### 8.2.4. Adaptación y despliegue de la aplicación UWP

El desarrollo de la aplicación UWP se ha realizado con el mismo IDE que el cliente de la solución adoptada (*Visual Studio 2015 Community*), pero esta vez en la máquina virtual de *Windows 10* mencionada anteriormente. Esto se debe a que el despliegue a un dispositivo remoto como la *Raspberry Pi*, no es realizable mientras no sea un proyecto UWP. Por lo tanto, la aplicación de consola en C# desarrollada para la solución adoptada no es desplegable y debe ser adaptada a UWP.

El código es similar al cliente original, y prácticamente lo único que cambia son las credenciales utilizadas (en lugar de a un *Event Hub*, envía los datos a un *IoT Hub*). La única diferencia es que, en lugar de utilizar los paquetes definidos en la solución adoptada, necesitará un paquete denominado “*Windows.Azure.Devices.Client*” y



la referencia “Windows IoT Extensions for the UWP”, para conectarse con *Azure IoT Hubs* y controlar el diodo LED respectivamente.

El despliegue se realiza configurando el proyecto para que la ejecución del mismo se realice en un dispositivo remoto en lugar de en la máquina local (<Project-Name> → Properties → Debug [Target device: Remote Machine; Remote Machine: <Device-Name>; Authentication Mode: Universal (Unencrypted Protocol)]).

## 8.3. Problemas encontrados

---

A continuación, se define una recopilación de problemas encontrados (con soluciones, en el caso de que las tengan) y descripción de algunas limitaciones. Todos los problemas y limitaciones descritos están comprobados en **junio de 2016**.

### 8.3.1. Aplicación y Raspberry Pi

- No es posible desplegar la aplicación utilizando *Windows 7*. Esta es la razón por la que se utiliza una máquina virtual de *Windows 10* para desplegar la aplicación desarrollada a la *Raspberry Pi*. Referencia: <http://www.xataka.com/makers/que-puedes-y-que-no-puedes-hacer-con-windows-10-iot-core-en-las-raspberry-pi-2>, comentario #31 (david2200).
- La *Raspberry Pi* daña las tarjetas SD, por eso a veces se corrompe el *Kernel* de *Windows* y hace que la placa no sea accesible a través de la red. A pesar de que aportan una solución en el link de referencia, no es realmente válida, ya que es necesario disponer de varias cosas:
  - Un sistema *Windows 10* (máquina virtual).
  - La aplicación gratuita [Windows 10 IoT Core Dashboard](#), instalada en la máquina virtual.
  - Adaptador de MicroSD (USB).

La solución consiste en reinstalar el sistema *Windows 10 IoT Core* en la tarjeta dañada, utilizando la herramienta que ofrece *Microsoft*. Sin embargo, es necesario hacerlo con un adaptador de microSD que sea USB. Esto se debe a que, si se dispone de un adaptador de tarjetas integrado, a pesar de que esta se pueda detectar como un disco duro más, la aplicación no la detectará como una tarjeta SD.

Sin embargo, si se utiliza un adaptador USB, la máquina virtual detectará un nuevo dispositivo universal, detectando este como una tarjeta SD. En ese punto, la herramienta puede detectar la tarjeta como una SD y puede procederse a la reinstalación del sistema operativo.

- Error “{Message: The specified SAS token is expired}”. Problema expuesto por *Eric* en el link de referencia. Esto puede surgir el día siguiente en el que todo funcionaba correctamente. Hay que tener en

cuenta que las firmas de acceso compartido disponen de fecha de caducidad. También hay que tener en cuenta que la *Raspberry Pi* no dispone de pila interna, con lo que si se desconecta de la red eléctrica, el reloj no avanza. Por lo tanto, la razón por lo que ocurre esto es que como el reloj está muy atrasado con respecto al gestor de firmas de acceso compartido, el dispositivo pide un nuevo *token*. Para el gestor de *tokens* siempre estarán caducados y, por consiguiente, revoca los envíos realizados por la *Raspberry Pi*.

La solución a este problema es simple. Una vez se accede a la *Raspberry Pi* a través de PowerShell (como se ha mencionado en el apartado anterior), ejecutando los comandos

```
[<new-name>] set-date MM/DD/YYYY  
y  
[<new-name>] set-date HH:MM<AM/PM>
```

es posible ajustar la fecha y hora de la *Raspberry Pi* respectivamente. De esta forma, se resuelve este problema.

### 8.3.2. Azure IoT Hubs (ingestión de la información)

Los mismos descritos en el [apartado 7.2.1. \(Azure Event Hubs\)](#).

## 9. Gestión del proyecto

En este apartado se muestra la planificación del proyecto, como se ha desarrollado a lo largo de su ejecución y cuál ha sido su desviación.

### 9.1. Planificación

En este apartado se describe cual es la planificación seguida en el proyecto.

#### 9.1.1. Identificación de tareas

La definición de tareas no ha sido trivial, puesto que desde el inicio del proyecto, se ha intentado adaptarse tanto al contexto como al estado del arte del mismo.

A consecuencia de esto, ha habido varias replanificaciones desde la planificación inicial, las cuales añadían y eliminaban distintas tareas. Esto ha ocurrido debido a que, a pesar de que todavía se estaba estudiando el contexto y el estado del arte cuando se desarrolló la primera versión de la planificación, no se había definido ningún caso de uso a desarrollar.

Esto se debe a que la oferta del proyecto ofrecida por la empresa tenía una descripción muy genérica, dónde a base de aprender sobre las tecnologías del ámbito, se acabo por definir dicho caso de uso. Esta definición se terminó de declarar en mitad del desarrollo del proyecto.

Por lo tanto, a continuación se van a definir las tareas resultantes de todas esas replanificaciones, siendo consciente de que en esto no consiste la identificación de tareas, sino en identificar las tareas a partir de una planificación inicial.

<b>Nombre de la tarea:</b>	<b>R: Reuniones</b>
<b>Periodo de desarrollo estimado:</b>	25/01/2016 (S1) – 10/06/2016 (S20)
<b>Descripción:</b>	Asistencia a reuniones con el tutor de la empresa para comunicar el progreso semanal del proyecto. Seguimiento y control del proyecto.
<b>Dedicación estimada:</b>	1 hora por reunión

<b>Nombre de la tarea:</b>	<b>C: Capacitación general del proyecto</b>
<b>Periodo de desarrollo estimado:</b>	25/01/2016 (S1) – 19/02/2016 (S4)
<b>Descripción:</b>	Capacitación sobre el estado del arte y el contexto del proyecto.
<b>Dedicación estimada:</b>	160 horas

<b>Nombre de la tarea:</b>	<b>T1: Publicador de prueba</b>
<b>Periodo de desarrollo estimado:</b>	15/02/2016 (S4) – 25/03/2016 (S9)
<b>Descripción:</b>	Capacitación teórico-práctica en el desarrollo de publicadores y desarrollo de un cliente.
<b>Dedicación estimada:</b>	49 horas

<b>Nombre de la tarea:</b>	<b>T2: Capacitación sobre <i>Azure Event Hubs</i></b>
<b>Periodo de desarrollo estimado:</b>	15/02/2016 (S4) – 08/04/2016 (S9)
<b>Descripción:</b>	Capacitación teórico-práctica sobre este sistema de mensajería distribuida de <i>Azure</i> .
<b>Dedicación estimada:</b>	61 horas

<b>Nombre de la tarea:</b>	<b>T3: Capacitación sobre <i>Stream Analytics</i></b>
<b>Periodo de desarrollo estimado:</b>	22/02/2016 (S5) – 08/04/2016 (S9)
<b>Descripción:</b>	Capacitación teórico-práctica sobre este procesador de flujos de datos de <i>Azure</i> .
<b>Dedicación estimada:</b>	85 horas

<b>Nombre de la tarea:</b>	<b>T4: Capacitación sobre <i>Power BI</i></b>
<b>Periodo de desarrollo estimado:</b>	29/02/2016 (S6) – 08/04/2016 (S11)
<b>Descripción:</b>	Capacitación teórico-práctica sobre esta herramienta para visualización de <i>Office 365</i> .
<b>Dedicación estimada:</b>	61 horas

<b>Nombre de la tarea:</b>	<b>T5: Capacitación sobre <i>DocumentDB</i></b>
<b>Periodo de desarrollo estimado:</b>	07/03/2016 (S7) – 15/04/2016 (S12)
<b>Descripción:</b>	Capacitación teórico-práctica sobre este servicio de bases de datos NoSQL.
<b>Dedicación estimada:</b>	49 horas

<b>Nombre de la tarea:</b>	<b>T6: Capacitación sobre <i>Machine Learning</i></b>
<b>Periodo de desarrollo estimado:</b>	29/02/2016 (S6) – 15/04/2016 (S12)
<b>Descripción:</b>	Capacitación teórico-práctica sobre este servicio para realizar experimentos de análisis de datos.
<b>Dedicación estimada:</b>	37 horas

<b>Nombre de la tarea:</b>	<b>T7: Pruebas de casos reales</b>
<b>Periodo de desarrollo estimado:</b>	11/04/2016 (S12) – 29/04/2016 (S14)
<b>Descripción:</b>	Realizar pruebas con casos propuestos por el tutor de la empresa.
<b>Dedicación estimada:</b>	44 horas

<b>Nombre de la tarea:</b>	<b>T12: Desarrollo del sistema predictivo</b>
<b>Periodo de desarrollo estimado:</b>	21/03/2016 (S9) – 13/05/2016 (S16)

<b>Descripción:</b>	Implementación del caso de uso definido sobre la monitorización del aerogenerador.
<b>Dedicación estimada:</b>	139 horas

<b>Nombre de la tarea:</b>	<b>T13: Capacitación sobre Azure SQL</b>
<b>Periodo de desarrollo estimado:</b>	21/03/2016 (S9) – 08/04/2016 (S11)
<b>Descripción:</b>	Capacitación teórico-práctica sobre este servicio de bases de datos relacionales.
<b>Dedicación estimada:</b>	30 horas

<b>Nombre de la tarea:</b>	<b>T10.M: Redacción de la memoria</b>
<b>Periodo de desarrollo estimado:</b>	25/01/2016 (S1) – 10/06/2016 (S20)
<b>Descripción:</b>	Redacción de la memoria del proyecto.
<b>Dedicación estimada:</b>	119 horas

<b>Nombre de la tarea:</b>	<b>T11.P: Preparar presentación del proyecto</b>
<b>Periodo de desarrollo estimado:</b>	25/01/2016 (S1) – 10/06/2016 (S20)
<b>Descripción:</b>	Elaborar las transparencias de apoyo y practicar la presentación.
<b>Dedicación estimada:</b>	20 horas

### 9.1.2. Diagrama de Gantt

A continuación, se define el diagrama donde se muestra gráficamente el desarrollo del proyecto sobre las tareas que se han ido planificando en el apartado anterior.

Cada mes está dividido en cuatro periodos, ya que la granularidad que se ha utilizado para realizar la gestión del proyecto ha sido de **una semana**. Las dependencias entre tareas se marcan con flechas azules.

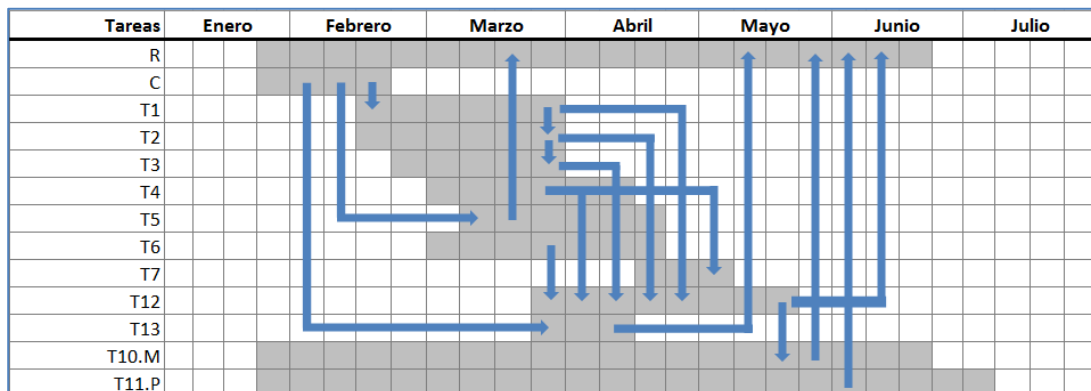


Ilustración 85. Diagrama de Gantt

### 9.1.3. EDT

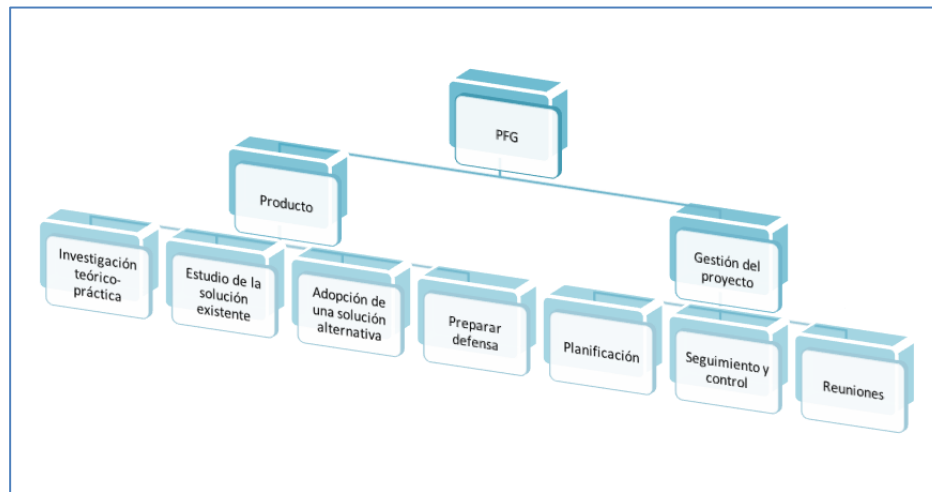


Ilustración 86. EDT

### 9.1.4. Gestión de riesgos

En este apartado se toman en cuenta los distintos riesgos que pueden acaecer en pleno desarrollo del proyecto y cuáles serían las soluciones a los mismos.

- i. Indisposición: En caso de no encontrarse en disposición de seguir con el proyecto por causa de alguna enfermedad, se aplazarán las tareas hasta recuperarse de la misma. Al desarrollar el proyecto en empresa, además de avisar al tutor de la universidad (Oscar Díaz), se deberá avisar al tutor de la empresa (Cristóbal Arellano) para que sean conscientes de este evento.
- ii. Pérdida de información parcial o total: Este riesgo se solventa realizando respaldos de la información mediante copias de seguridad. Los datos originales se almacenan en el terminal de trabajo proporcionado por la empresa y las copias se realizan en una cuenta de *Google Drive*. De este modo, la copia es accesible desde cualquier lugar. Además, como medida de seguridad adicional, cada primero de mes se hace una copia total de la información residente en la nube a un disco duro extraíble de gran capacidad, cuya única función es hacer copias de seguridad y restablecer información perdida.
- iii. Pérdida del puesto de trabajo: En el caso de que este fallara de forma prolongada, y dado que es de la empresa, esta debería de ofrecer una reparación del terminal o proveer uno nuevo. En la segunda situación, no habría gran problema, ya que, a pesar de que la instalación de Visual Studio suele durar unos minutos, la mayor parte del proyecto se desarrolla en una plataforma a través de Internet. Con lo cual, con tener instalado un navegador, el proyecto podría proseguir casi sin interrupciones.
- iv. Incumplimiento de plazos: En el caso de que las tareas tomen más tiempo de lo esperado, se realizarán reajustes de los tiempos, o replanificaciones en casos extremos.

### 9.1.5. Gestión de comunicaciones

Para mantener la comunicaciones con los tutores se utilizará el correo corporativo de la empresa, para comunicar avances, planificar reuniones de seguimiento, informar de cambios en la planificación, comentar problemas,...

Eventualmente, se utilizará el correo corporativo de la universidad para mantenerse en contacto con el tutor de la universidad (Oscar Díaz) para realizar consultas y dudas respecto al proyecto.

### 9.1.6. Análisis de viabilidad

La identificación de tareas en la primera planificación no podía ser certera, debido a la baja visibilidad sobre los objetivos del proyecto. Mientras tanto, se toma como referencia de periodo de desarrollo, el tiempo de estancia en la empresa.

La viabilidad en este proyecto, depende de en qué momento del mismo se analice. Al comienzo del proyecto, se podría decir que es completamente **inviable**, puesto que dispone de una muy alta incertidumbre, mientras que a la mitad del desarrollo se puede decir que es perfectamente viable. A continuación, se muestra la incertidumbre del proyecto en el tiempo (por semanas).

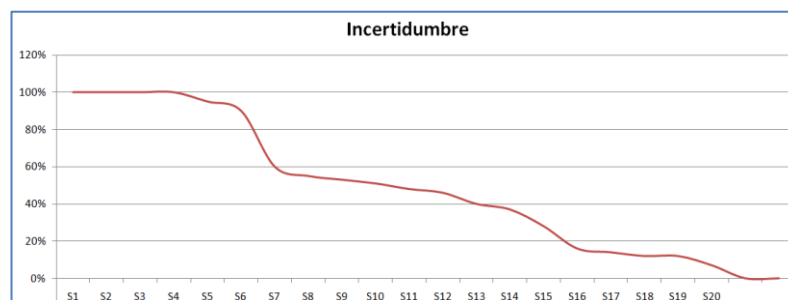


Ilustración 87. Incertidumbre del proyecto

Cuanta mayor incertidumbre, menor es la posibilidad de cuantificar en qué grado se enmarcaba la completitud de las tareas planificadas (uno de los riesgos potenciales definidos), y por lo tanto, mayor es la posibilidad de cometer errores a la hora de acertar en la planificación de tareas. Esto es lo que ocasionó la sucesión de replanificaciones a lo largo de todo el proyecto; **una especificación inicial demasiado abierta** sobre el objetivo del proyecto.

Como ya se ha comentado, al comienzo sería un proyecto inviable. Sin embargo, en una etapa más avanzada, se puede comprobar que es viable, ya que hay mayor visibilidad sobre los eventos futuros y se pueden controlar el progreso del proyecto con mayor flexibilidad.

## 9.2. Seguimiento y control

Como bien se ha comentado en el apartado anterior, se han tenido que realizar varias replanificaciones y reajustes sobre la planificación inicial, tanto para la definición de objetivos como para gestionar riesgos (incumplimiento de plazos). Estos ajustes se hacen teniendo en cuenta el costo estimado (en tiempo) de cada tarea. A continuación, se describe una tabla comparativa de las tareas definitivas del proyecto sobre el tiempo real y el estimado, donde se explica el motivo de la desviación. Se debe añadir que algunas desviaciones pueden deberse a replanificaciones, consecuentes de adición o eliminación de tareas.

Tarea	Estimado	Realidad	Desviación	Motivo
<b>R</b>	20h	23h 30'	+3h 30'	Algunas reuniones de seguimiento a veces se alargaban un poco más de la cuenta.
<b>C</b>	160h	119h	-41h	En la fase de capacitación, se identificaron varios aspectos para decantarse por una definición del proyecto. Por lo tanto, no fue necesaria más capacitación.
<b>T1</b>	49h	33h 30'	-15h 30'	Desarrollar un publicador no fue tan complicado como se esperaba.
<b>T2</b>	61h	59h	-2h	Capacitarse en <i>Azure Event Hub</i> y entender su funcionamiento interno, prácticamente se ha realizado en el tiempo estimado.
<b>T3</b>	85h	48h 30'	-36h 30'	Capacitarse en <i>Azure Stream Analytics</i> no ha llevado tanto tiempo como se estimaba.
<b>T4</b>	61h	36h	-25h	Capacitarse en <i>Power BI</i> no ha llevado tanto tiempo como se estimaba.
<b>T5</b>	49h	1h	-48h	Esta tarea se tuvo que cancelar para poder realizar otras más significativas con la línea del proyecto.
<b>T6</b>	37h	81h	+44h	Ha llevado más tiempo, puesto que es la parte nuclear del proyecto y tiene más detalles que cualquier



				otra parte.
<b>T7</b>	44h	9h 15'	-34h 45'	Se desarrollaron unas pruebas, pero a continuación, se canceló esta tarea.
<b>T12</b>	139h	103h 45'	-35h 15'	El desarrollo del caso de uso principal, de media, se ha conseguido realizar en menos tiempo del estimado. Esto se debe a que algunos aspectos que podían llevar a una prolongación en su desarrollo, ya se trataron en las correspondientes tareas sobre capacitación.
<b>T13</b>	---	4h 30'	4h 30'	No se estimó nada, ya que iba a ser una tarea donde se pretendía realizar una pequeña capacitación. Sin embargo, se echó atrás ya que otras tareas tenían mayor preferencia.
<b>T10.M</b>	119h	134h 15'	+15h 15'	La dedicación estimada para la redacción de la memoria ha sido más de la que en un principio se estimaba.
<b>T11.P</b>	---	---	---	Realizado fuera de la planificación.

Para terminar, se muestra como la dedicación real en el tiempo de las tareas anteriormente descritas. Para ello, se muestra el diagrama de Gantt en ausencia de la dependencia entre tareas. Para entender el esquema, hay que tener en cuenta lo siguiente:

- La dedicación fuera del plazo planificado se marca en rojo.
- La dedicación antes de iniciar el plazo (o dedicación temprana) se marca en verde.
- Las replanificaciones se marcan con sucesiones verticales de asteriscos rojos (en la semana que se realiza).
- La realización de la planificación inicial se marca con una sucesión vertical de asteriscos verdes (en la semana que se realiza).
- Las terminaciones definitiva de tareas (es decir, que se planifica para que no se continúe desarrollando) se marca con una equis roja.

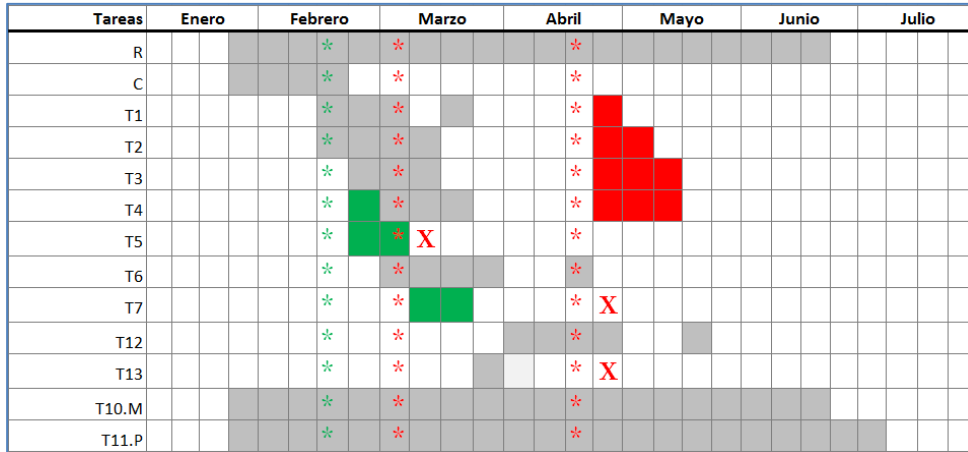


Ilustración 88. Diagrama de Gantt con desviaciones

La razón por la cual algunas de las tareas han llegado a realizarse fuera de plazo es que, se decidió que la temática era interesante y se profundizó más en ellas. Por tanto, fueron necesarias realizar varias pequeñas pruebas, para poder entender mejor varios aspectos de las comentadas tecnologías.

La razón de la dedicación temprana de algunas tareas fue por el hecho de que dichas tareas podían iniciarse, sin afectar al desarrollo de las que ya se estaban realizando.

La terminación de algunas tareas se debe a que, o bien dichas tareas no iban a ser necesarias para el desarrollo del caso de uso final, o porque se considera que la dedicación a dicha tarea fue suficiente.

Por último, y como se ha comentado anteriormente, la razón de las replanificaciones es debido a la alta incertidumbre del proyecto (sobre todo al comienzo del mismo, ver **ilustración 81**).

---

## 10. Conclusiones

---

- Este acercamiento a un sistema de monitorización para la predicción de fallos, ofrece la posibilidad a las empresas de producción de energía eólica de aprovechar una monitorización preventiva de sus estructuras.
- Las tareas definidas como capacitaciones de las tecnologías, han ayudado a afrontar, con mayor facilidad, problemas potenciales que podrían haber surgido a la hora de desarrollar el caso de uso definido para el proyecto. Gracias a esto, el desarrollo del caso de uso no ha sufrido desviaciones, y no han existido grandes complicaciones a la hora de hacer pruebas y ver que el funcionamiento del mismo es el esperado.
- Las dedicaciones realizadas fuera de plazo se podrían definir como **desviaciones positivas**, ya que han ayudado a profundizar en el conocimiento de las tecnologías utilizadas en el proyecto.
- El desarrollo del sistema predictivo (núcleo del proyecto) ha consistido en la implementación de un diseño previamente existente, utilizando un lenguaje para análisis estadístico denominado “R”, del cual se requiere tener conocimientos avanzados, tanto a la hora de desarrollar el caso de uso como a la hora de adaptarlo a los experimentos, con el objetivo de prever posibles errores; los cuales no son fáciles de resolver.
- Muchos de los enlaces descritos en el [apartado 7.2.](#) hacen referencia a foros que buscan mejoras para los servicios y tecnologías descritos (foros del estilo “¿Cómo podemos mejorar esta tecnología?”). Esto se debe a que *Microsoft* (en el ámbito de las tecnologías basadas en la nube) es “menos veterano” que competidores con servicios y tecnologías mejor definidas y más completas. Amazon puede considerarse como un competidor veterano en este ámbito ([Amazon Web Services](#)).
- Los servicios y tecnologías utilizados han ido evolucionando (y siguen evolucionando) según se ha ido desarrollando el proyecto. Esto hace ver que, a pesar de que los tecnologías y servicios de la nube de *Microsoft* actualmente tienen limitaciones, están en **constante evolución** y, en un futuro, podrán llegar a tener unos servicios mejor definidos y más completos, como lo son los que provee *Amazon Web Services*.
- Las soluciones basadas en la nube son populares entre las PYME, ya que estas se ahorran, entre otras cosas, gastos en infraestructuras y su mantenimiento.
- La solución alternativa de monitorización preventiva desarrollada, sirve de toma de contacto a los investigadores de IK4-Ikerlan, para tener una especie de prueba de concepto desarrollada utilizando tecnologías de *Microsoft*. Gracias a esto tienen la oportunidad de aprender el funcionamiento de algunas de las tecnologías y servicios que *Microsoft* provee mediante su plataforma, y que progresivamente puedan desarrollar soluciones de mayor escala con estas, frente a clientes que requieran soluciones de negocio en este ámbito.
- Como conclusión final, he aprendido que el desarrollo de soluciones basadas en la nube puede favorecer a una solución escalable, accesible,

segura y económica. Por el contrario, puede ocasionar un problema en el desarrollo de una solución de grandes dimensiones, en el caso de que los servicios y tecnologías de la plataforma estén evolucionando o simplemente en mantenimiento. En el caso de que estén en mantenimiento, puede ser un contratiempo menor que afecte al desarrollo. Sin embargo, en el caso de que estén evolucionando (caso en el que se ha desarrollado la solución adoptada), puede que los gestores de la plataforma creen, modifiquen o incluso eliminen servicios de la misma. Esto último puede afectar a una solución en pleno desarrollo, teniendo que replantear su diseño, en función de dichas modificaciones realizadas por los gestores anteriormente mencionados.

- Como opinión personal, creo que este tipo de soluciones (las basadas en la nube), pueden ser importantes y útiles herramientas para empresas que estén comenzando con su negocio y que no les importe subir su información a una plataforma pública. Además, considero que las plataformas como servicio (siempre y cuando no estén cambiando constantemente), disponen de potentes tecnologías y servicios que, una vez se saben manejar, se pueden desarrollar interesantes soluciones con ellas en un plazo bastante reducido de tiempo.

---

## 11. Líneas futuras

---

A continuación se describen algunas de las líneas futuras que puede tomar este proyecto.

- Extender el servicio de *Machine Learning* con fuentes de datos que tengan capacidad de cambiar independientemente del servicio (por ejemplo, bases de datos). Puntero de referencia: <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-import-data-from-online-sources/>
- Adaptar varias fases de análisis y procesado de datos con HDInsight (Hadoop). Puntero de referencia: <https://azure.microsoft.com/es-es/services/hdinsight/>
- Sustituir la parte de visualización por otra tecnología o servicio más personalizable, flexible y escalable.
- Dotar a la solución aportada de más publicadores, y monitorización de lo que estos envían a la plataforma.
- Dotar a la solución aportada de más consumidores, como por ejemplo, bases de datos que actúan como históricos.
- Crear un agente de monitorización para los modelos predictivos, que compruebe la confiabilidad de las predicciones, y sea capaz de reentrenarlos en su debido momento, utilizando estados de modelos entrenados previamente almacenados. Puntero de referencia: <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-retrain-models-programmatically/>
- Enlazar una VPC de AWS con una red virtual de *Azure*, con el fin de mantener ambas redes interconectadas en un entorno LAN y poder desplegar máquinas virtuales que dispongan de visibilidad a nivel de red local de ambos entornos. Además, estas máquinas pueden aprovechar los servicios de ambas plataformas. Puntero de referencia: <http://geeks.ms/enterprise/2015/04/26/conectando-microsoft-azure-con-amazon-web-services/>
- Basándose en el [apartado 8](#), extenderlo aún más, haciendo que todo el sistema disponga de una inteligencia superior, haciendo que, aparte de que los dispositivos se comuniquen con la nube, la nube “hable” con los dispositivos (comunicación bidireccional).

---

## 12. Referencias

---

- Acciona. (2014). *¿Qué es un aerogenerador?* Recuperado de <http://www.acciona.com/es/energias-renovables/energia-eolica/aerogeneradores/>
- Andy De George. Abril 19 de 2016. *Configuring SSL for an application in Azure* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/cloud-services-configure-ssl-certificate/>
- Anoop P. Verma. (2012). *Performance monitoring of wind turbines : a datamining approach (Tesis doctoral, University of Iowa)* Recuperado de <http://ir.uiowa.edu/cgi/viewcontent.cgi?article=3343&context=etd>
- Azure. Abril 22 de 2016. *Windowing (Azure Stream Analytics)* Recuperado de <https://msdn.microsoft.com/en-us/library/azure/dn835019.aspx>
- Benjamin GUINEBERTIERE. Septiembre 24 de 2014. *How to upload an R package to Azure Machine Learning* Recuperado de <https://blogs.msdn.microsoft.com/benjuin/2014/09/24/how-to-upload-an-r-package-to-azure-machine-learning/>
- Build Azure. Septiembre 23 de 2015. *What is Azure Service Bus?* Recuperado de <https://buildazure.com/2015/09/23/what-is-azure-service-bus/>
- cacsar. Febrero 3 de 2015. *Answer to "Azure eventhub multiple partition key points to same partition"* Recuperado de <http://stackoverflow.com/questions/28292330/azure-eventhub-multiple-partition-key-points-to-same-partition>
- C.J. Gronlund. Abril 28 de 2016. *Introduction to machine learning on Microsoft Azure* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-what-is-machine-learning/>
- Cortana Intelligence and ML Blog Team. Junio 1 de 2015. *The Azure Stream Analytics Query Language* Recuperado de <https://blogs.technet.microsoft.com/machinelearning/2015/06/01/the-azure-stream-analytics-query-language/>
- Dan Rosanova MSFT. Febrero 2 de 2015. *Event Hub Publisher Policy in Action* Recuperado de <https://blogs.msdn.microsoft.com/servicebus/2015/02/02/event-hub-publisher-policy-in-action/>
- DataScienceDojo. (2016). *Building custom R models in Azure Machine Learning Studio* Recuperado de <http://datasciencedojo.com/custom-r-models-azure-ml/>
- Hiteshmadan. Febrero 10 de 2016. *Creating Endpoints* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-create-endpoint/>
- Interoute. (2013). *¿Qué es PaaS?* Recuperado de <http://www.interoute.es/what-paas>
- Gary Ericson. Marzo 9 de 2016. *Machine learning tutorial: Create your first experiment in Azure Machine Learning Studio* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-create-experiment/>

- Gary Ericson. Marzo 9 de 2016. *Walkthrough Step5: Deploy the Azure Machine Learning web service* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-walkthrough-5-publish-web-service/>
- Gary Ericson. Abril 18 de 2016. *Azure Machine Learning Frequently Asked Questions (FAQ): Billing, capabilities, limitations, and support* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-faq/>
- Gary Ericson. Mayo 22 de 2016. *How to consume an Azure Machine Learning web service that has been deployed from a Machine Learning experiment* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-consume-web-services/>
- Ginger Grant. Noviembre 27 de 2015. *Incorporating Azure Streaming Analytics with Azure ML – Part 1.* Recuperado de <http://www.desertislesql.com/wordpress1/?p=954>
- James van den Berg. Septiembre 10 de 2014. *Overview poster of Azure features, services, and common uses* Recuperado de <https://mountainss.wordpress.com/2014/09/10/overview-poster-of-azure-features-services-and-common-uses-azure-cloud-sysctr-hyperv/>
- Jeff Stokes. Mayo 3 de 2016. *How to create a data analytics processing job for Stream Analytics* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/stream-analytics-create-a-job/>
- Jeff Stokes. Mayo 3 de 2016. *Tutorial: Perform sentiment analysis using Stream Analytics and Machine Learning* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/stream-analytics-machine-learning-integration-tutorial/>
- Jeff Stokes. Mayo 3 de 2016. *Stream Analytics & Power BI: A real-time analytics dashboard for streaming data* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/stream-analytics-power-bi-dashboard/>
- Jeff Stokes. Mayo 3 de 2016. *What is Stream Analytics?* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/stream-analytics-introduction/>
- Larry Franks. Abril 22 de 2016. *Quickstart tutorial for the R programming language for Azure Machine Learning* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-r-quickstart/>
- Ipanzano. Julio 18 de 2010. *La plataforma Windows Azure* Recuperado de <https://blogs.msdn.microsoft.com/luispanzano/2010/07/18/la-plataforma-windows-azure/>
- m00nbeam360. Agosto 20 de 2014. *Azure Subscription ID vs Account ID* Recuperado de <http://stackoverflow.com/questions/25411835/azure-subscription-id-vs-account-id>
- Marcin Szeliga. Enero 25 de 2015. *Pros and Cons of Azure Machine Learning* Recuperado de <http://blog.sqlexpert.pl/2015/01/25/pros-and-cons-of-azure-machine-learning/>
- mart. Octubre 20 de 2015. *PLATFORM AS A SERVICE (PAAS)* Recuperado de <http://brainymart.com/?p=9>



- mkarich. Mayo 5 de 2011. *¿Qué es Office 365?* Recuperado de <https://blogs.technet.microsoft.com/microsoftlatam/2011/05/05/qu-es-office-365/>
- MSDN. Diciembre 7 de 2015. *Overview of Power BI REST API* Recuperado de <https://msdn.microsoft.com/en-us/library/dn877544.aspx>
- Nino Crudele. Diciembre 12 de 2014. *Azure Event Hub – all my thoughts* Recuperado de <https://ninocrudele.me/2014/12/12/azure-event-hub-all-my-thoughts/>
- Office Support. (2016). *Yammer integration with Office 365* Recuperado de <https://support.office.com/en-us/article/Yammer-integration-with-Office-365-4086681f-6de1-4d39-aa72-752b2af1cbd7>
- OneOfSome. Febrero 9 de 2015. *Event Hubs Publisher Policy – Explained* Recuperado de <http://readtocode.blogspot.com.es/2015/02/event-hubs-publisher-policy-explained.html>
- Pixabay. (2016). Varias imágenes recuperadas del dominio <https://pixabay.com>
- Power BI (2016). *Power BI documentation* Recuperado de <https://powerbi.microsoft.com/en-us/documentation/powerbi-service-get-started/>
- RaymondL. Mayo 23 de 2016. *Retrain Machine Learning model programmatically* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-retrain-models-programmatically/>
- Seth Manheim. Marzo 9 de 2016. *Azure Service Bus* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/service-bus-fundamentals-hybrid-solutions/>
- Seth Manheim. Abril 15 de 2016. *Azure Event Hubs overview* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/event-hubs-overview/>
- Seth Manheim. Abril 15 de 2016. *Event Hubs API overview* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/event-hubs-api-overview/>
- Seth Manheim. Abril 15 de 2016. *Event Hubs programming guide* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/event-hubs-programming-guide/>
- Steve Weston & Rich Calaway. Octubre 13 de 2015. *Getting Started with doParallel and foreach* Recuperado de <https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf>
- Tamra Myers. Abril 25 de 2016. *Get started with Azure Blob storage using .NET* Recuperado de <https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/>
- TechNet (Microsoft). Mayo 19 de 2005. *Understanding Digital Certificates* Recuperado de <https://technet.microsoft.com/en-us/library/bb123848%28v=exchg.65%29.aspx>
- Twenergy. (2012). *¿Qué son las energías renovables?* Recuperado de <http://twenergy.com/a/que-son-las-energias-renovables-516>
- Yukei. Julio 16 de 2014. *Cómo usar un certificado SSL autofirmado de forma segura* Recuperado de <https://www.yukei.net/2014/07/certificado-ssl-autofirmado-seguro/>



---

## 13. Anexo: Glosario de términos

---

En este apartado se introducen una serie de términos cuyo significado no es obvio, con el objetivo de facilitar la lectura del proyecto. Los términos, que están ordenados alfabéticamente, son los siguientes:

- **Acción:** Actividad que realiza un módulo para procesar el *dataset* entrante o relacionar varios (si los tiene. En caso de que no tenga, su acción consiste en generar la información u obtenerla de alguna manera). Cada módulo tiene una acción diferente.
- **API:** *Application Programming Interface* o Interfaz de Programación de Aplicaciones. Conjunto de funciones que ofrece una biblioteca desarrollada por terceros, para ser utilizada por aplicaciones propias, de manera segura y confiable.
- **Área de trabajo o *workspace*:** Instancia de *Machine Learning* donde se desarrollan y almacenan experimentos, conjuntos de datos (o *datasets*), servicios web desplegados, modelos entrenados y otro tipo de recursos relacionados con este servicio. Actualmente (2016), las regiones disponibles para este servicio son: Oeste de Europa, Sudeste de Asia y Estados Unidos del Sur-Central.
- **Anemómetro:** Sensor de un aerogenerador que mide la velocidad del viento.
- **Barquilla:** Cápsula donde se encuentra la maquinaria en el extremo superior de la torre de un aerogenerador. También se le suele denominar **góndola**.
- **Blob:** Archivo de cualquier tipo y tamaño.
- **Buje:** Pieza cilíndrica y parte del rotor que transfiere el movimiento de las palas del aerogenerador al eje lento.
- **Centro de datos (o *datacenter*):** Ubicación donde se concentran los recursos necesarios para el procesamiento de la información de una organización.
- **Certificado autofirmado:** Firma electrónica que acredita una identidad y sus credenciales, asegurando las comunicaciones punto a punto a nivel de transporte (*SSL/TLS*), y que no existen terceros que puedan validar la identidad del mismo.
- **Comunicación punto a punto:** Conexión limitada a dos extremos, donde existe una relación emisor-receptor en ambos sentidos.
- **Contenedor:** Espacio donde se almacena un conjunto de *blobs*. Puede almacenar un número “ilimitado” de *blobs*.
- **Corona:** Elemento rotatorio que permite a la barquilla una libertad de giro de 360° encima de la torre.
- **Cuenta de almacenamiento:** Instancia del servicio *Azure Blob Storage* que sirve para almacenar objetos o *blobs* de gran tamaño, dentro de uno de sus contenedores. Puede disponer de un número “ilimitado” de contenedores.
- **Dataset:** Conjunto de datos estructurado, generalmente originario de algún proveedor de datos, que tienen organizada la información en formato tabular y actúa como fuente de datos. Las filas son registros, y las columnas, los distintos campos existentes por cada registro; similar a las tablas de una base

de datos. En *PowerBI* ([apartado 5.2.2.](#)), es un conjunto de tablas en lugar de ser una sola.

- Eje lento: Parte del rotor que transfiere el movimiento a la multiplicadora.
- Eje rápido: Eje que transfiere el movimiento de la multiplicadora al generador.
- Estado: Fase conductual en el que se sitúa un aerogenerador.
- Espacio de nombres o *namespace*: Instancia del servicio *Azure Service Bus*. Desde él se gestionan servicios como *Azure Event Hubs*.
- *Event Hub*: Instancia del servicio *Azure Event Hubs*.
- Evento o *EventData*: Estructura de datos con la que se manipula la información que entra o sale de un *Event Hub*.
- Experimento: Grupo de módulos relacionados entre sí, los cuales en su conjunto definen una lógica de procesamiento.
- Generador: Parte del aerogenerador que convierte la energía mecánica en energía eléctrica.
- Incertidumbre: Situación donde los resultados no son conocidos y cuyas probabilidades de ocurrencia no se pueden cuantificar. Tampoco se conoce la probabilidad de que ocurra un desenlace, lo que puede afectar negativamente a la gestión de riesgos.
- Instancia: Recurso generado dentro de un servicio. Por ejemplo, en un servicio de bases de datos ofrecidas por una plataforma, una base de datos sería una instancia.
- *IoT Hub*: Instancia del servicio *Azure IoT Hubs*.
- *Machine Learning*: Disciplina científica que trata de que los sistemas aprendan automáticamente. Aprender en este contexto quiere decir identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. En este proyecto, este concepto se utiliza más para nombrar un servicio ofrecido por el *PaaS* de *Microsoft Azure* ([apartado 5.1.4.](#)).
- Modelo predictivo: Estructura y proceso para predecir valores de variables especificadas en un conjunto de datos.
- Módulo: Componente que ejecuta una **acción** con uno o varios *datasets*, que a continuación devuelve un *dataset*, como resultado de dicha ejecución.
- Multiplicadora: Elemento del aerogenerador que multiplica las revoluciones de giro del rotor.
- Partición: Instancia reservada para la ingesta y consumo de la información dentro de *Azure Event Hubs*. Dentro de esta, la información fluye y se persiste temporalmente, utilizando una estructura de datos denominada “*EventData*” para manipular la información.
- Patrón de estados: Secuencia de estados, que se generan en un periodo de tiempo determinado y que conllevan una avería.
- *Random Forest*: Uno de los mejores algoritmos de clasificación, capaz de organizar grandes cantidades de datos con exactitud. Es una combinación de árboles predictores en los que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.
- Región: Hace referencia a una división territorial a nivel mundial, donde se sitúan los distintos centros de datos de *Microsoft*.

- Reglas de Asociación: Algoritmo de *Machine Learning* que se utiliza para descubrir acontecimientos en común dentro de un determinado conjunto de datos.
- Riesgo: Variabilidad en el desarrollo del proyecto, que puede llevar a desviaciones respecto a la planificación.
- Rotor: Conjunto de tres palas engarzadas en el buje y el eje lento.
- Servicio web: Interfaz de software para describir una operación a ejecutar, o datos a intercambiar con otro servicio web, accesible a través de la red, usando protocolos basados en el lenguaje XML.
- Sistema predictivo: Sistema dotado de uno o varios modelos predictivos.
- Sniffer: Programa que analiza indiscriminadamente el tráfico de la red en la que se encuentra.
- Torre: Elemento que sitúa el generador a una mayor altura, donde los vientos son de mayor intensidad. También permite el giro de las palas y transmite la energía eléctrica generada a las líneas subterráneas para su posterior distribución.
- Twenergy: Comunidad online creada por *Endesa* con el objetivo de servir de referencia en el campo de la eficiencia energética y el desarrollo sostenible, ofreciendo pautas para un consumo responsable de energía.
- Veleta: Sensor de un aerogenerador que mide la dirección del viento.