ESPECIFÍCACIÓN

**CODIGO.**

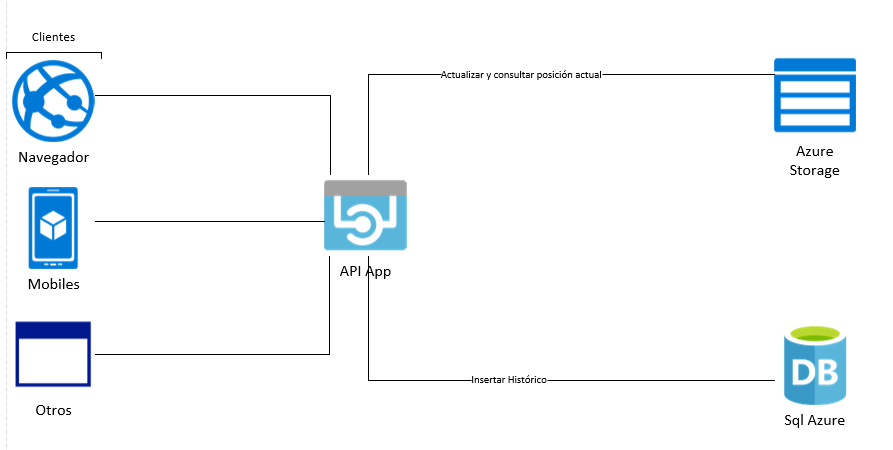
Se desarrollo una solución para atender el requerimiento dado.

La Solución comprende las siguientes capas:

* Tools
  + Conjunto de librerías de propósito general (capa transversal). Compre proyectos de acceso a base de datos SQL, acceso al Azure Storage, Obtención de recursos de configuración, componente de loggeo de errores, excepciones personalizadas, etc.
* Database
  + Tiene los scripts de las estructuras de base de datos y procedimientos almacenados usados.
* Application
  + Contiene el proyecto de **API Rest**, el cual es la interfaz para poder realizar el registro/actualización de la posición del vehículo, así como consultar dicha posición como su histórico.
  + Contiene las librerías de Dominio, Lógica de negocio y de Acceso a Datos.
* Testing
  + Proyecto de pruebas unitarias
* Docs
  + Se agregan los documentos de insumos y entregables.
    - Requerimiento
    - Diagrama de arquitectura
      * Se agrega un diagrama PROPUESTA de cómo podría ser otra opción más robusta para atender el requerimiento.
    - Evidencias de pruebas
    - Especificación
    - Diagrama de Base de Datos
* Architecture
  + Se agrega diagramación UML (Falto)

**ARQUITECTURA.**

Se crea un diagrama de componentes para indicar el despliegue de la solución.



**COMPONENTES.**

Se detalla la función de cada componente.

**Clientes:**

* Conjunto de sistemas que consumen la API

API App:

* Aplicación ASP NET API Rest, es la aplicación que expone la funcionalidad para registrar/actualizar y consultar las coordenadas del taxi.

Estas son las acciones permitidas:

**1.-**

***http://gbmchallengeservice.azurewebsites.net/api/geolocation/coordinates/add***

Permite agregar o actualizar la ubicación actual del taxi, también almacena el registro en el histórico.

La “ubicación actual del taxi” se registra en una tabla de Azure, por temas de rendimiento es más optimo usar este tipo de almacenamiento no relacionado, porque para la consulta de la ubicación actual solo se hace por la llave primaria (PartitionKey y RowKey) (TravelId y TaxyId). Debido al gran nivel de particionamiento que tiene la tabla, las consultas se hacen extremadamente rápidas sin importar el tamaño ni cantidad de registros de la tabla. Las tablas de Azure son recomendadas para la alta transaccionalidad en consultas que solo impliquen PartitionKey y RowKey.

El registro del histórico de las “ubicaciones del taxi” se almacena en una tabla de SQL, debido a que las consultas no pueden hacerse por la llave primaria sino por otros campos (TravelId y TaxyId), este tipo de almacenamiento es el mas idóneo, ya que la consulta podría regresar una gran cantidad de registros. Aquí se aprovecha la indexación y cache del motor de SQL Server. Independiente de esta simple consulta se pueden crear mas tipos de consultas mas complejas como consulta por DriverId, Por fechas, etc. Ya si aprovechar el esquema relacional de datos.

Se usa mediante POST.

**2.-**

***http://gbmchallengeservice.azurewebsites.net/api/geolocation/coordinates/current***

Permite la consulta de la “ubicación actual del taxi”. La consulta la hace por (TravelId y TaxyId) hacia la tabla de Azure. Regresa una entidad de Geolocalización.

Se usa mediante POST.

**3.-**

***http://gbmchallengeservice.azurewebsites.net/api/geolocation/coordinates/history***

Permite la consulta del histórico de ubicaciones “ruta” de un viaje de un taxi. La consulta la hace hacia la base de datos “GBMChallengeDB” de SQL mediante ADO.NET con procedimiento almacenado.

Se usa mediante POST.

**Azure Storage**

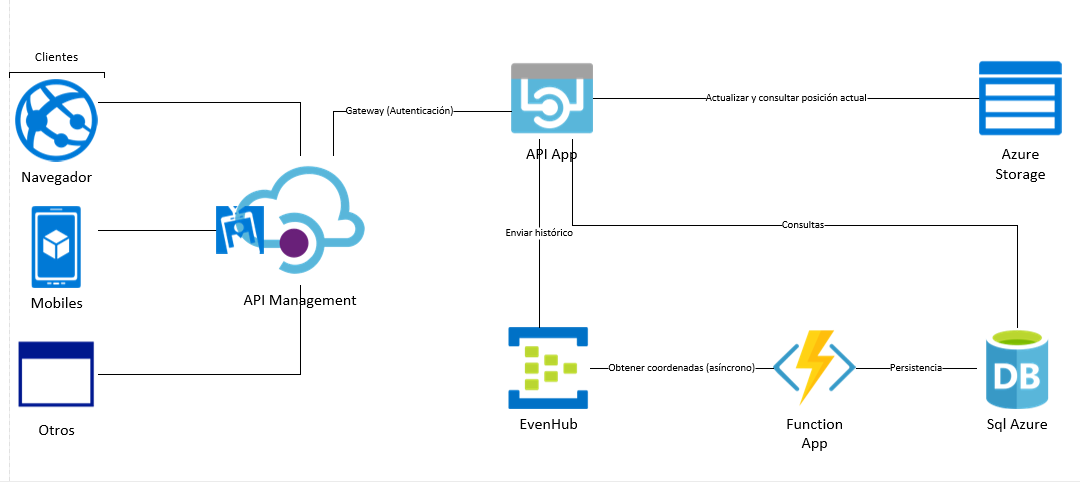
* Almacenamiento NOSQL de la plataforma de Microsoft Azure, se usan solo las Tablas, permiten un crecimiento sin perder rendimiento. Y es almacenamiento de muy bajo coste.

**SQL Azure**

* Almacenamiento relacional, incorpora casi todas las bondades del producto MS SQL Server, sin embargo, es en un entorno más integrado y menos administrable, para fines de prototipos es ideal. No requiere licenciamiento

**PROPUESTA.**

A continuación, se muestra el diagrama de arquitectura de una mejor propuesta:



Se propone una mejora en el registro del histórico, ya que actualmente se registra la “ubicación actual” y la “histórica” en el mismo proceso SINCRONO, se propone que el registro del histórico sea **ASINCRONO**. Y que la seguridad este a cargo del componente “API Management”

**API Management**

* Gateway que brinda seguridad hacia diferentes aplicaciones “Apps”, expone extremos seguros. Se propone implementar seguridad vía Token y Certificado (SSL, Https, Autenticación mutua, Token de validación y demás políticas de acceso) para cada uno de los 3 extremos REST de la API.

**Event Hub**

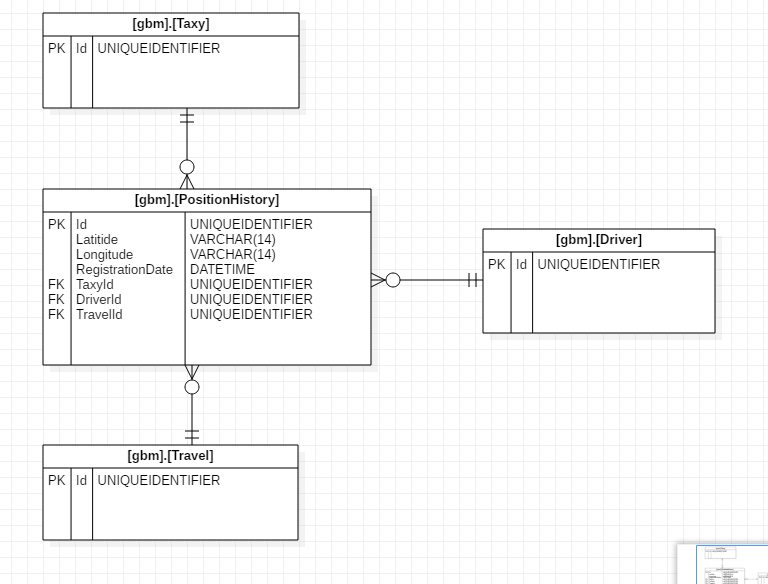
* En vez de registrar el histórico, es más rápido encolar un mensaje con las coordenadas. Se propone encolar mensajes hacia un “Event Hub”. En el proceso síncrono cuando se actualiza la ubicación actual se encola un mensaje con las coordenadas para que a posterior un proceso asíncrono lo persista en la base de datos.

**Function App**

* Función de Azure, que se ejecuta en ambiente sin servidor “Serverless”, este proceso se dispara de manera automática cuando entra un mensaje al Event Hub, Obtiene el mensaje y lo registra en la tabla de histórico de la base de datos.

**BASE DE DATOS.**

Se muestra el diagrama de la base de datos (solo es representativo) lo que se debe notar es la tabla de histórico “[gbm].[PositionHistory]” y su relación con otras tablas que ya debería existir.



Como el registro de la “Ubicación actual” se almacena en una tabla del Storage, dicha tabla ya no se incluye en el diagrama.

El acceso a la base de datos y al storage se encuentra en el archivo de configuración del Web API Rest.

SALUDOS…Cualquier duda quedo atento.

NOTA:

* Ya no me dio tiempo hacer las pruebas unitarias automatizadas, pero si hice pruebas “Se anexa documento de evidencias”
* Tampoco el Challenge 2 ☹
* Desplegué la API: **http://gbmchallengeservice.azurewebsites.net/**