Descripción General

La solución propuesta se compone de diferentes capas y servicios, cada uno seleccionado para cubrir aspectos clave: disponibilidad, rendimiento, seguridad, escalabilidad, observabilidad y facilidad de mantenimiento.

Objetivos Clave:

- Ofrecer una API B2B que procese solicitudes complejas con tiempos aproximados a los 15s por petición en condiciones normales.
- Manejar grandes volúmenes de datos.
- Garantizar picos de 50 solicitudes/seg durante ciertos meses del año (marzo, abril, mayo) y mantener costes razonables el resto del tiempo.
- Proporcionar una seguridad robusta.
- Registrar métricas y logs para análisis mensual, seguimiento de errores y performance.
- Implementar monitorización E2E que dispare alertas y cree incidencias automáticamente si el sistema falla.

Para ello la arquitectura se basará en un conjunto de microservicios desarrollados en .NET 8 y contenedorizados, que se desplegarán en un cluster de Kubernetes, lo que permite mayor control sobre el entorno de ejecución, mayor flexibilidad en la configuración de escalado.

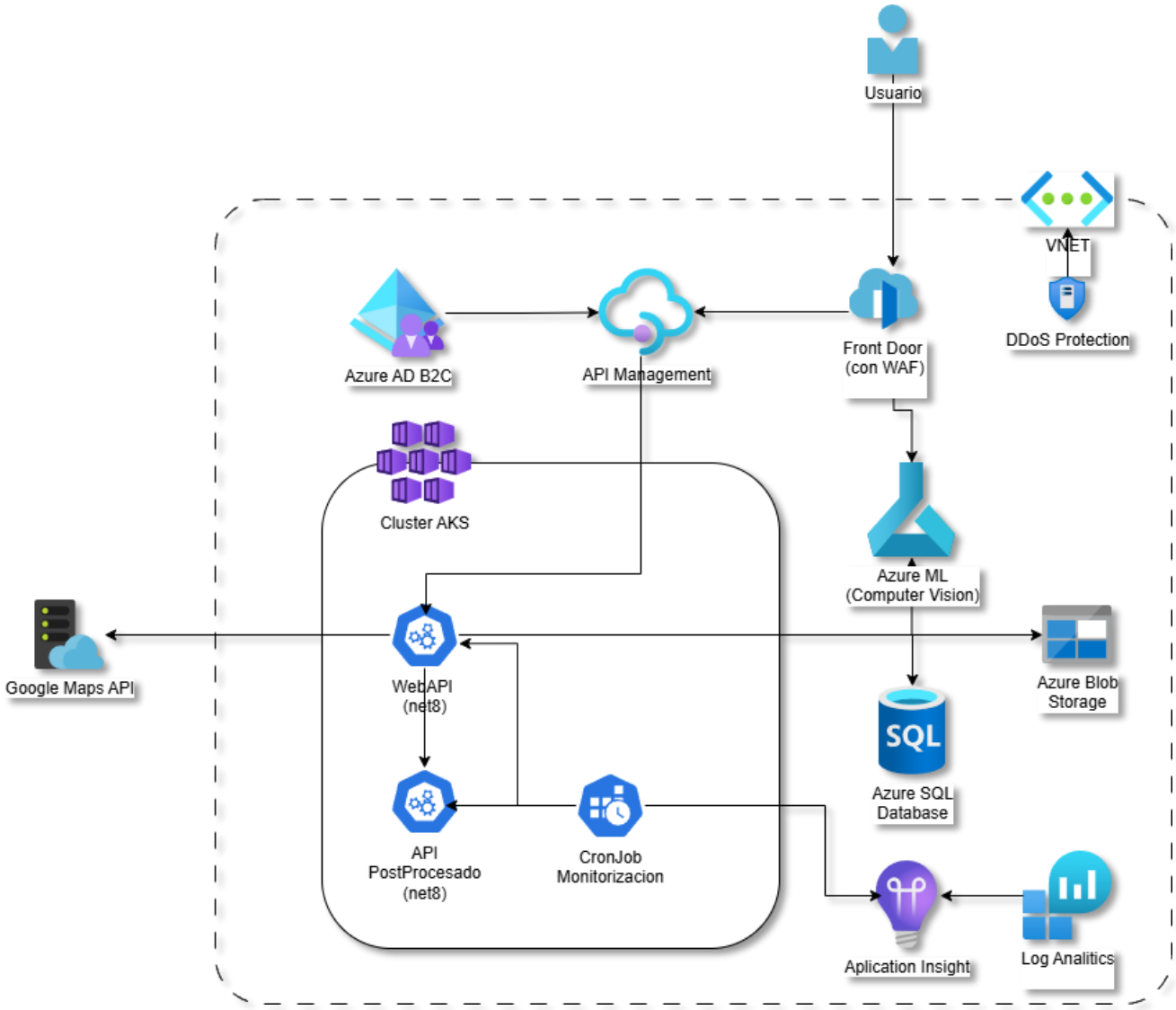
Estos microservicios utilizan a su vez un conjunto de servicios de Azure para su correcta ejecución.

Requerimientos y Consideraciones

Servicio Principal	• Consulta a servicios externos (Google Maps). • Acceso a una base de datos con 150M de registros de límites catastrales. • Llamada al proceso de Inferencia • Llamada al servicio de Post-Procesado
Modelo de Inferencia	• Modelo de Computer Vision (AI)
Servicio de Post-Procesado	• Lógicas de negocio adicionales sobre los resultados del modelo de inferencia
Almacenamiento de BDD	• Base de datos con 150 millones de registros con datos geoespaciales.
Logging y Estadísticas Mensuales	• Registros de tiempos, número de peticiones, errores, etc.
Monitorización del E2E	• Petición de prueba periódica con resultado esperado. • Si falla o excede el timeout, crear incidencia en JIRA.

Alertas en Azure	• Para degradación del servicio, falta de disponibilidad
Carga	• 2M peticiones/año (~5.500 peticiones/día en promedio). • Picos de 50 peticiones/sec. en marzo, abril, mayo. • Valles de 1-2 peticiones/sec. el resto del año.
Almacenamiento de ficheros	• JSON de 5KB a 30KB. • Imágenes de 400KB a 1MB.
Control de Acceso por Cliente	• Autenticación • Autorización • *Rate-limiting* • Facturación por uso

Diagrama de la Arquitectura



Ejemplo de Flujo de una Petición Normal:

1. El cliente B2B envía una petición con su token JWT a través de Front Door.
2. Front Door (con WAF) enruta a APIM.
3. APIM valida el token JWT, verifica cuotas, si todo OK, reenvía la petición a la API en AKS.
4. La API recoge la petición y:
    - Consulta Google Maps (2s).
    - Consulta la BD catastro (PostgreSQL Hyperscale) para obtener límites (cachea resultado en Redis si es común).
5. La API envía las imágenes o datos preprocesados al endpoint de AML. (10s).
6. El AML Endpoint retorna el resultado de la inferencia.
7. Una vez con la inferencia, la API realiza el post-procesado (3s).
8. La API guarda el resultado (JSON, imagen) en Blob Storage.
9. La API retorna la respuesta al cliente con la referencia al resultado.
10. Application Insights registra el tiempo total (~15s), Logs guardan detalles, métricas se actualizan.

Descripción de la Arquitectura por Capas

Capa de Entrada

- **Azure Front Door** que sirve como punto de acceso global al API.

  Permite balanceo de carga global, en caso de necesitar un futuro despliegue geografico, además de ofrecer una Alta disponibilidad y definir reglas de enrutamiento flexibles. Además de proveer de SSL/TLS.

- **Web Application Firewall (WAF)** integrado con Front Door proporciona protección contra amenazas comunes.

Capa de Gestión de la API

- **Azure API Management (APIM)** se encargará de exponer las APIs a los clientes B2B.

  Integrándose con Azure AD B2C proporciona autenticación y autorización basadas en OAuth2/JWT

  Permite la aplicación de políticas de rate-limiting y cuotas por cliente a la par que monitoriza el uso por parte de los clientes de cara a la facturación.

  Además, genera automáticamente la documentación de la API y soporte el versionado de esta.

Capa de Procesado

- **Azure Kubernetes Service (AKS)** donde se ubicaran y ejecutaran los contendores de los diferentes servicios.

  Estará formado por dos *pools* de nodos:

  - **Pool principal (CPU nodes):** Aloja los servicios de producción:
    - **Servicio Principal** (Web API en net8)**:** Endpoints de la API con las llamadas necesarias para la obtención de datos necesarias para el modelo de inferencia, la llamada al modelo de inferencia y la llamada al servicio de post-procesado.
    - **Servicio Postprocesamiento** (Web API en net8)**:** Endpoints con la lógica de negocio para procesar la respuesta del modelo de inferencia.
    - **Servicio de Monitorización E2E** (Aplicación de consola en net8)**:** Programación con Kubernetes CronJobs que cada X tiempo se ejecute la aplicación, realice la prueba E2E y llame a JIRA en caso de fallo.
  - **Pool secundario (CPU nodes):** Que alojará los servicios de Preproduccion. Lo que nos permitirá su apagado en los tiempos donde no se estén haciendo pruebas (por la noche, por ejemplo)

  Cada pool tendrá configurado *Node Autoscaler* para añadir nodos en picos de carga.

  Cada servicio estará definido en su propio *deployment* de AKS, tendrá configurado *Horizontal Pod Autoscaler* según CPU/RAM.

  Para evitar en un primer momento tener dos clusters, uno para PRE y otro para PRO, trabajaremos con los *namespaces* de modo que crearemos un *namespace* para cada entorno, y configuraremos los *deployment* de cada servicio para que se despliebe en un pool u en otro dependiendo del entorno en el que se esté desplegando.

- **Azure Machine Learning Managed Endpoint** donde se encontrará desplegado, como un *endpoint*, Modelo de inferencia. Y estará configurado para que se escalar tanto vertical como horizontalmente.

Capa de Base de Datos

- **Azure Cosmos DB**: ya que se requieren consultas geoespaciales, además de una alta escalabilidad horizontal.
- Al ser datos de parcelas de terreno que probablemente se consultan por coordenadas, podemos aprovechar los *Spatial Indexes* de Cosmos DB.

Capa de Almacenamiento de Ficheros

- **Azure Blob Storage (posiblemente ADLS Gen2 si se requiere Data Lake)** donde alojar uno o varios contendores (en función de la estructura que se necesite) para los ficheros JSON y las imágenes.

Se securizará el acceso mediante el uso de tokens SAS o Managed Identities.

Y se configurará con redundancia geográfica para asegurar una alta disponibilidad en caso de necesitar un futuro despliegue geográfico del sistema.

Capa de Logs y Monitorización

- **Azure Application Insights** + **Azure Monitor**: para recolectar y a su vez poder consultar toda la telemetría y ls métricas de los servicios de manera detallada (tiempos de respuesta, errores, métricas personalizadas). Además nos permite mantener los logs centralizados.

- **Azure Log Analytics Workspace**: junto con Azure Monitor tenemos los logs centralizados lo que nos permite un análisis avanzado de estos mediante consultas KQL
- **Azure Monitor Alerts** nos permite configurar alertas basadas en las métricas (latencia, errores 5XX, tiempos mayores a x segundos, etc.).

Capa de Seguridad

- **Azure AD B2C** para la gestión de usuario asi como la autenticación y autorización de estos. Este emitirá los tokens JWT que valide posteriormente el API Management y aplique políticas de acceso.
- **Azure Front Door + WAF** ya descritos en la Capa de Entrada
- **DDoS Protection Standard** en las subnets de VNET para impide saturación a nivel de red.
- **Identidades Gestionadas (Managed Identities)** usadas por los Servicios para acceder a la BD y al Storage sin almacenar credenciales.
- **Key Vault** para almacena secretos (si se requieren claves para cifrado, tokens para Google Maps, etc.).

Integración de código y despliegue

CI/CD con Azure DevOps

1. Repositorio de Código (Azure Repos):
    - Dicho repositorio tendrá tres ramas
        1. **Develop**: donde se integrará el código implementado por los desarrolladores
        2. **Main**: Donde se integrará el código para su despliegue en PRE y en PRO. A esta rama solo se le podrán hacer Pull Request desde *develop*
2. Pipelines de Build (Azure Pipelines) configurados para que se ejecuten cuando re realice *un Pull Request* correctamente sobre la rama *Main* Los pasos principales serán:

- Compilar los proyectos .NET.
- Ejecutar test unitarios.
- Crear imágenes Docker a partir del Dokerfile definido en la solución.
- Subir a Azure Container Registry la imagen generada.
- Generar y publicar los “artefactos” necesarios para su posterior despliegue. Esto principalmente serán los manifiestos de kubernetes tanto para PRE como para PRO

3. Pipelines de Release configurados para que se ejecuten cuando se haya ejecutado correctamente su build correspondiente.
    - Toma las imágenes de Azure Container Registry.
    - Toma los “artefactos” publicados por la build
    - Despliega a AKS mediante manifiestos K8s.
    - Cada Pipeline tendrá dos stages, una para el despliegue en PRE y otra para su despliegue en PRO
    - La ejecucion de cada stage requerirá autorización manual y los despliegues en PRO se programarán para su ejecución en horas de bajo tráfico.

Estimación de Costos

Suposiciones principales

1. **Tráfico y Escalado**:
    - Se estiman unos 2 millones de peticiones/año, unas ~5.500/día en promedio.
    - Picos: 50 req/seg en marzo, abril y mayo (3 meses); resto del año 1-2 req/seg.
    - Las peticiones duran ~15s en total. Para 50 req/s, hay ~750 peticiones concurrentes.
2. **Recursos Clave**:
    - Front Door + WAF
    - API Management (APIM)
    - AKS (clúster principal con autoscaling) + ACR
    - Servicio de Inferencia (Azure ML Managed Endpoints)
    - Base de Datos (Cosmos DB con índices geoespaciales)
    - Blob Storage para JSON e imágenes
    - Application Insights / Log Analytics / Monitor
    - Azure AD B2C para autenticación

- DDoS Protection Standard
- Key Vault

Costos Mensuales (Aproximados)

1. **Azure Front Door + WAF:** puede rondar entre 80€ - 150€/mes según tráfico y reglas.
2. **API Management (APIM):** Usando Tier Standard (Incluye funcionalidades de autenticación, cuotas, etc.): ~500 €/mes (aprox.)
3. **AKS (Compute + Control Plane)**
   - Nodo D4_v3 (4vCPU, 16GB RAM) ~135€/mes/nodo (aprox.)
   - En valles: 2 nodos = 270€/mes
   - En picos (3 meses al año): ~8 nodos = ~1080€/mes. Promediando el año (9 meses a bajo coste, 3 meses a alto coste) da un promedio intermedio.
   - Sumando costes del plano de control, LB, etc.: ~550 €/mes de media anual.
4. **Azure ML Managed Endpoints:**
   - Si la inferencia requiere 10s y necesitamos manejar picos de 50 req/seg, asumamos un escalado a ~10 instancias en picos y 1-2 instancias en valles.
   - Cada instancia (por ejemplo D4_v2) ~200 €/mes.
   - En picos (3 meses): 10 instancias = 2000 €/mes
   - Resto del año (9 meses): 1 instancia = 200 €/mes
   - Promedio año: (9*200* + *3*2000) /12 = (1800+6000)/12 = 7800/12 ≈ 650 €/mes
5. **Cosmos DB:**
   - 150M registros, con consultas espaciales.
   - RU/s autoscale, digamos hasta 5k-10k RU/s.
   - Un coste aproximado podría oscilar entre 500€ - 2000€/mes dependiendo del uso real.
6. **Azure Blob Storage:**
   - ~2M peticiones/año = ~166k/mes.
   - Imágenes (400KB - 1MB) y JSON (5-30KB).
   - Almacenando unos GBs al mes. Coste de almacenamiento + operaciones: ~50€/mes.
7. **Logging y Monitorización (App Insights, Log Analytics):**
   - 20-30GB de logs/mes ~ 50€ - 80€ mes (ingestión + retención).
   - Añadir coste de Application Insights, consultas, alertas: ~80 €/mes total.
8. **Azure AD B2C** por demanda. Si se asume ~166k solicitudes/mes y caché de tokens, quizá 200€ mes.
9. **DDoS Protection Standard:** Entorno pequeño/mediano: ~260 €/mes
10. **Azure Key Vault:** Uso moderado: ~20 €/mes

Suma de Costes Aproximada

Azure Front Door + WAF	100 €
Azure API Management	500 €
Azure Kubernetes Service	550 €
Azure ML Managed Endpoints	650 €
Cosmos DB	1000 €
Azure Blob Storage	50 €
Logging y Monitorización	80 €
Azure AD B2C	200 €
DDoS Protection Standard	260 €
Azure Key Vault	20 €
TOTAL Aproximado	**≈ 3410 € / mes**

Estrategias de Optimización de Costes

1. **Escalado Dinámico y Sobredimensionamiento Controlado:**
    - Ajustar el Horizontal Pod Autoscaler (HPA) en AKS para que sólo escale a más nodos/pods durante picos reales.
    - Para la inferencia en AML, evaluar caching, batching o colas asíncronas, reduciendo el número de instancias en picos.
    - Apagar entornos de preproducción por la noche.
2. **Capa de Datos (Cosmos DB)**
    - Implementar caching (Azure Cache for Redis) para reducir RUs en Cosmos DB.
    - Ajustar índices, particiones y usar autoscale con límites mínimos más bajos.
    - Considerar almacenamiento en caliente vs. frío y archivado de datos antiguos.
3. **API Management:**
    - Evaluar el uso de un tier inferior si el volumen de peticiones lo permite.
    - Consolidar funciones y reducir overhead si no se necesitan todas las características premium.
4. **Modelo de Inferencia:**
    - Optimizar el modelo de inferencia para reducir el tiempo de respuesta (10s a algo menor) o aumentar la concurrencia por instancia.

- Evaluar contenedores propios en AKS en lugar de AML Managed Endpoint si el coste es elevado, controlando el escalado con Keda o HPA.
- Batch inference (si la lógica de negocio lo permite) y usar colas (Azure Queue Storage + función asíncrona) para procesar peticiones en lotes.

5. **Blob Storage:**
    - Utilizar almacenamiento Archive o Cool para datos no frecuentes.
    - Reducir operaciones de lectura/escritura innecesarias.
6. **Logs y Monitoring:**
    - Filtrar y reducir el nivel de detalle de logs.
    - Ajustar los períodos de retención y eliminar datos antiguos.
    - Monitorizar con alertas específicas y métricas clave en lugar de capturar absolutamente todo.
7. **Autenticación (B2C):**
    - Implementar caching de tokens para reducir el número de autenticaciones totales.
    - Ajustar políticas de expiración de tokens.