Universitat Oberta de Catalunya

uoc.edu

Desarrollo e implementación de modelos de Machine Learning para aplicaciones de gestión y eficiencia energética

Tomás Ruiz Brückel
Máster Universitario en Ingeniería Informática
Business Intelligence

David Amorós Alcaraz
Ferran Prados Carrasco

08/01/2020

Ficha de trabajo final

Título del trabajo:	*Desarrollo e implementación de modelos de Machine Learning para aplicaciones de gestión y eficiencia energética*
Nombre del autor:	*Tomás Ruiz Brückel*
Nombre del consultor/a:	*David Amorós Alcaraz*
Nombre del PRA:	*Ferran Prados Carrasco*
Fecha de entrega (01/2020):	*01/2020*
Titulación:	*Máster Universitario en Ingeniería Informática*
Área del Trabajo Final:	*El nombre de la asignatura de TF*
Idioma del trabajo:	*Business Intelligence*
Palabras clave	

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El propósito del proyecto es el desarrollo de un sistema de **Aprendizaje Automático** que permita realizar predicciones de consumo de suministros eléctricos. También debe permitir la detección de errores de datos de series temporales de consumo para su posterior análisis y corrección. Todo ello tomando como datos de entrenamiento el histórico de series temporales de cada uno de los suministros eléctricos proporcionados por las comercializadoras.

El enfoque del proyecto es el de desarrollo de una solución de software completa que ofrezca una base para futuras implementaciones de modelos de **Aprendizaje Automático** en materia de eficiencia energética. El código desarrollado está disponible en la cuenta de **GitHub** del autor del proyecto: **github.com/bruckel/Inergy.ML** y está desarrollado con la plataforma **.NET Core**.

Los productos desarrollados de los que consta el proyecto son una web **API** de gestión de datos de series temporales, una aplicación de migración de datos de series temporales, una aplicación de generación de modelos de **Aprendizaje Automático**, una aplicación para el consumo de modelos de **Aprendizaje Automático** y una demostración como aplicación web.

El proyecto permite la investigación y desarrollo en materia de aprendizaje automático, gestión de servicios en la nube y desarrollo de aplicaciones con la plataforma **.NET Core**.

Abstract (in English, 250 words or less):

The purpose of the project is the development of a **Machine Learning** system that allows predictions of consumption of electrical supplies. It also allows the detection of errors in the data of time series of consumption for later analysis and correction. All this taking as training data the historical time series of each of the supplies provided by the electrical companies.

The focus of the project is to develop a complete software solution that offers a basis for future implementations of **Machine Learning** models in energy efficiency. The code developed is available in the **GitHub** account of the author

of the project: **github.com/bruckel/Inergy.ML** and is developed with the **.NET** platform.

The developed products of which the project consists are a time series data management web **API**, a time series data migration application, a **Machine Learning** model generation application, an application for the consumption of **Machine Learning** models and a demonstration as a web application.

The project allows research and development in machine learning, cloud services management and application development with the **.NET Core** platform.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

1.1.1. Antecedentes

El autor del proyecto desarrolla su labor profesional en **Inergy**, una pequeña empresa especializada en aplicación y servicios de gestión y eficiencia energética. Entre los distintos servicios que ofrece **Inergy** destaca el software de gestión **Sistema de Información Energética, SIE**, que es utilizado por organizaciones privadas y administraciones públicas.

SIE trabaja con datos de facturación energética: electricidad, agua, gas, gasóleo, etc. estos datos de facturación están asociados a suministros eléctricos, que, en la mayoría de los casos, son instalaciones: colegios, polideportivos, cuadros de iluminación pública, etc.

Este software de gestión es una aplicación web desarrollada con la plataforma **.NET Framework.** De hecho, todos los productos tecnológicos que desarrolla la empresa se realizan con dicha plataforma.

La principal característica de **SIE** es que su funcionalidad se centra en el análisis retrospectivo de los datos de facturación. Es decir, se analiza el consumo, el importe y las posibles desviaciones que se hayan podido realizar. Tiene herramientas que permiten optimizar el gasto energético: desde la validación de los datos de facturación o la optimización de potencias eléctricas contratadas e instaladas.

Al iniciar el trabajo de desarrollo de la implementación en **SIE** del cálculo de precios indexados de electricidad se comenzó a obtener datos de consumo de carácter horario. Estos datos son ofrecidos por las comercializadoras energéticas en distintos formatos. Son importados vía CSV, XML o texto, dependiendo de cada una de las comercializadoras energéticas, dado que no existe un formato estándar.

Estos datos horarios se almacenan mediante una aplicación realizada a medida en .**NET Framework**, denominada **MDMS**, para el almacenamiento de datos de series temporales de manera organizada en una colección de base de datos relacionales **SQL Server**. Esta solución carece de algunas características en la **API** que limitan su funcionalidad: imposibilidad de eliminar serie temporales ya almacenadas, poca flexibilidad a la hora de añadir más campos, etc.

Además de almacenar en **MDMS** el consumo por hora de los suministros también se almacenan datos de dispositivos de monitorización, como sensores de temperatura, potenciómetros, etc. Estos datos de monitorización pueden tener una granularidad horaria de carácter diverso, aunque no son el objeto del presente proyecto, podrían tener un tratamiento similar en posteriores desarrollos.

1.1.2. Propósito

El propósito del proyecto es el desarrollo de un sistema de **Aprendizaje Automático** o **Machine Learning** (en lo sucesivo, ML) que permita realizar predicciones de consumo de suministros eléctricos presentes en la aplicación **SIE**, así como la detección de errores de los datos de las series temporales horarias de consumo para una posterior corrección.

Este sistema permitiría a **SIE** y otras aplicaciones de **Inergy** dotarse de gestión energética no sólo en retrospectiva sino también a modo de proyección futura. Por otro lado, se desea utilizar al máximo las nuevas herramientas y librerías que ofrece la plataforma **.NET Core**, realizando un ejercicio de investigación y desarrollo en el marco de la organización, de manera que se pueda implementar a corto plazo nuevas tecnologías en el desarrollo de software.

Concretamente, se desea desarrollar dos modelos ML:

A. Un modelo de predicción a futuro, por ejemplo, en los próximos dos meses, de consumo (especificados en *kWh*) por suministro eléctrico.

   Para su desarrollo se necesita del histórico de las series de datos de consumo por hora, así como otros parámetros que ayuden a mejorar la precisión del modelo: temperatura por hora, tipo de serie temporal, etc.

   Este modelo permite mostrar los datos de consumo que forman parte de la predicción mostrándolos en un gráfico. Esta posibilidad no existe actualmente en las aplicaciones de **Inergy**, en concreto en **SIE**, con lo que las funcionalidades ofrecidas a los clientes se verán incrementadas.

B. Otro modelo consistente en la detección de anomalías y errores en las series temporales de los datos de consumo (especificados en kWh) basándose en el histórico de los datos.

   Se trata de detectar huecos, picos o valles que supongan anomalías, normalmente, la ausencia de un dato para un *timestamp* concreto. Esta detección de anomalías y errores permitiría su corrección de manera preventiva, porque actualmente esta detección se hace al

usar estos datos para otros objetivos, con lo que se obtienen resultados no válidos.

Por ejemplo, en la aplicación **SIE**, la validación sobre precios indexados de tarifas eléctricas hace uso de los datos de series temporales, si se da el caso que estas son incompletas o corruptas esta validación genera resultados no válidos.

1.2 Objetivos del Trabajo

⇒ Desarrollar un sistema de **aprendizaje automático** funcional y consistente en dos modelos: predicción de consumo energético de suministros y detección de errores en series temporales horarias de consumo energético de suministros.

⇒ Utilizar herramientas de publicación en la nube para la puesta en marcha del sistema de **aprendizaje automático:** las aplicaciones planteadas en el proyecto y su almacenamiento de datos.

⇒ Implementar un sistema de almacenamiento no relacional para las series temporales de datos, así como una **API** que asegure su correcto mantenimiento.

⇒ Implementar modelos de predicción y entrenarlos con los datos ya existentes y automatizar el proceso para que el entrenamiento sea periódico a medida que los datos se ven incrementados.

⇒ Realizar un ejercicio de investigación y desarrollo en materia de nuevas tecnologías a adoptar por la organización

1.3 Enfoque y método seguido

El enfoque del proyecto es, principalmente, el de desarrollo de una solución de software completa que ofrezca una base para futuras implementaciones de modelos de **ML** en las aplicaciones de la organización.

La decisión se toma no sólo para enmarcar el trabajo en un ámbito más práctico, sino con el propósito de dar un paso más allá de la teoría y conceptos aprendidos en el máster: Inteligencia Artificial, Desarrollo Avanzado de Software, Herramientas de almacenamiento y Publicación Cloud, etc.

La organización se beneficia de un proyecto de este tipo, porque, como se especifica en el capítulo de Arquitectura, las tecnologías usadas son una prueba de concepto sobre una solución real, con el objetivo de ser

implementadas a corto plazo en el catálogo de productos de software de la organización

El código desarrollado está disponible en la cuenta de usuario de **GitHub** del autor del proyecto: github.com/bruckel/Inergy.ML Esta previsto que los archivos de código serán complementados con documentación sobre su utilización, ofreciendo series temporales para realizar pruebas en una máquina de desarrollo, tutorial de instalación, etc. De manera paralela se mantiene una copia del proyecto en el repositorio corporativo de **Inergy** en **Azure DevOps**, que es el que utiliza el equipo de desarrollo.

1.4 Planificación del Trabajo

El proyecto se descompone en distintos hitos, desglosados por cada producto a desarrollar y descomponiendo la tarea de desarrollo del producto en distintas tareas de menor tamaño.

Estos hitos son actualizados a medida que se presentan los distintos informes de seguimiento del proyecto, adaptando la estimación temporal a la realidad del desarrollo de cada una de las tareas.

1.4.1 Planificación inicial

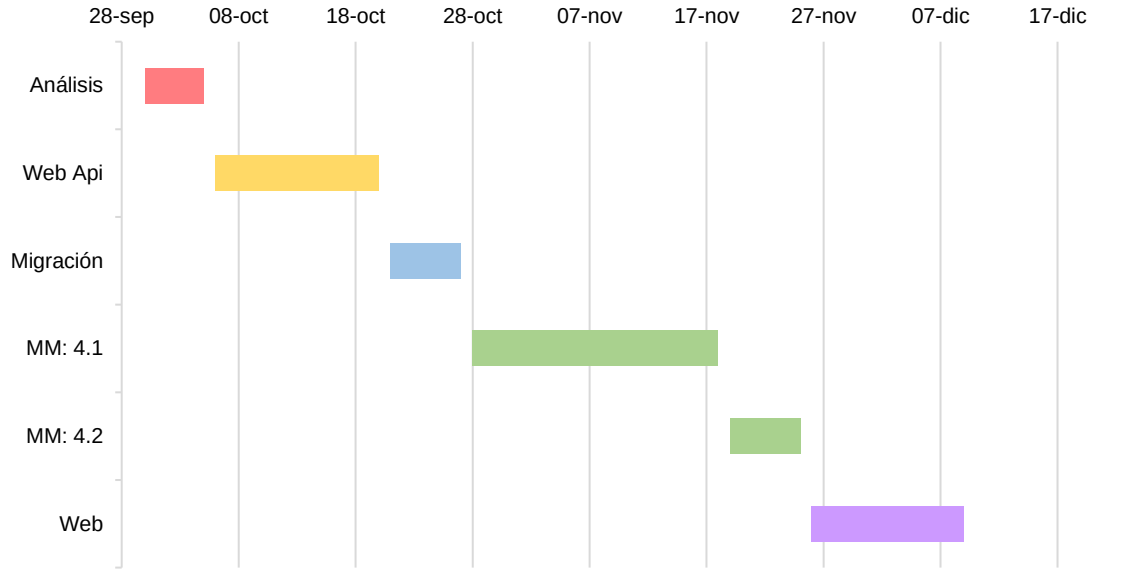
Esta es la planificación inicial de los hitos del proyecto:

a) **Análisis** y diseño de los productos. **06/10/2019**
b) **Web API** de gestión de datos series temporales. **21/10/2019**
c) Aplicación de **Migración** de datos **MDMS**. **28/10/2019**
d) Módulo de **Aprendizaje Automático**.

- **i.** **MM: 4.1.** Generador de Modelos. **18/11/2019**
- **ii.** **MM: 4.2.** Web Api para el consumo de modelos. **25/11/2019**
- **iii.** **Web** de demostración. **09/12/2019**

El **Diagrama de Gantt** correspondiente a la planificación inicial del proyecto corresponde con la **figura 1.4.1.**

Figura 1.4.1 *Diagrama de Gantt de la planificación inicial.*



1.4.2 Planificación del primer informe de seguimiento

Durante el periodo de este informe los hitos se descomponen en tareas de más pequeñas, de esta manera se puede ajustar y corregir las desviaciones en los tiempos de desarrollo. El estado de ejecución de las tareas para este informe de seguimiento es:

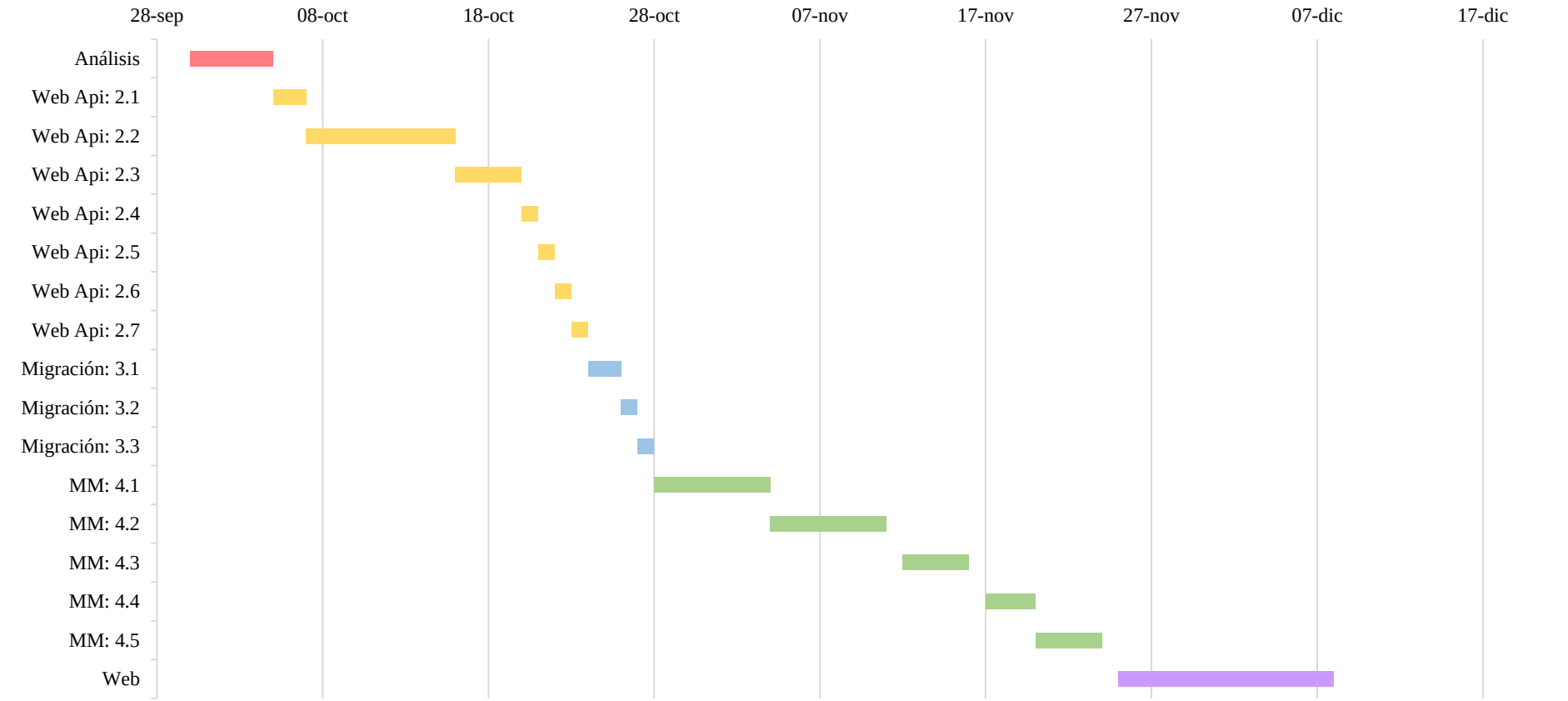
- Finalizado
- En progreso.
- Pendiente.

El código asignado ejerce de identificador de la tarea en el **Diagrama de Gantt** adjuntado la **tabla 1.4.2** y la **figura 1.4.2.**

Tabla 1.4.2 *Planificación de tareas del primer informe de seguimiento.*

Código.	Descripción.	Finalización.
Análisis.	Análisis y diseño de la solución.	6/10/2019
Web Api para la consulta, creación, modificación y eliminación de datos de series temporales.		
Web Api: 2.1.	Modificación y publicación como paquete de la Librería *Inergy.Tools*.	8/10/2019
Web Api: 2.2.	Desarrollo del código y configuración de repositorios de la lógica de datos y de negocio de la solución.	17/10/2019
Web Api: 2.3.	Implementación de una aplicación de consola para realizar pruebas, así como desarrollo de la Web Api.	21/10/2019
Web Api: 2.4.	Aplicar solución para la consulta de datos horarios por zona horaria y transformación de los datos a UTC.	22/10/2019
Web Api: 2.5.	Configuración según la especificación *Open API* de la Web Api.	23/10/2019
Web Api: 2.6.	Configuración *Azure Blob Storage* para los archivos de registro de la solución.	24/10/2019
Web Api: 2.7.	Instalar la Web Api en Azure Web App.	25/10/2019
Aplicación de Migración de datos ya existentes, desde *MDMS* a *Cosmos DB* utilizando la Web Api.		
Migración: 3.1.	Desarrollo del código de la aplicación, reutilizando parte de código ya utilizado actualmente para el consumo de datos de series temporales de MDMS.	27/10/2019
Migración: 3.2.	Configuración de *Azure Functions* para la ejecución diaria de la aplicación de migración.	28/10/2019
Migración: 3.3.	Realización de pruebas con conjuntos de datos limitados, ampliando dichos conjuntos en diferentes pruebas y terminado con la puesta en producción.	29/10/2019
Módulo de **Aprendizaje Automático**.		
MM: 4.1.	Modelo de predicción de consumo (kWh) para suministros eléctricos.	5/11/2019
MM: 4.2.	Modelo de detección de errores de los datos de series temporales.	12/11/2019
MM: 4.3.	Desarrollo de la aplicación de entrenamiento con el histórico de datos de MongoDB obtenidos mediante la migración previa.	17/11/2019
MM: 4.4.	Modificar la API web para ofrecer el consumo de los modelos por parte de terceras aplicaciones.	21/11/2019
MM: 4.5.	Implementación de pruebas.	25/11/2019
Web.	Web de demostración de utilización de los modelos.	9/12/2019

Figura 1.4.2 *Diagrama de Gantt del primer informe de seguimiento.*



1.4.3 Planificación del segundo informe de seguimiento

De manera similar al primer informe los hitos se descomponen en tareas más pequeñas. También se modifican tareas en función de cambios en los productos del proyecto, modificaciones, problemas de desarrollo, etc. El estado de ejecución de las tareas para este informe de seguimiento es:

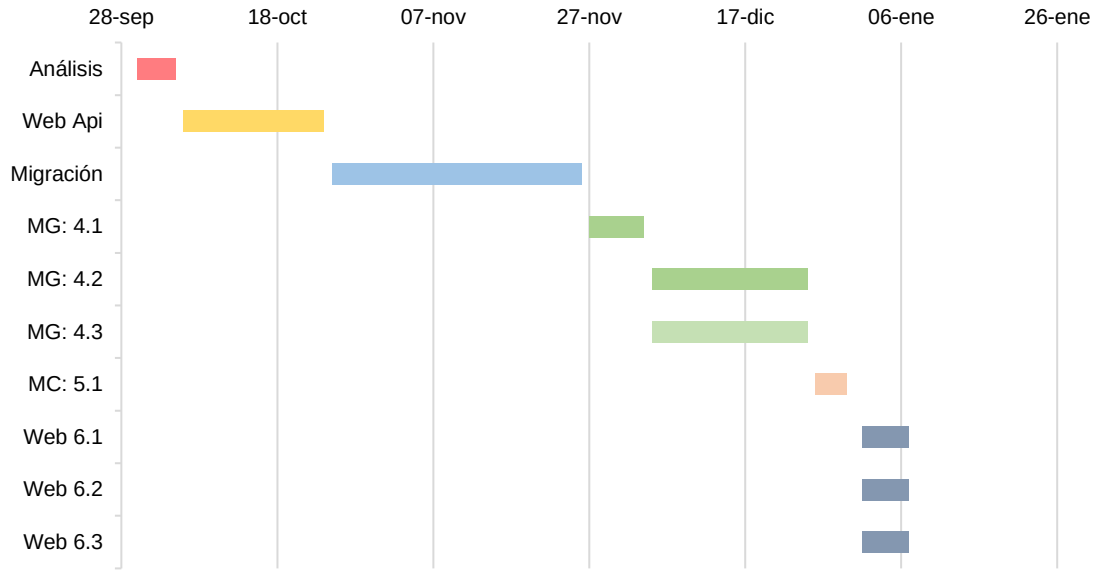
- Finalizado.
- En progreso.
- Pendiente.

El código asignado ejerce de identificador de la tarea en el **Diagrama de Gantt** adjuntado la **tabla 1.4.3** y la **figura 1.4.3**.

Tabla 1.4.3 *Planificación de tareas del segundo informe de seguimiento.*

Código.	Descripción.	Finalización.
Análisis.	Análisis y diseño de la solución.	6/10/2019
Web Api para la consulta, creación, modificación y eliminación de datos de series temporales.		
Web Api.	Desarrollo de la Web Api y publicación en Azure Web App.	25/10/2019
Aplicación de Migración de datos ya existentes, desde *MDMS* a Mongo *DB* utilizando la Web Api.		
Migración: 3.1.	Desarrollo de la aplicación de migración y carga inicial en preproducción.	27/11/2019
Migración: 3.2.	Configuración de *Azure Web Jobs* para la ejecución diaria de la aplicación de migración.	01/12/2019
Aplicación de generación de modelos **ML**.		
MM: 4.1.	Modelo de predicción de consumo (kWh) para suministros eléctricos. Algoritmo *ForecastBySsa.*	5/12/2019
MM: 4.2.	Modelo de predicción de consumo (kWh) para suministros eléctricos. Algoritmo *FastTreeTweedie*.	26/12/2019
MM: 4.3.	Modelo de detección de errores de los datos de series temporales.	26/12/2019
Aplicación de consumo de modelos **ML**.		
MM: 5.1.	Desarrollo de la aplicación para el consumo de modelos ML:	31/12/2019
Web de demostración de utilización de los modelos.		
Web: 6.1.	Gráficas y tabla para el Modelo de predicción de consumo (kWh) para suministros eléctricos. Algoritmo *ForecastBySsa.*	8/01/2020
Web: 6.2.	Gráficas y tabla para el Modelo de predicción de consumo (kWh) para suministros eléctricos. Algoritmo *FastTreeTweedie*.	8/01/2020
Web: 6.3.	Gráficas y tabla para el Modelo de detección de errores de los datos de series temporales	8/01/2020

Figura 1.4.3 *Diagrama de Gantt del segundo informe de seguimiento.*



1.4.4 Estado de la planificación al finalizar el proyecto

A fecha de finalización del proyecto se han quedado algunas tareas pendientes de ejecución, tal como se refleja en la **tabla 1.4.4**. El estado de ejecución de las tareas sigue la misma escala que en los informes de seguimiento previos:

- Finalizado.
- En progreso.
- Pendiente.

Tabla 1.4.4 *Planificación de tareas al finalizar el proyecto.*

Código.	Descripción.	Finalización.
Análisis.	Análisis y diseño de la solución.	6/10/2019
Web Api para la consulta, creación, modificación y eliminación de datos de series temporales.		
Web Api.	Desarrollo de la Web Api y publicación en Azure Web App.	25/10/2019
Aplicación de Migración de datos ya existentes, desde *MDMS* a Mongo *DB* utilizando la Web Api.		
Migración: 3.1.	Desarrollo de la aplicación de migración y carga inicial en preproducción.	27/11/2019
Migración: 3.2.	Configuración de *Azure Web Jobs* para la ejecución diaria de la aplicación de migración.	01/12/2019
Aplicación de generación de modelos **ML**.		
MM: 4.1.	Modelo de predicción de consumo (kWh) para suministros eléctricos. Algoritmo *ForecastBySsa.*	5/12/2019
MM: 4.2.	Modelo de predicción de consumo (kWh) para suministros eléctricos. Algoritmo *FastTreeTweedie*.	
MM: 4.3.	Modelo de detección de errores de los datos de series temporales.	26/12/2019
Aplicación de consumo de modelos **ML**.		
MM: 5.1.	Desarrollo de la aplicación para el consumo de modelos ML:	
Web de demostración de utilización de los modelos.		
Web: 6.1.	Gráficas y tabla para el Modelo de predicción de consumo (kWh) para suministros eléctricos. Algoritmo *ForecastBySsa.*	8/01/2020
Web: 6.2.	Gráficas y tabla para el Modelo de predicción de consumo (kWh) para suministros eléctricos. Algoritmo *FastTreeTweedie*.	
Web: 6.3.	Gráficas y tabla para el Modelo de detección de errores de los datos de series temporales	8/01/2020

1.5 Breve sumario de productos obtenidos

Para el desarrollo del sistema de **aprendizaje automático** planteado es necesario que el proyecto tenga los siguientes productos:

⇒ **API Web de gestión de datos de series temporales.** Desarrollada como **API Web** ejerce de servicio que interactúa con el sistema de almacenamiento de datos no relacional **Mongo DB** donde se guardan las series temporales. Permite la consulta, creación, edición y eliminación de los datos. **URL** de la versión de preproducción: **inergyapi.azurewebsites.net/swagger/index.html**

⇒ **Aplicación de migración de datos de series temporales.** Necesario para la migración entre el sistema actual de almacenamiento de las series temporales, denominado **MDMS,** a un nuevo sistema de almacenamiento de datos no relacional.

Para ello se hace uso de **Mongo DB** como solución para el almacenamiento de los datos y **Azure Web Jobs** para ejecutar la aplicación periódicamente. Esta aplicación migra todo el histórico de datos y, posteriormente, realiza una ejecución periódica actualizando la información mientras **MDMS** siga en producción.

⇒ **Módulo de generación de modelos ML.** Es la propia aplicación objeto del proyecto y la parte más importante del mismo. La definición de los modelos de aprendizaje, su entrenamiento y su consumo se realiza con la librería **ML .NET**.

Esta aplicación es la responsable de la definición de los modelos, entrenamiento y almacenamiento de los dos modelos previstos en el proyecto: la **predicción de consumo eléctrico** en base a las series temporales de consumo previos y otros parámetros (como, por ejemplo, información de temperatura) y la **detección y corrección de errores** en esas mismas series temporales.

La funcionalidad de esta aplicación debe ser escalable, porque en el futuro, la necesidad de modelos se puede ver incrementada con el desarrollo de nuevos modelos que den respuesta a las necesidades de la organización.

⇒ **Aplicación para el consumo de modelos ML.** Aplicación desplegada en **Azure Functions** que, mediante una petición HTTP devuelve los valores de predicción de los modelos generados y guardados anteriormente. Esta aplicación es necesaria para el consumo de los modelos desde distintas aplicaciones de la organización.

⇒ **Una demo como aplicación web.** Una aplicación web desarrollada con **Blazor** y componentes **Telerik** a modo de demostración del funcionamiento de los modelos. Debe ser sencilla y permitir, como parámetros, la elección de un determinado suministro eléctrico y un periodo temporal. En función de estos datos de entrada se muestra por pantalla una serie de gráficos y datos que muestran la validez del modelo y sus resultados. Se puede consultar la **URL** de la versión de preproducción: **inergymldemo.azurewebsites.net**

1.6 Breve descripción de los otros capítulos de la memoria

Los siguientes capítulos de este documento se centran en dos aspectos.

⇒ Explicar la arquitectura de cada uno de los productos del proyecto mediante el uso de diagramas que especifican su diseño funcional y una descripción de cada uno de los procesos que intervienen en las distintas aplicaciones.

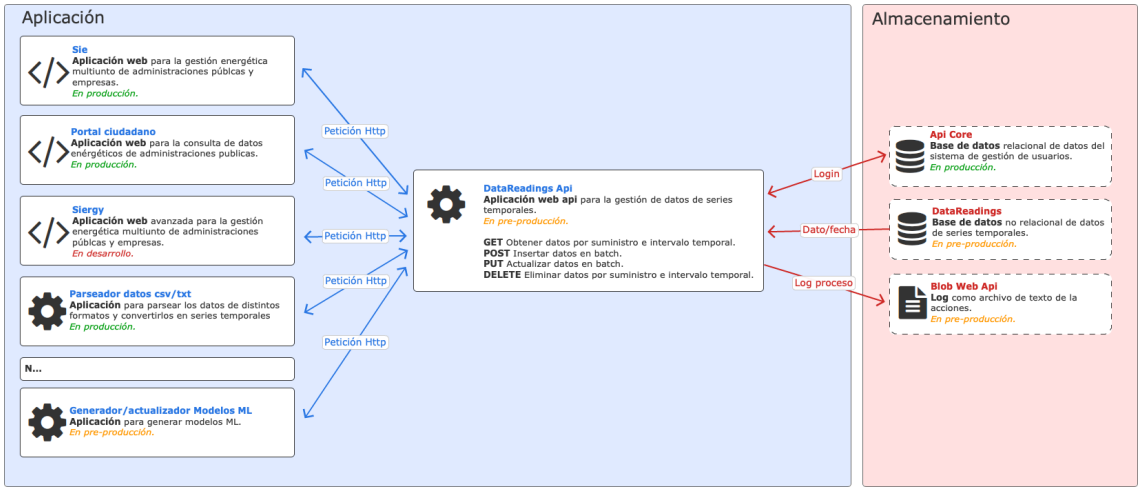
⇒ Exponer las tecnologías con las que se han desarrollado los productos del proyecto. También se utilizan diagramas para su comprensión, además de las referencias a las páginas webs correspondientes.

2. Arquitectura

El proyecto necesita del desarrollo de una serie de aplicaciones y sistemas de almacenamiento (bases de datos, archivos de texto, etc.) para la consecución de los objetivos.

2.1 API de datos de series temporales.

Figura 2.1. *Diagrama que muestra la integración de la* ***API Web*** *con el resto de las aplicaciones y su capa de datos.*



La **API web** es la aplicación que interactúa con el sistema de base de datos no relacional **Mongo DB** donde se almacenan los datos de las series temporales. Para ello se ha creado código extensible, escalable y limpio que no sólo sirve para la **API Web**, sino que ejerce de base para el resto de las aplicaciones a desarrollar.

La aplicación es una **API REST** con lógica de negocio necesaria para la consulta, creación, modificación y eliminación de los datos de series temporales:

- ⇒ **GET** Obtener datos con el código de suministro e intervalo temporal como parámetros.
- ⇒ **POST** Insertar datos en lote con la colección de series temporales a insertar como parámetro.
- ⇒ **PUT** Actualizar datos en lote con la colección de series temporales a actualizar como parámetros.

⇒ **DELETE** Eliminar datos con el código de suministro e intervalo temporal.

La **Web API** permite la sobreescritura de datos de series temporales en el caso de que ya estén almacenados. Esta funcionalidad es necesaria porque es habitual que existan datos facilitados que sean erróneos y que, a través de aplicaciones de terceros, se quieran corregir.

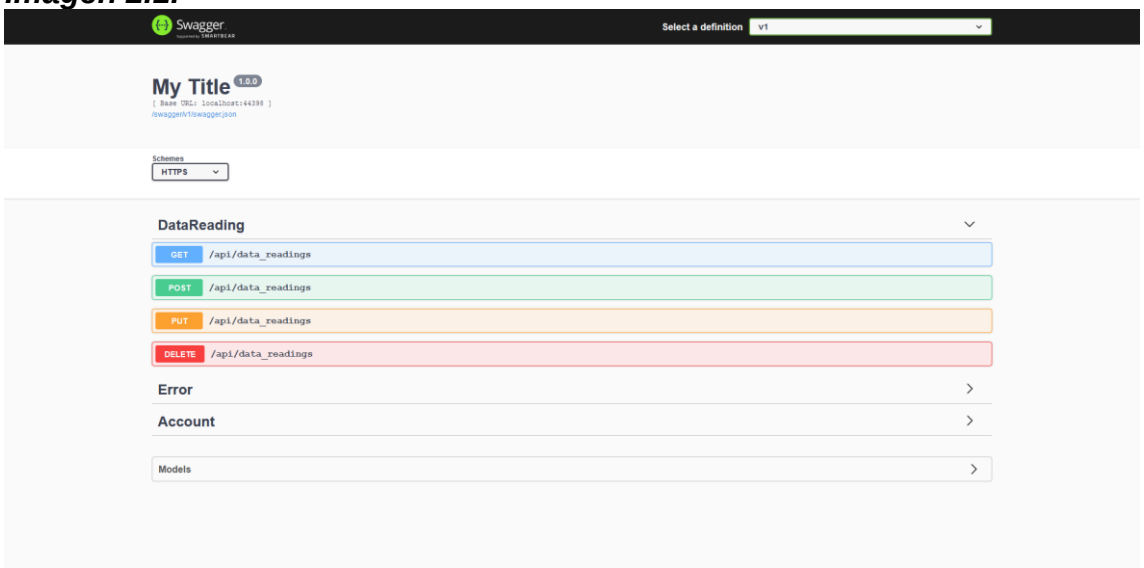
La documentación de la **Web API** esta configurada con la especificación **Open API**. Como todas las aplicaciones desarrolladas en el proyecto se implementa un sistema de registro de las peticiones realizadas a la aplicación. Existe un archivo de registro diario con los datos creados, modificados o eliminados, la hora de ejecución de estas acciones y las excepciones registradas.

La aplicación tiene implementada un sistema de control del *timestamp* de las series temporales para su almacenamiento en formato **UTC**. Actualmente, el sistema **MDMS** guarda esta propiedad con el *timezone* por defecto de la máquina servidor (Europa/Madrid).

El objetivo es permitir la internacionalización y globalización del proyecto permitiendo la creación y edición de series de datos temporales con un *timezone* especificado como parámetro. Para ello, a cada ítem de la serie temporal se le añade una propiedad *timeoffset*, que indica la desviación en nº de horas del timestamp en función del *timezone*.

Por ultimo, la **Web API** implementa el sistema de autorización y autenticación **Identity** de **.NET.** De esta manera sólo se permite el uso de la aplicación a aquellos usuarios registrados y, en función del rol asignado, con permiso para obtener, crear, modificar o eliminar datos de series temporales.

Imagen 2.1.



2.2 Aplicación de migración de datos.

La finalidad de esta aplicación es la migración de los datos del sistema actual de almacenamiento de series temporales, denominado **MDMS,** a un nuevo sistema de almacenamiento de datos no relacional.

MDMS es una aplicación desarrollada con la plataforma **.NET Framework** encargada de gestionar datos de series temporales con un identificador, fecha y valor. Para ello gestiona, mediante una **API**, un número variable de bases de datos **SQL Server** con un número variable de tablas mapeadas para almacenar los datos clave-valor. Sin entrar en más detalle, es un sistema que da respuesta a una necesidad de almacenamiento no relacional, aunque está desarrollado en un momento en que no existían este tipo de bases de datos.

MDMS tiene una serie de inconvenientes que hacen necesaria su migración hacia un sistema de almacenamiento no relacional:

⇒ La **API** no tiene implementada un método para la eliminación de registros de series temporales para un determinado identificador e intervalo temporal.

⇒ Sólo se puede almacenar un valor para un determinado identificador y *timestamp*. A menudo es necesario añadir más parámetros al *timestamp* en función de las características de la serie temporal.

⇒ **MDMS** se aloja en un servidor virtual dedicado, consumiendo recursos que se pueden optimizar si el sistema de almacenamiento es no relacional.

El sistema de base de datos no relacional seleccionado es **Mongo DB**, con el cual la organización ya tiene experiencia en el desarrollo de aplicaciones.

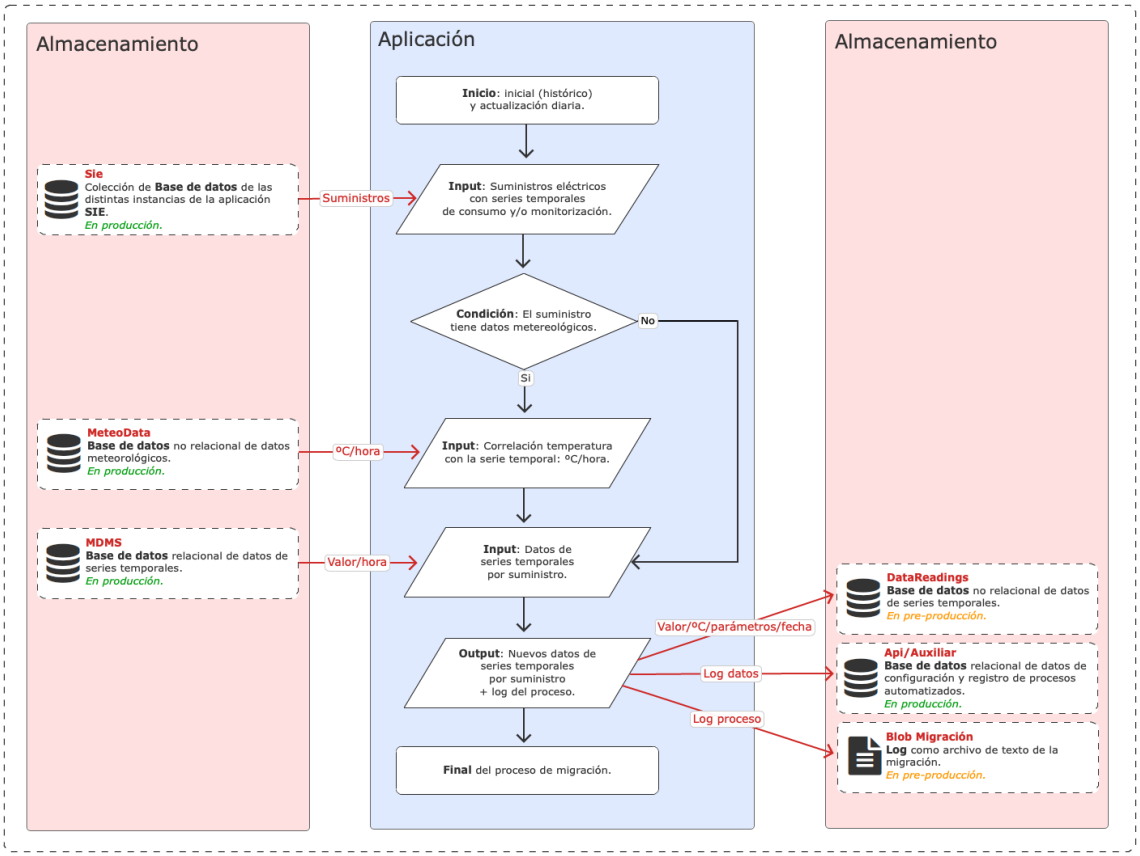
La gestión de las series temporales, es decir, su consulta, creación, modificación y eliminación se delega en la aplicación **Api Web**, ya detallada en el capítulo 2.1.

Mongo DB se basa en un sistema de "**colecciones**" donde cada ítem puede tener una estructura diferenciada de la anterior. Esto permite almacenar ítems mucho más complejos que un simple registro clave-valor. De hecho, el planteamiento es tener dos tipos de series temporales al realizar la migración, ambos se almacenan en la misma colección, mejorando la facilidad de gestión de los datos:

⇒ Series temporales horarias de consumo (kWh) de los suministros eléctricos

⇒ Series temporales de granularidad temporal y valor variable de los dispositivos de monitorización asociados a suministros de todo tipo: eléctricos, de gas, de agua, fotovoltaica, etc.

Figura 2.2. *Flujo de la aplicación de migración de datos de MDMS y sistemas de almacenamiento que intervienen en el flujo.*



La **figura 2.1** representa el flujo de la aplicación de migración de datos de series temporales:

1. Se obtienen todos los identificadores de suministros que disponen de datos de series temporales de todos los tipos. Estos datos se obtienen de las aplicaciones en producción de la organización.

2. La organización dispone de un repositorio de datos meteorológicos obtenidos mediante la **API** de *Dark Sky*. De esta manera, es posible asociar a los datos de series temporales horarias la temperatura a una determinada hora. Esta vinculación se realiza mediante la geolocalización cruzada del suministro (longitud y latitud) y la estación meteorológica con datos más cercana.

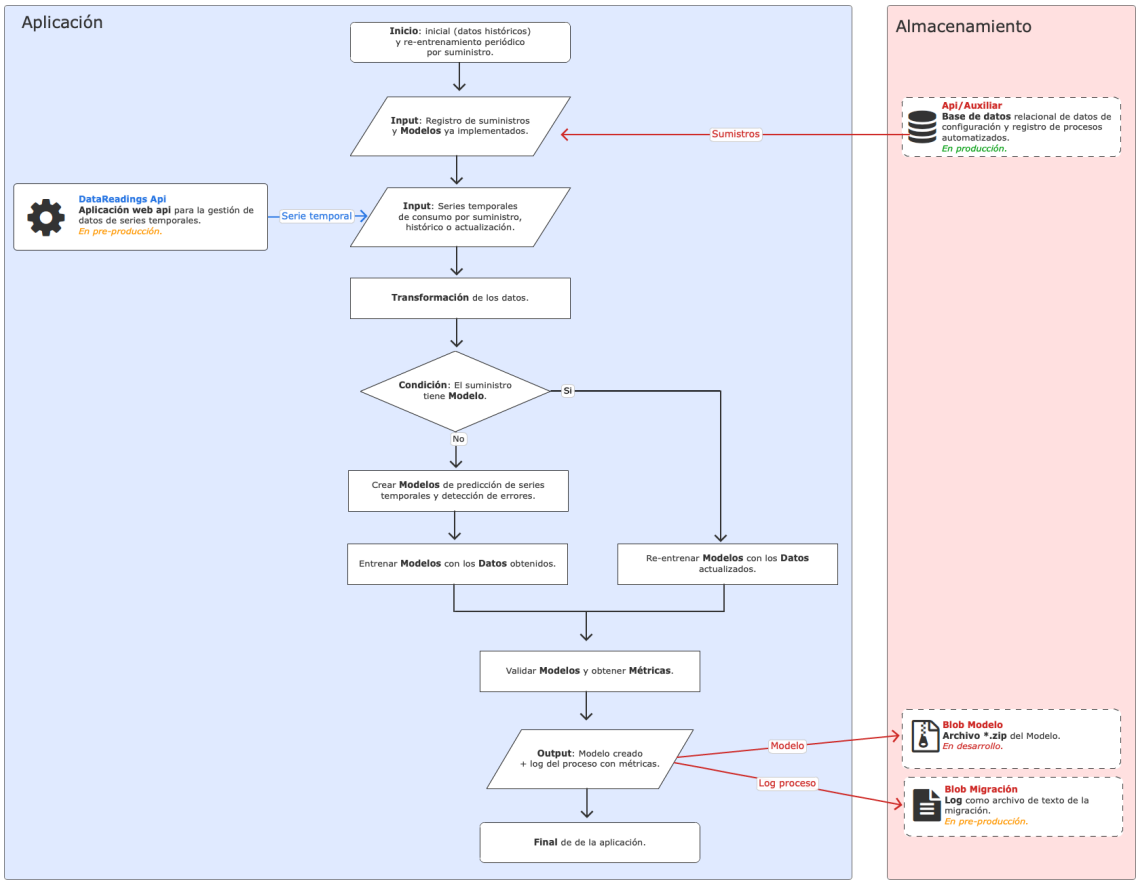
3. Se obtienen los datos de **MDMS** para cada suministro, se serializan de manera estándar y se incorpora la información complementaria necesaria: tipo de serie temporal, temperatura, etc.

4. Se procede a insertar los datos en el sistema de almacenamiento no relacional mediante la **Web API**. También se crea un registro del proceso de migración con información diaria de relevancia y un histórico de datos migrados.

Esta aplicación se debe ejecutar de manera inicial importando todo el histórico de series temporales de todos los suministros con datos de consumo horario y datos de monitorización, y, posteriormente, periódicamente, actualizando los datos de series temporales mientras **MDMS** siga en producción. A medio plazo la recomendación es dar de baja **MDMS** para utilizar en exclusiva el nuevo sistema.

2.3 Aplicación de generación de modelos de ML.

Figura 2.3. *Flujo de la aplicación de generación de modelos ML y sistemas de almacenamiento que intervienen en el flujo.*



Es la aplicación más importante del proyecto, ya que trata de generar los modelos de **aprendizaje automático** con los datos en poder de la organización y gestionados con las aplicaciones anteriores: datos migrados desde su almacenamiento actual y posteriormente tratados con la **Web API** para su gestión.

En esta aplicación se definen los modelos de aprendizaje, se realiza el entrenamiento y, por último, la evaluación mediante el cálculo de distintas métricas. Por último, se almacena el modelo para su consumo por terceras aplicaciones. Para el desarrollo de estos modelos se utiliza la librería **ML .NET**.

La funcionalidad debe ser escalable, porque en el futuro la necesidad de modelos se puede ver incrementada.

La **figura 2.3.** representa el flujo de la aplicación de modelos de **ML**:

1. Se obtienen todos los identificadores de suministros eléctricos, denominados **CUPS**, que disponen de datos de series temporales de consumo horario. Estos datos se obtienen del registro de datos de la aplicación de migración. Así mismo, se identifica que suministros tienen un modelo válido almacenado previamente.

2. Se obtienen las series temporales de consumo horario por cada suministro eléctrico. Si el modelo no está almacenado se obtiene todo el histórico, pero en el caso de un modelo ya guardado se obtiene los datos necesarios para realizar un reentrenamiento del modelo.

3. Se procede a la transformación de los datos obtenidos. En el caso de que el modelo ya exista se reentrena el modelo incorporando los datos obtenidos y que son los que complementan el modelo para su nuevo entrenamiento.

4. Para los nuevos modelos se define cada uno de los algoritmos planteados para los casos de uso del proyecto:

    ⇒ Predicción de consumo horario. Para este caso de uso se implementan dos algoritmos, lo que permite realizar una comparación del nivel de precisión haciendo uso de las métricas resultantes.

    a) Series temporales. **Algoritmo ForecastBySsa.**

    b) Regresión lineal. **Algoritmo FastTreeTweedie.**

    ⇒ Detección de errores en el consumo horario. Para este caso de uso se implementa el algoritmo **DetectSpikeBySsa**.

5. Una vez definidos los algoritmos se entrenan los nuevos modelos con los datos obtenidos previamente.

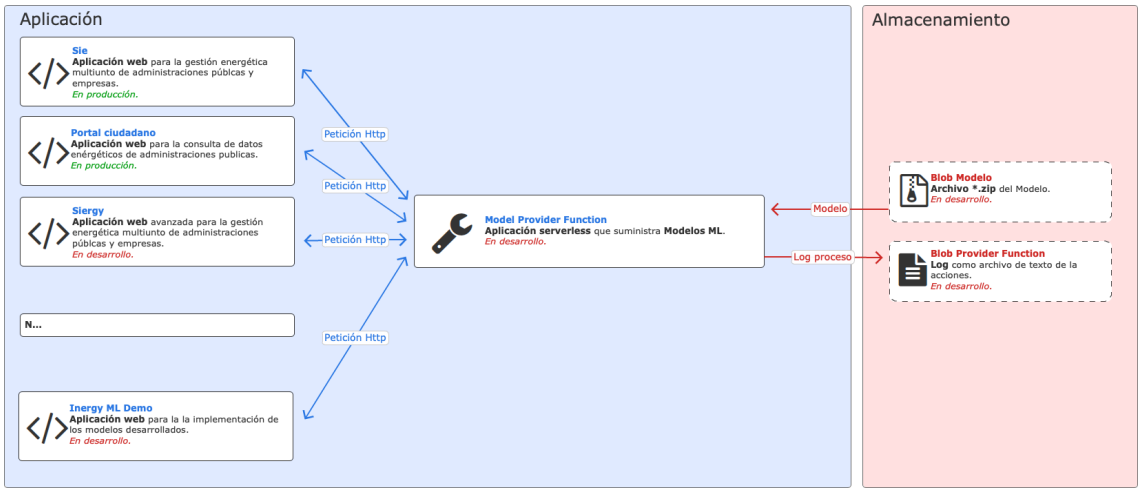
6. Tanto los nuevos modelos como aquellos que son objeto e reentrenamiento pasan por una fase de evaluación de precisión. De esta manera se obtienen una serie de métricas que se almacenan en el registro diario del proceso de la aplicación.

7. El registro diario del proceso almacena en un archivo de texto las métricas y aquellas excepciones en la implementación de los modelos. El modelo creado, entrenado y evaluado se guarda para su posterior consumo.

Esta aplicación se debe ejecutar de manera inicial, creando tantos modelos como suministros con datos de series temporales de consumo, y, posteriormente, se ejecuta de manera periódica, actualizando los modelos creados anteriormente y creando nuevos modelos en el caso de nuevos suministros con datos de series temporales.

2.4 Aplicación de consumo de modelos de ML.

Figura 2.4. *Diagrama que muestra la integración de la aplicación de consumo de modelos y el almacenamiento relacionado.*



Se trata de una aplicación **serverless** que tiene como objeto ofrecer los resultados del modelo como resultado de una petición **HTTP**. De esta manera los modelos están disponibles para todo el ecosistema de aplicaciones de la organización.

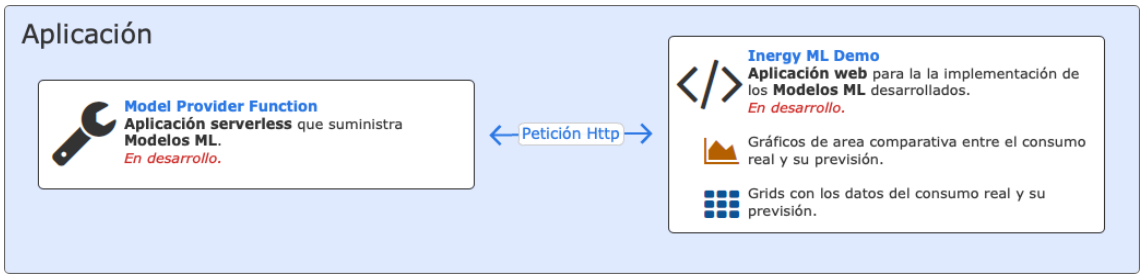
El modelo se guarda por la aplicación de generación de modelos mediante el servicio de almacenamiento **Azure Blob Storage**.

A través de una petición **HTTP POST** y mediante el envío de los parámetros de consulta para la utilización del modelo la aplicación se retornan los resultados serializadas del modelo.

Al igual que el resto de las aplicaciones del proyecto se guarda un archivo de texto a modo de registro diario de las consultas realizadas a la aplicación: el modelo solicitado, parámetros y excepciones registradas, en el caso de que las hubiere.

2.5 Aplicación web de demostración.

Figura 2.5. *Diagrama que muestra las características de la* ***Web Demo*** *y como se comunica con la aplicación de consumo de modelos.*

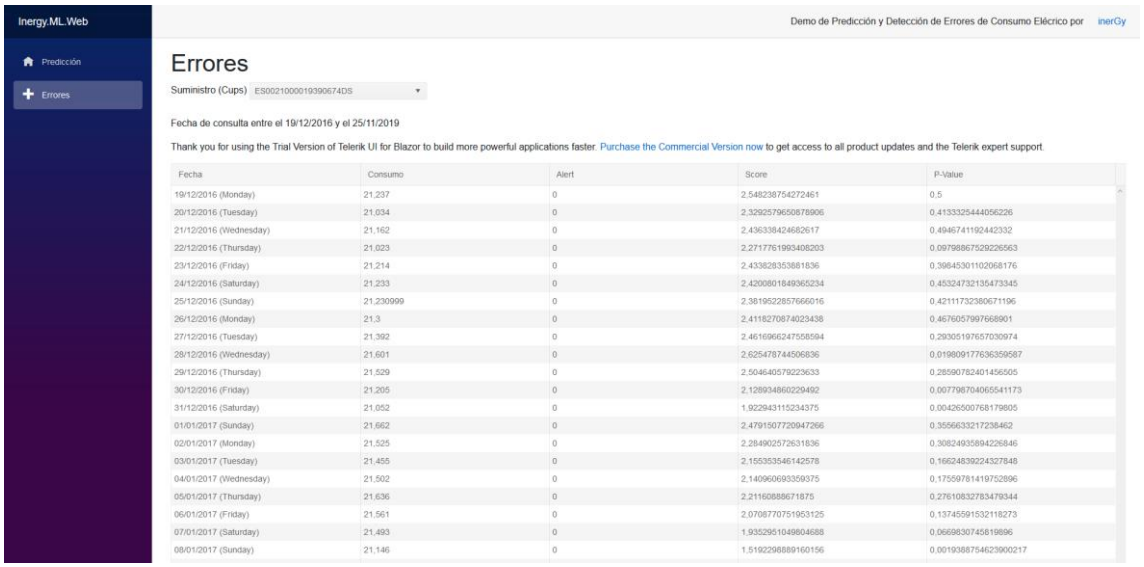


Para realizar una demostración del consumo de los modelos generados y a modo de colofón del desarrollo del proyecto se plantea una pequeña aplicación web que muestre los resultados.

Debe ser sencilla y permitir, como parámetros, la elección de un determinado identificador de suministro (**CUPS**). En función de estos datos de entrada se muestra por pantalla una serie de gráficos y datos que muestran la validez del modelo y sus resultados.

Imagen 2.5.1 *Gráfica comparativa de la predicción de datos de consumo respecto a los datos reales.*

Imagen 2.5.2 *Tabla que muestra los datos agrupados diariamente indicando aquellos días con anomalías detectadas.*



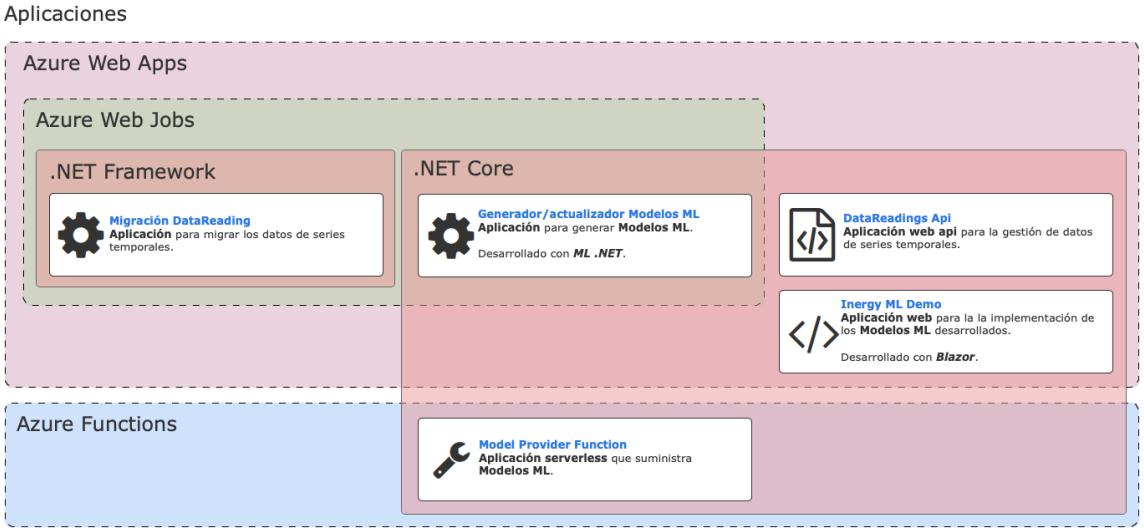
Fecha	Consumo	Alert	Score	P-Value
19/12/2016 (Monday)	21,237	0	2,548238754272461	0,5
20/12/2016 (Tuesday)	21,034	0	2,3292579650878906	0,4133325444056226
21/12/2016 (Wednesday)	21,162	0	2,436338424682617	0,4946741192442332
22/12/2016 (Thursday)	21,023	0	2,2717761993408203	0,09798867529226563
23/12/2016 (Friday)	21,214	0	2,433828353881836	0,3964530110206817 6
24/12/2016 (Saturday)	21,233	0	2,4200801849365234	0,45324732135473345
25/12/2016 (Sunday)	21,230999	0	2,3819522857666016	0,42111732380671196
26/12/2016 (Monday)	21,3	0	2,4118270874023438	0,4676057997668901
27/12/2016 (Tuesday)	21,392	0	2,4616966247558594	0,29305197657030974
28/12/2016 (Wednesday)	21,601	0	2,625478744506836	0,019809177636359587
29/12/2016 (Thursday)	21,529	0	2,504640579223633	0,28590782401456505
30/12/2016 (Friday)	21,205	0	2,128934860229492	0,007798704065541173
31/12/2016 (Saturday)	21,052	0	1,922943115234375	0,00426500768179805
01/01/2017 (Sunday)	21,662	0	2,4791507720947266	0,3556633217238462
02/01/2017 (Monday)	21,525	0	2,284902572631836	0,30824935894226846
03/01/2017 (Tuesday)	21,455	0	2,155353546142578	0,16624839224327848
04/01/2017 (Wednesday)	21,502	0	2,140960693359375	0,17559781419752896
05/01/2017 (Thursday)	21,636	0	2,21160888671875	0,27610832783479344
06/01/2017 (Friday)	21,561	0	2,0708770751953125	0,13745591532118273
07/01/2017 (Saturday)	21,493	0	1,9352951049804688	0,0669830745819896
08/01/2017 (Sunday)	21,146	0	1,5192298889160156	0,0019388754623900217

3. Tecnologías

3.1. Tecnologías de desarrollo de aplicaciones.

La **Figura 3.1** es un resumen de la propuesta de tecnologías para el desarrollo del proyecto. Se pueden diferenciar dos aspectos: los entornos de desarrollo de las aplicaciones y los servicios de los que se hace uso para su puesta en marcha.

Figura 3.1. *Aplicaciones del proyecto y servicio de Azure en los que se despliega, así como Framework .NET correspondiente.*



Para el desarrollo de las aplicaciones se opta por el entorno de trabajo **.NET** de **Microsoft**, porque es el usado para el desarrollado de la mayoría de las soluciones de software de **Inergy.** Para este proyecto se utilizan distintas versiones del entorno, debido a distintas necesidades y etapas de desarrollo de las aplicaciones.

La aplicación de **Migración de Datos de Series Temporales** es una aplicación de consola desarrollada con la versión 4.6.1 de **.NET Framework**, porque aprovecha librerías y código ya desarrollado, como, por ejemplo, la **API MDMS**, desarrollada con esa versión.

En cambio, para el resto de las aplicaciones del proyecto se ha optado por la versión 3.0 (lanzado en octubre de 2019) de **.NET Core**. Esta versión es la apuesta de Microsoft por el software libre y multiplataforma, ya que las aplicaciones desarrolladas con **.NET Core** pueden ser ejecutados en máquinas **Windows, Linux** o **macOS**.

El **Generador/Actualizador de Modelos ML** es una aplicación de consola que desarrolla los modelos con la API de la librería **ML .NET** de en su versión 1.4.0. el objetivo de Microsoft es crear una plataforma de aprendizaje automático que pueda competir con las de otras plataformas, como **R** o **Python**.

La **API REST** para a la **Gestión de Datos Temporales** y la **Web Demo** son aplicaciones web. La **Web Demo** se desarrolla con **Blazor**, un nuevo paradigma para el desarrollo de aplicaciones web asíncronas sin necesidad de utilizar **Javascript**. Para ello se usan los componentes **Telerik**, los mismos que en otros proyectos de la organización.

En lo relativo a los servicios en la nube de **Azure** se ha optado por **Azure Web Jobs** para la ejecución programada de la aplicación de **Migración de Datos de Series Temporales** y también del **Generador/Actualizador de Modelos ML.**

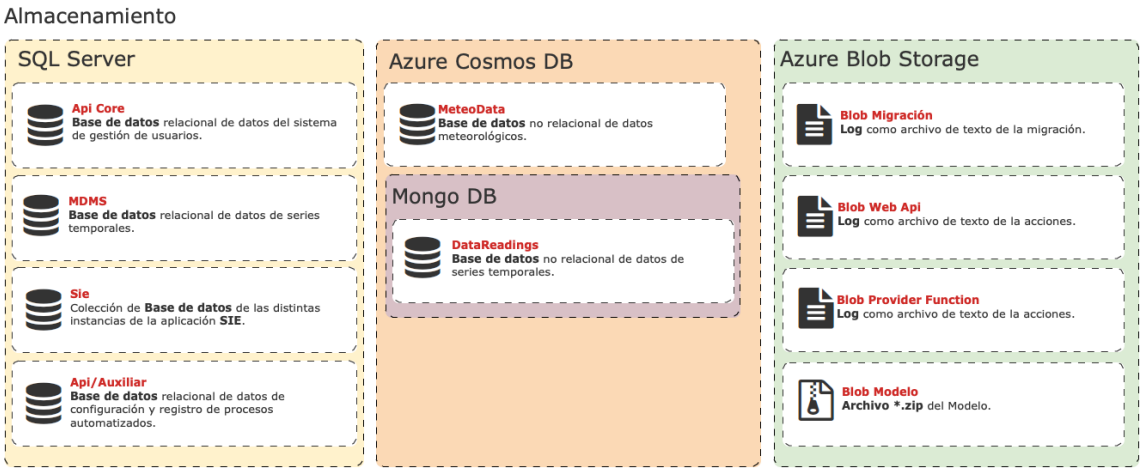
La **Web API** y la **Web Demo** se despliegan en **Azure Web Apps** y el proveedor de modelos es una aplicación implementada en **Azure Functions.**

El lenguaje utilizado para el desarrollo en el entorno **.NET** es **C#**.

3.2. Tecnologías de almacenamiento de datos.

La Figura 3.2 es un resumen de la propuesta de tecnologías para el almacenamiento de los datos necesario para el desarrollo del proyecto. Se pueden diferenciar tres sistemas de almacenamiento distintos que se ajustan a cada una de las aplicaciones del proyecto.

Figura 3.2. *Tipologías de almacenamiento del proyecto, así como bases de datos y tipologías de archivos almacenados.*



Para el almacenamiento de la gestión de usuarios (**Identity .NET**) de la **Web API** se utiliza una base de datos relacional **SQL Server**. Bajo la misma arquitectura se encuentran los datos de series temporales previos a su migración, el sistema denominado **MDMS**.

La aplicación **SIE** también almacena sus datos en **SQL Server**, así como el registro que utiliza el programa de **Migración de Series Temporales**.

Para el almacenamiento de los datos de series temporales se hace uso de **Mongo DB** implementado en el servicio **Azure Cosmos DB**, que permite utilizar la API de gestión de datos no relacionales.

La **API** de gestión de datos no relacionales utilizada es **Mongo DB** dado que ya ha sido utilizado en otros proyectos de la organización, aunque nunca utilizando el servicio **Azure Cosmos DB**, sino con un servidor de **Mongo DB** instalado en las máquinas de desarrollo y producción.

Al elegir **Mongo DB** la curva de aprendizaje es menor y se podría afrontar una posible portabilidad de las colecciones de una manera más sencilla.

El servicio de **Azure Blob Storage** es una solución de almacenamiento de datos no estructurados del que hace uso los registros de las aplicaciones del proyecto. Los registros se basan en archivos de texto diarios, estructurados en carpetas que se ordenan temporalmente.

Los modelos generados por la aplicación se almacenan también como dato no estructurado utilizando este mismo servicio.

4. Conclusiones

El desarrollo de este proyecto me ha permitido realizar un trabajo en investigación y desarrollo que complementa la labor diaria que realizo en el seno del equipo de desarrolladores de **Inergy**.

No es habitual destinar tiempo del trabajo de este equipo más allá de los establecido en la planificación establecida para el desarrollo de los productos de software, sobre todo si se trata de una pequeña empresa, como es el caso.

El equipo de desarrolladores y los responsables de producto son conscientes de la necesidad de invertir tiempo en investigación y desarrollo mediante la realización de proyectos paralelos o pruebas de concepto, pero su encaje en la planificación es difícil.

No obstante, el desarrollo de evolutivos y de nuevos productos de software requiere de un permanente conocimiento de las tecnologías existentes en el mercado, aprovechando al máximo lo que estas ofrecen.

SIE es el principal producto de **Inergy** y a lo largo de sus más de diez años de vida ha vivido constantes evolutivos, mejoras y migraciones. Actualmente se tienen almacenados datos de más de dos millones de facturas y un número ingente de suministros eléctricos, así como de otras fuentes energéticas. Empezar a utilizar estos datos para realizar tareas de aprendizaje automático es una necesidad, un requerimiento y una gran oportunidad.

Es habitual que cada organización enfoque el desarrollo de sus soluciones de software a un determinado ecosistema de tecnologías. En el caso de **Inergy** es, desde un inicio, el entorno de trabajo .**NET** de **Microsoft**. Esto no significa una devoción ciega, sino que se pretende estar al día de todas las novedades existentes, no sólo en el ecosistema .NET sino a nivel global.

La apuesta de **Microsoft** por **ML .NET** significa que los desarrolladores que trabajamos con este entorno de desarrollo tenemos la oportunidad de implementar soluciones de aprendizaje automático en **.NET.**

Así mismo, permite la aplicación de los conocimientos adquiridos en la asignatura **Inteligencia Artificial Avanzada** en casos de uso relacionados con la eficiencia energética, el modelo de negocio de **Inergy**.

De manera complementaria, se ha podido realizar desarrollos con los servicios en la nube de **Azure** para el despliegue de las aplicaciones del proyecto, dejando a un lado los servidores virtuales utilizados hasta ahora por la organización.

El plan de desarrollo de **.NET** que tiene **Microsoft** plantea que el entorno **.NET Framework** se quede sin soporte a corto plazo, por lo que se recomienda el uso de **Microsoft .NET Core**. Un entorno multiplataforma y de código abierto.

Este proyecto sirve como entorno de pruebas para la implementación de este nuevo entorno de trabajo, dado que todas las soluciones de **Inergy** han estado desarrolladas con **.NET Framework**.

En definitiva, el proyecto ha servido para poner en común una serie de conceptos teóricos sobre aprendizaje automático y la apuesta por nuevas tecnologías que seguramente servirán para mejorar los productos desarrollados por **Inergy**.

Personalmente es muy gratificante disponer de la posibilidad de experimentar con nuevas tecnologías y apostar por el desarrollo de soluciones que se alejan del día a día habitual, pero con aplicación inmediata.

5. Glosario

1. **.NET.** Conjunto de entornos de desarrollo de Microsoft. **.NET Framework**, propietario y **.NET Core** de código abierto y multiplataforma.
2. **API.** Interfaz de programación de aplicaciones, métodos y funciones como biblioteca de código.
3. **Azure Blob Storage.** Servicio en la nube de almacenamiento de datos no estructurados.
4. **Azure Cosmos.** Servicio en la nube de bases de datos no relacionales.
5. **Azure Functions.** Servicio en la nube para la ejecución de tareas sin servidor.
6. **Azure Web Jobs.** Servicio en la nube para la ejecución de tareas programadas.
7. **Azure Web Apps.** Servicio en la nube para el alojamiento de aplicaciones web.
8. **Blazor.** Librería del entorno .NET de Microsoft para el desarrollo de aplicaciones web asíncronas.
9. **DetectSpikeBySsa.** Algoritmo para la detección de anomalías en una serie temporal. Documentación.
10. **Diagrama de Gantt.** Diagrama para la representación temporal de distintas tareas.
11. **FastTreeTweedie.** Algoritmo de regresión lineal en ML .NET. Documentación.
12. **ForecastBySsa.** Algoritmo para la previsión de valores de series temporales en ML .NET. Documentación.
13. **Framework.** Entorno de desarrollo de una determinada tecnología.
14. **Machine Learning Aprendizaje Automático.** Conjunto de algoritmos y técnicas para facilitar el entrenamiento de modelos de aprendizaje por parte de aplicaciones.
15. **MDMS.** Sistema de almacenamiento de series temporales en base de datos relacional en Inergy.
16. **ML .NET**. Librería y API del entorno .NET de Microsoft para el desarrollo de soluciones de aprendizaje automático.
17. **Mongo DB.** sistema de base de datos no relacional.
18. **Open API.** Especificación estándar para el desarrollo de web API.
19. **REST.** Arquitectura HTTP apara la implementación de API mediante dicho protocolo en la web.
20. **Serverless.** No requiere de servidor para la ejecución de una aplicación.
21. **SQL Server.** Servidor de bases de datos relacionales de Microsoft.
22. **SIE.** Sistema de información energética de Inergy.
23. **Telerik.** Componentes web propietarios para plataformas Javascript y .NET.
24. **UTC.** Hora universal.

6. Bibliografía

1. **.NET (web):** Documentación en inglés: dotnet.microsoft.com
2. **Azure App Services (web):** Descripción del servicio de *Azure* en castellano: azure.microsoft.com/es-es/services/app-service
3. **Azure Blob Storage (web):** Documentación del servicio en castellano: docs.microsoft.com/es-es/azure/storage/blobs/
4. **Azure Functions (web):** Documentación en castellano: docs.microsoft.com/es-es/azure/azure-functions
5. **Azure Web Jobs (web):** Documentación en castellano: docs.microsoft.com/es-es/azure/app-service/webjobs-create
6. **Blazor (web):** Documentación en inglés: dotnet.microsoft.com/apps/aspnet/web-apps/blazor
7. **C# Driver Mongo DB. (web):** Documentación en inglés del driver C# de **MongoDB**: docs.mongodb.com/ecosystem/drivers/csharp
8. **Cosmos DB (web):** Documentación en castellano: docs.microsoft.com/es-es/azure/cosmos-db/introduction
9. **Inergy (web):** Página web disponible en castellano, catalán e inglés: inergybcn.com
10. **Local Emulator Cosmos DB (web):** Documentación en castellano: docs.microsoft.com/es-es/azure/cosmos-db/local-emulator
11. **ML .NET (web):** Documentación en inglés: dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet
12. **Open API (web):** Página web en inglés: www.openapis.org
13. **SIE. (web):** Descripción en la página web: inergybcn.com/sie
14. **Telerik Blazor Components (web):** Documentación en inglés: docs.telerik.com/blazor-ui/introduction
15. Raúl Benítez, Gerard Escudero, Samir Kanaan. **Inteligencia artificial avanzada**. UOC **(pdf)**

7. Anexos

Figura 2.1. *Diagrama que muestra la integración de la* ***API Web*** *con el resto de las aplicaciones y su capa de datos.*

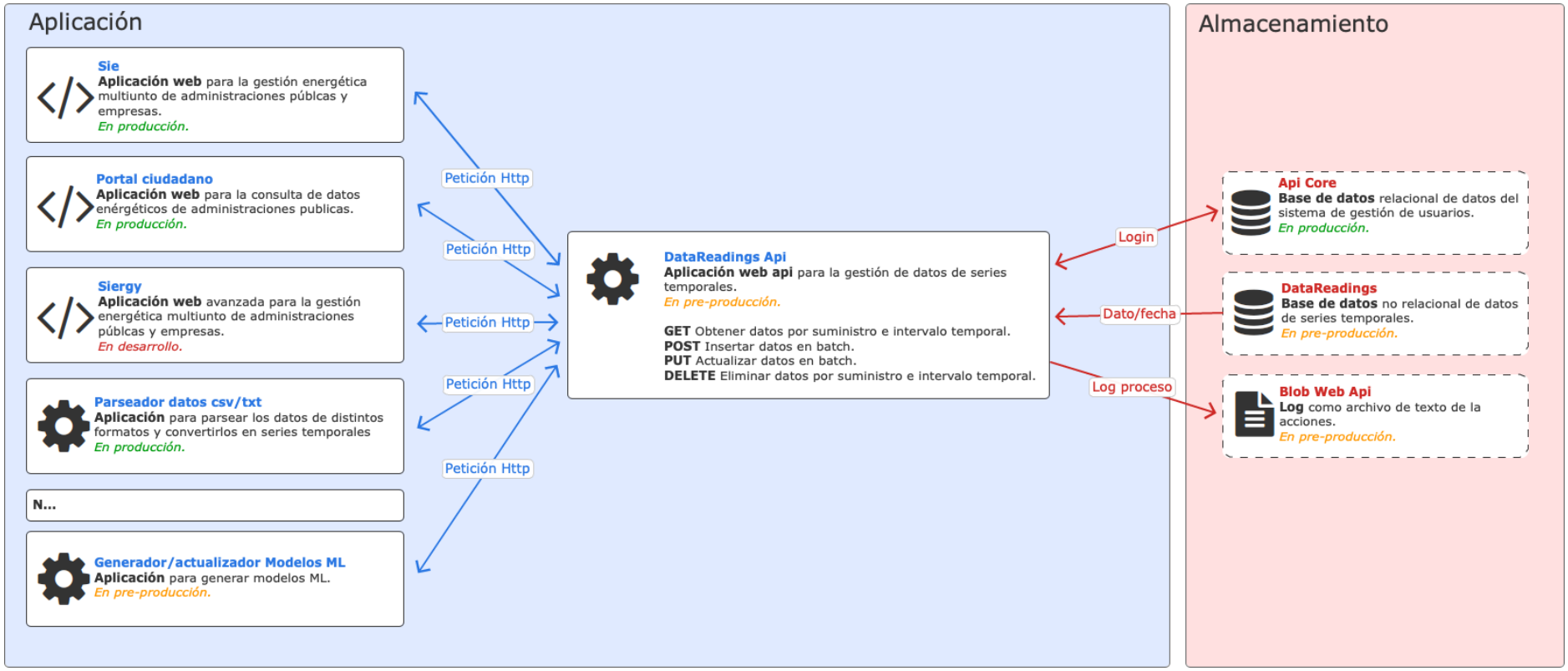


Figura 2.2. *Flujo de la aplicación de migración de datos de MDMS y sistemas de almacenamiento que intervienen en el flujo.*

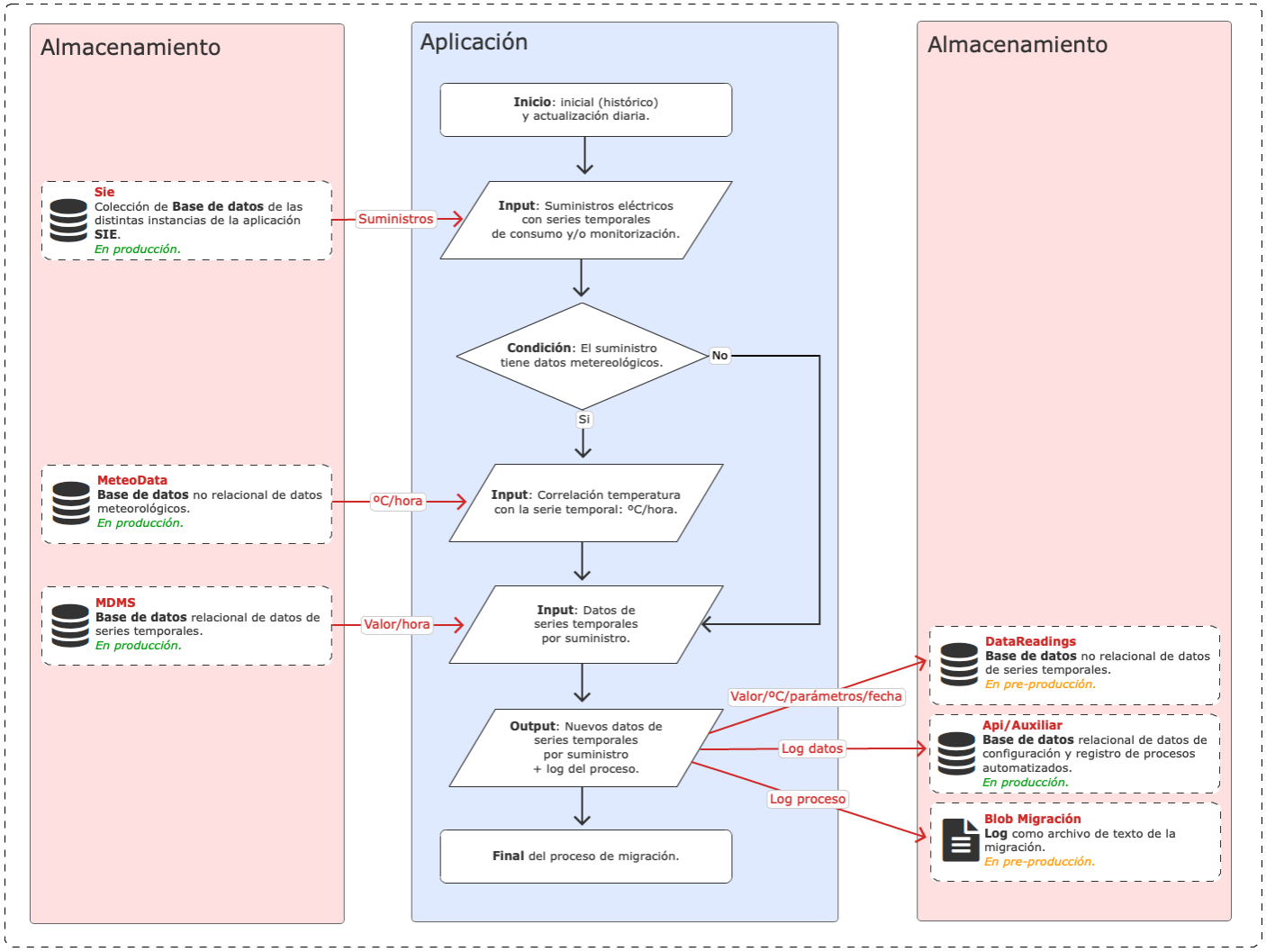


Figura 2.3. *Flujo de la aplicación de generación de modelos ML y sistemas de almacenamiento que intervienen en el flujo.*

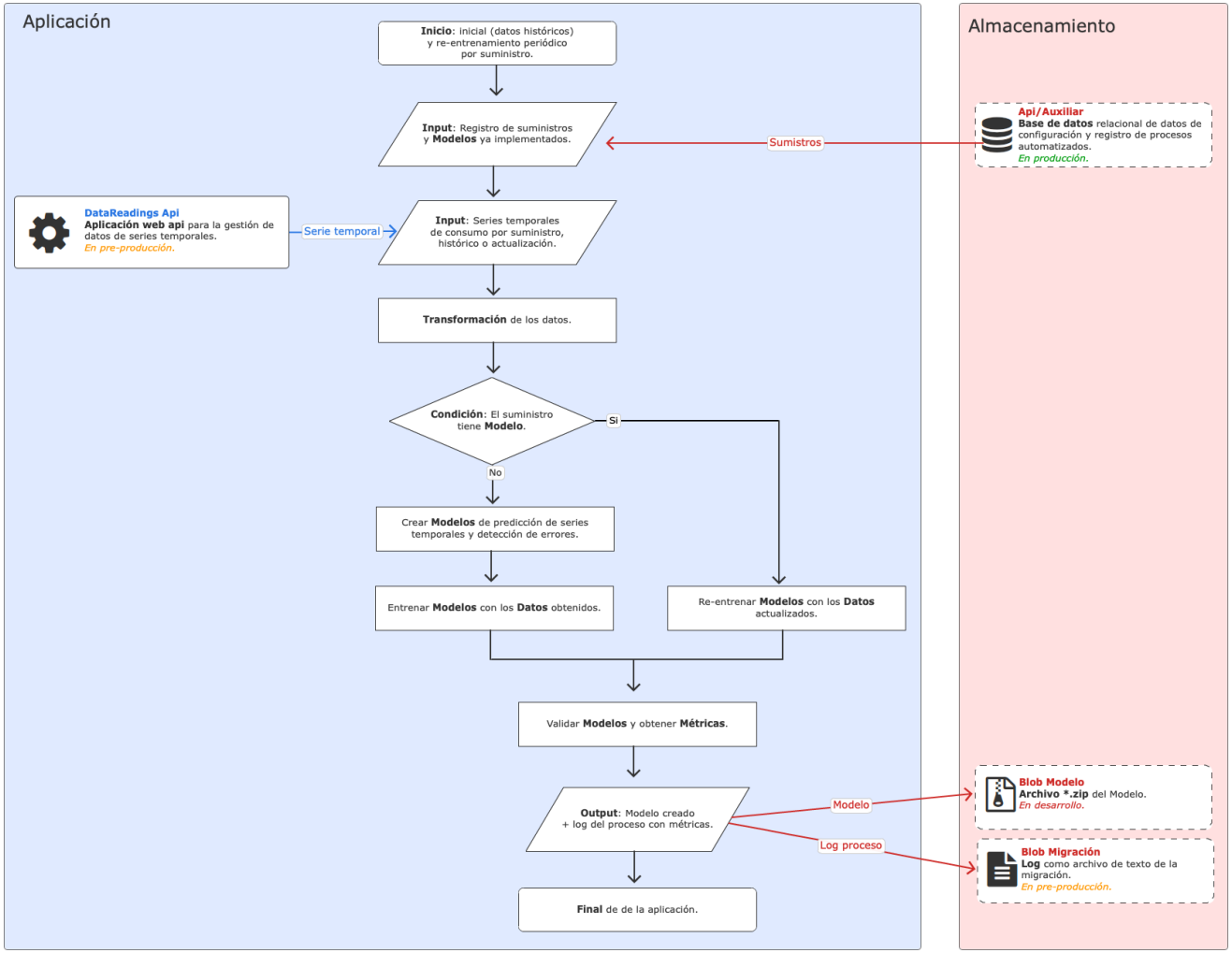


Figura 2.4. *Diagrama que muestra la integración de la aplicación de consumo de modelos y el almacenamiento relacionado.*

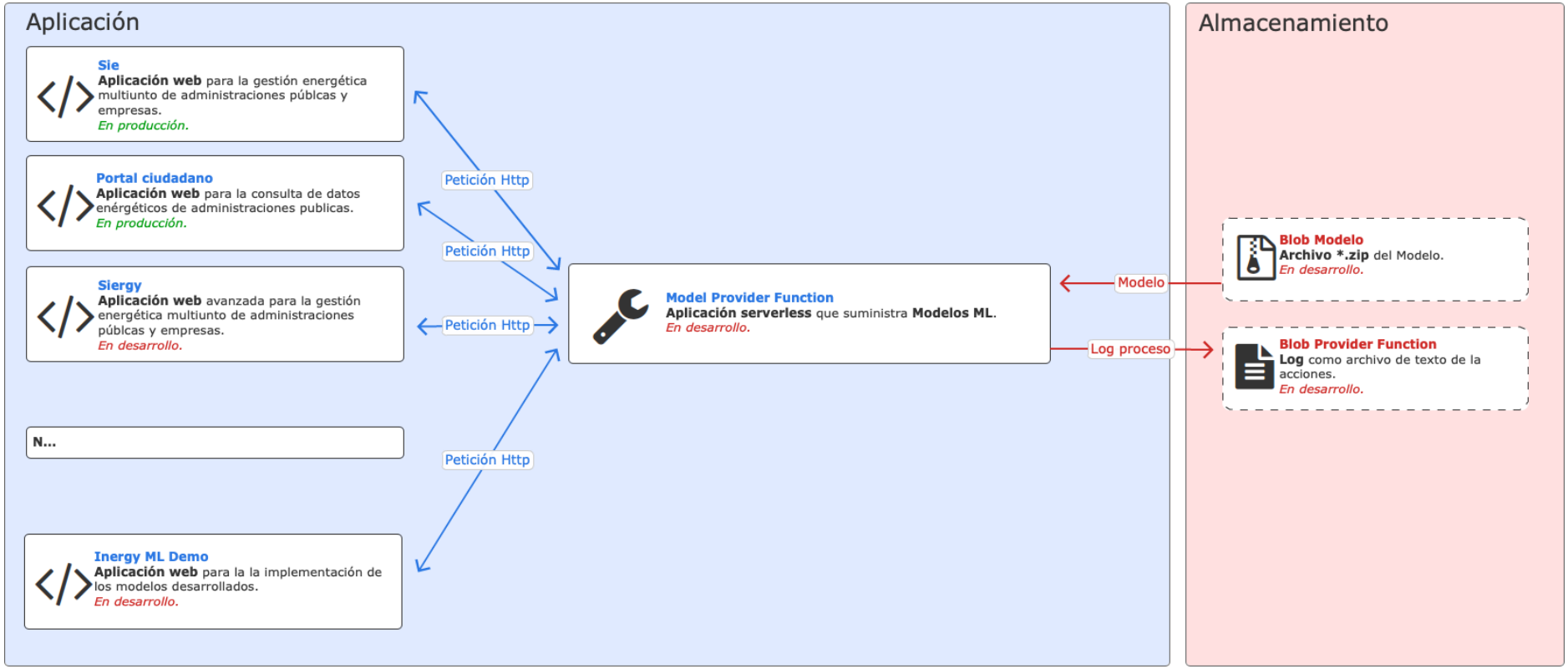


Figura 2.5. *Diagrama que muestra las características de la* ***Web Demo*** *y como se comunica con la aplicación de consumo de modelos.*

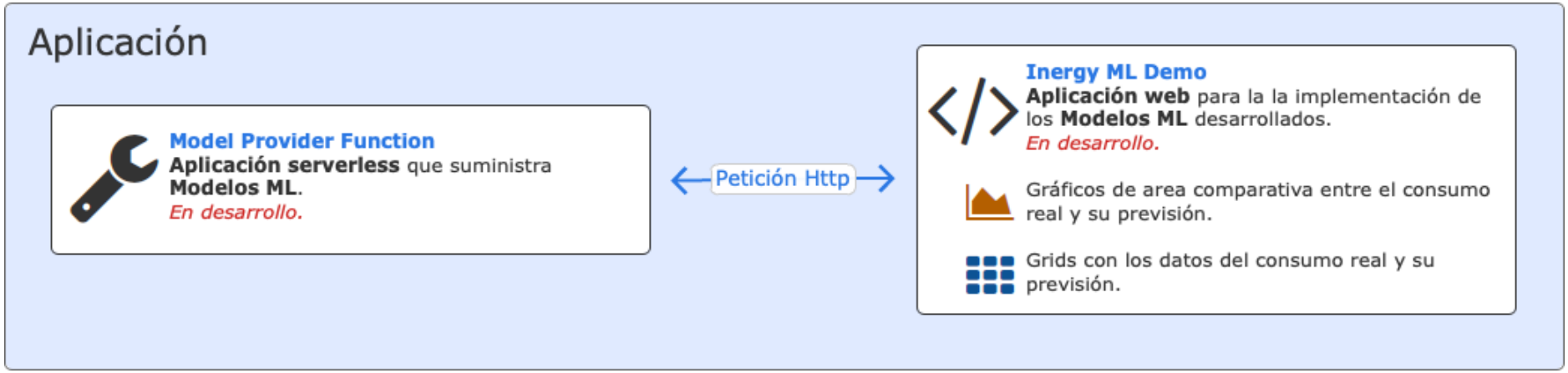


Figura 3.1. *Aplicaciones del proyecto y servicio de Azure en los que se despliega, así como Framework .NET correspondiente.*

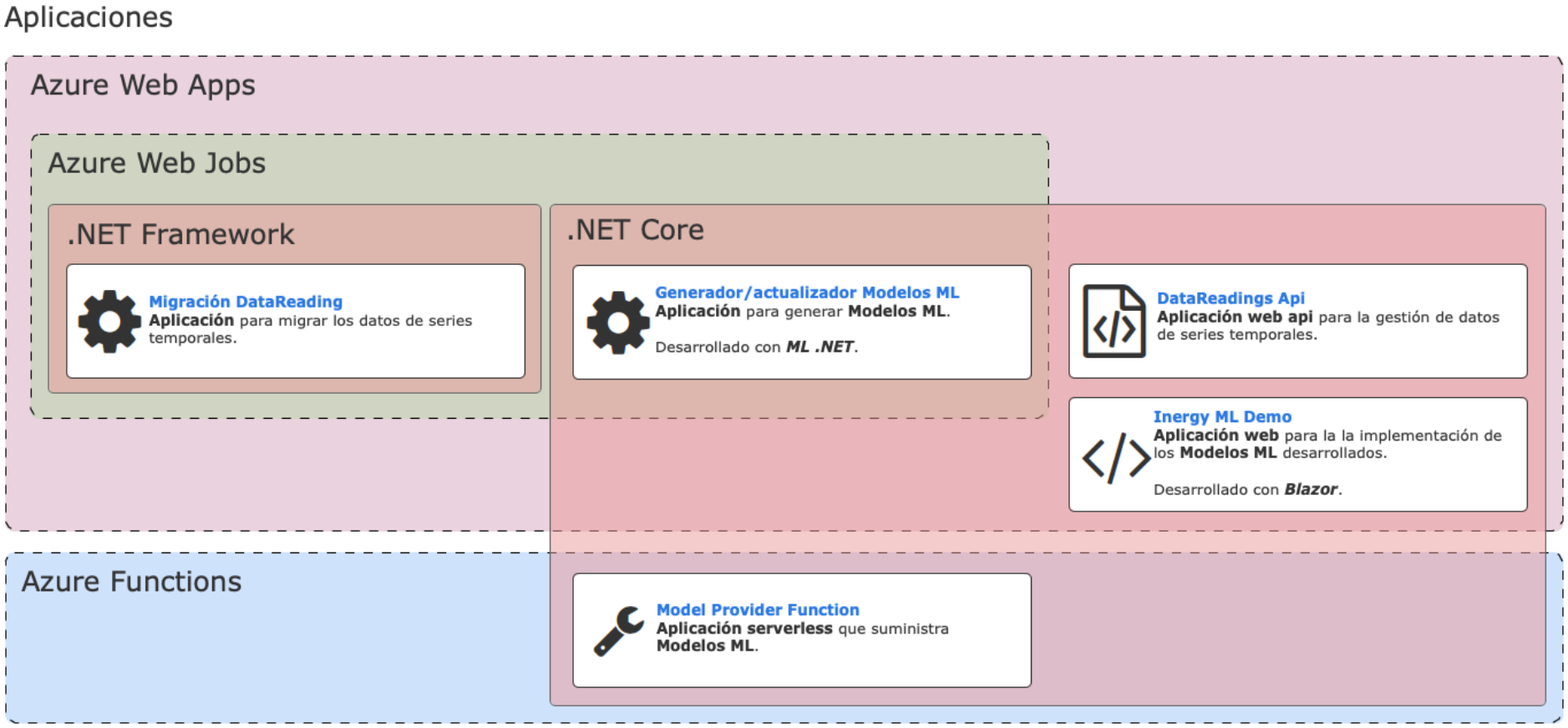


Figura 3.2. *Tipologías de almacenamiento del proyecto, así como bases de datos y tipologías de archivos almacenados.*