Moralar

Manual de instalación para Api Moralar

Tabla de contenido

1. Detalles del proyecto

Api Moralar: Es una Api hecha en **.NET8** donde ofrecemos servicios privados para nuestros aplicativos web y móviles de Moralar.

En esta api manejamos los estándares de una rest-api con toda la seguridad necesaria de JWT y toda la seguridad necesaria que nos ofrece las últimas versiones de .NET.

Los desarrolladores tanto front como back de Moralar podrán consultar nuestro **Swagger** de Moralar para que les sea más fácil consumir nuestros servicios https. Más adelante se les mostrará cómo acceder a esta gran herramienta.

MongoDB: La api de Moralar utiliza MongoDB para el llamado de datos no relacionados.

1.1 Chequear la raíz del proyecto

a. Opcion A:
   - Clonar el proyecto el cuál ya le han pasado el repositorio, acceder a la rama “Development” y luego ingresar a la raíz del proyecto donde la carpeta se llama ***moralar-api***.

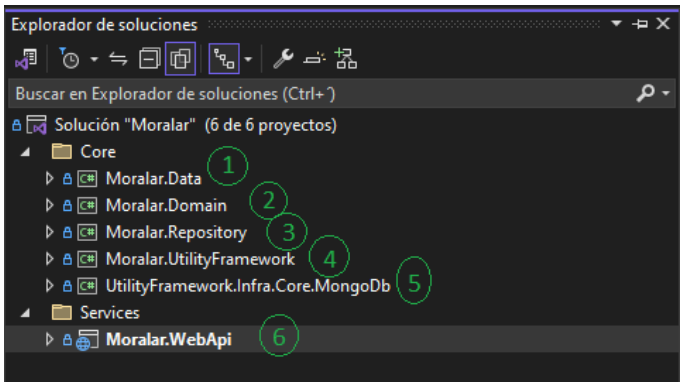
   Opción B:

   - Acceder a la carpeta de insumos e ingresar a la carpeta ***moralar-api,*** dentro de esa carpeta será la raíz del proyecto.

b. Ya teniendo en cuenta cuál es la raíz del proyecto, necesitaríamos revisar que dentro de esta carpeta sí se encuentra el archivo README.md, de esa forma sabemos que sí es la raíz del proyecto de la cuál es fundamental tener presente a la hora de ejecutar el proyecto en local.

1.2 Arquitectura básica del proyecto

En esta sección hablaremos de cómo se compone el proyecto para que les sea más fácil poder compilar y desplegar la api.



1. **Data:** Capa modelo que contiene las entidades.
2. **Domain:** Contiene servicios reutilizables y clases que utilizamos como modelos pero estas clases sirven para mapear de entidad a objetos simples.
3. **Repository:** Capa que sirve como mapeador de tabla para que sea fácil conectarnos a esa tabla por medio de un repositorio y poder acceder a los datos.
4. **UtilityFramework:** Es una librería interna que trae un sin fin de funcionalidades reutilizables de todo tipo.
5. **MongoDb:** sirve para hacer conexión a la base de datos y contener configuraciones personalizadas de la conexión.
6. **WebApi:** Es la capa principal del proyecto, siendo el controlador que sirve la información al consumidor del servicio. También se encarga de manejar la seguridad de la api, controla qué roles pueden acceder a qué servicio y mucho más.

2. Cómo ejecutar el programa en local

En esta sección se hablará de cómo poder ejecutar nuestra aplicación de angular desde una máquina local de windows, teniendo en cuenta que debe cumplir con los conocimientos previos (guía básica).

2.1 Requisitos antes de la instalación

Se debe tener instalado o configurado estas herramientas o programas para poder compilar y desplegar de forma correcta.

a. **Visual Studio:** Cualquier versión que soporte .net8 descargar
b. **Visual Studio Installer:** El visual studio debe estar habilitado para que acepte manejo de una web Api (guía básica)
c. Tener a la mano las credenciales del SMTP que querrán que envíe correos, como el host, puerto, si maneja ssl, el correo y contraseña (guía de gmail).
d. Previamente se tendrá que haber configurado el google maps api-key y que esté habilitado la librería 'Places' dentro del google maps api manager (guía).

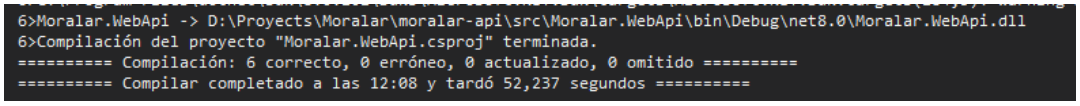
2.2 Conocimientos previos

1. Tener conocimiento básico en manejar y compilar visual studio para web-apps (guía básica).
2. Tener conocimiento básico en hacer publish en visual studio (guía básica).
3. Tener conocimiento básico en cómo desplegar un webApp a azure (guía).
4. Tener permisos para acceder la web-app de azure y poder hacer algunas configuraciones que veremos más adelante (Portal azure).
5. Saber reconocer los errores de consola que sean tipo CORS, para poder ir al módulo de cors en el portal azure y poder agregar el dominio que presentaba los errores de cors.
6. Tener permisos para acceder al mongoDb cluster para que puedas obtener los connectionString y también darle permisos de CORS a nuestra api.

3. Compilación del proyecto

3.1 Compilar en modo producción.

- Paso 1: Desde visual studio hacer desde la pestaña Compilar las siguientes acciones:
    - Limpiar solución
    - Recompilar solución
    - Compilar solución
- Paso final: Chequear que al final de compilar en la salida salga que todo salió correcto en el compilado.



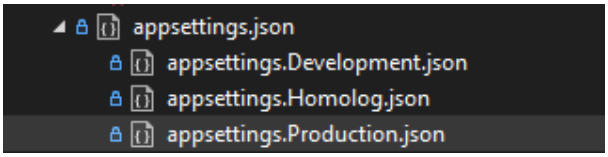
4. Cómo ejecutar el programa en local

4.1 Configuración de las variables de entorno

Es crucial que antes de compilar y desplegar sea necesario revisar que las variables de entorno sí están bien configuradas para que todo funcione bien.

- Paso 1: Ingresar a la raíz del proyecto ***moralar-api***
- Paso 2: Ir al paquete llamado **Moralar.WeApi** y allí buscar el archivo llamado ***appsettings.json***, allí encontrarás bastantes variables de entorno que debemos modificar.

    Nota: Explicaremos solo para un ***appsettings.json*** pero deberás revisar que este archivo tiene sub-archivos que según el ambiente que manejes deberás hacer modificaciones a ambos.



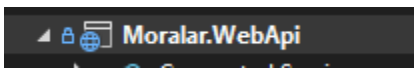
- Paso 3: Ingresa al archivo y reemplaza las variables que tengan corchetes por el valor real.
    - ConfigDev (Todas las variables dentro de esa sección)
    - Config (Todas las variables dentro de esa sección)
    - DATABASE (Todas las variables dentro de esa sección)

- Paso 4: Asegurarse de que no quede ninguna variable sin configurar y luego guardar los cambios correctamente.
- Paso 5: Buscar el archivo llamado ***Config.json*** que se encuentra dentro de la carpeta ***Settings***, allí encontrarás bastantes variables de entorno que debemos modificar.
- Paso 6: Ingresa al archivo y reemplaza las variables que tengan corchetes por el valor real.
    - googleMapsKey
    - suportEmail
    - SMTP (Todas las variables dentro de esa sección)
- Paso final: Asegurarse de que no quede ninguna variable sin configurar y luego guardar los cambios correctamente.

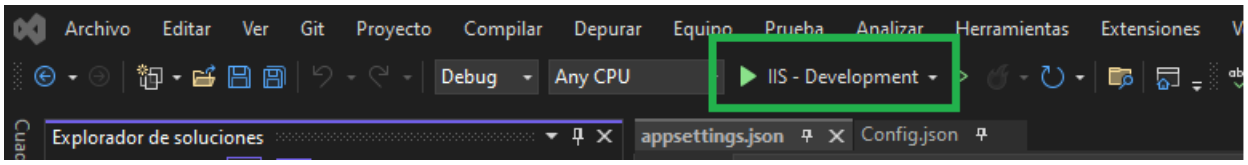
4.2 Ejecutar el aplicativo desde local

Es crucial que antes de compilar y desplegar sea necesario revisar que las variables de entorno sí están bien configuradas para que todo funcione bien.

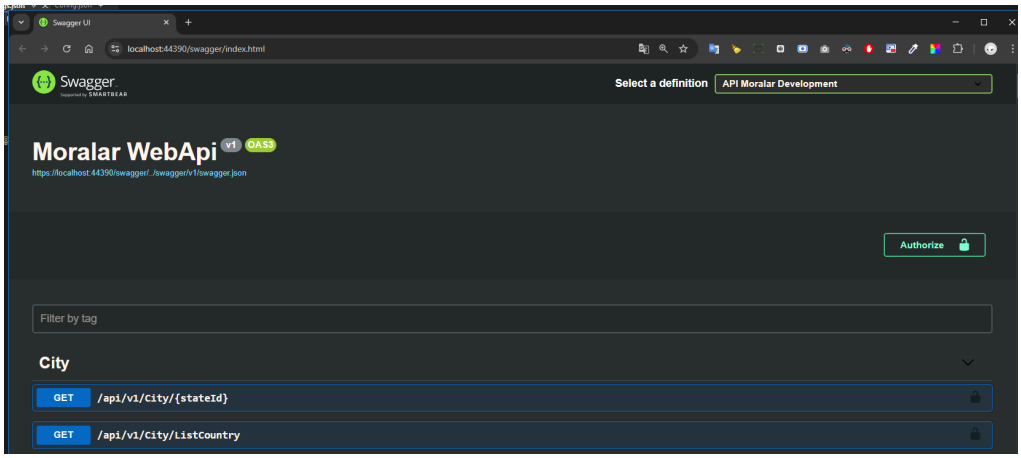
- Paso 1: Ingresar a la raíz del proyecto ***moralar-api***.
- Paso 2: Hacer clic derecho en el paquete **Moralar.WebApi** y asegurarse que el paquete sea el proyecto de inicio.



- Paso 3: Compilar el proyecto si no lo has hecho.
- Paso 4: Ejecutar el perfil **IIS -Development**



- Paso final: Podrás ver que ya estás en modo debugger y que podrás probar tus cambios desde la máquina local.



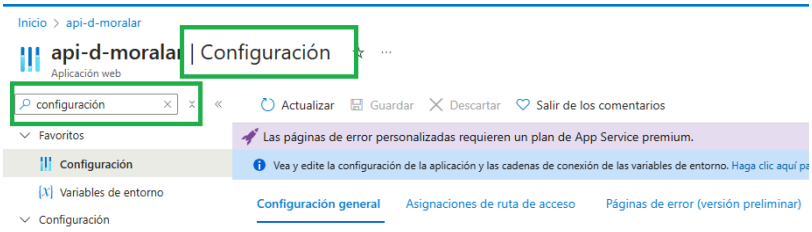
5. Despliegue de la api

5.1 Configurar la aplicación web en azure

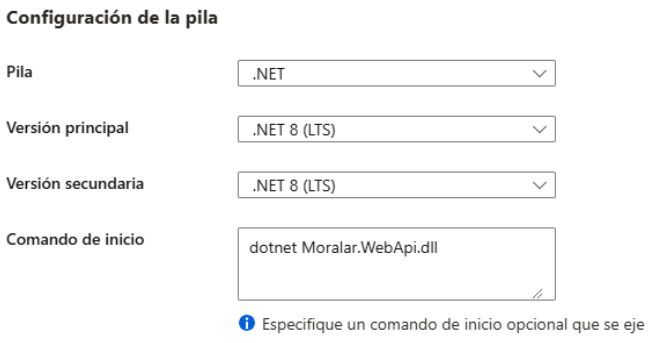
Antes de poder desplegar, queremos que el ambiente de azure sea apto para que la api de Moralar funcione. Solo es necesario hacer esta configuración una sola vez. Sigue estás instrucciones que sirven para que los aplicativos web y móvil puedan acceder a nuestra api sin restrinciones.

A. Chequear la configuración de la aplicación web

- Paso 1: Ingresar a la sección de **Configuración -> Configuración general** en azure.



- Paso 2: Verificar que las siguientes configuraciones estén similares:
  - Configuración de pila:

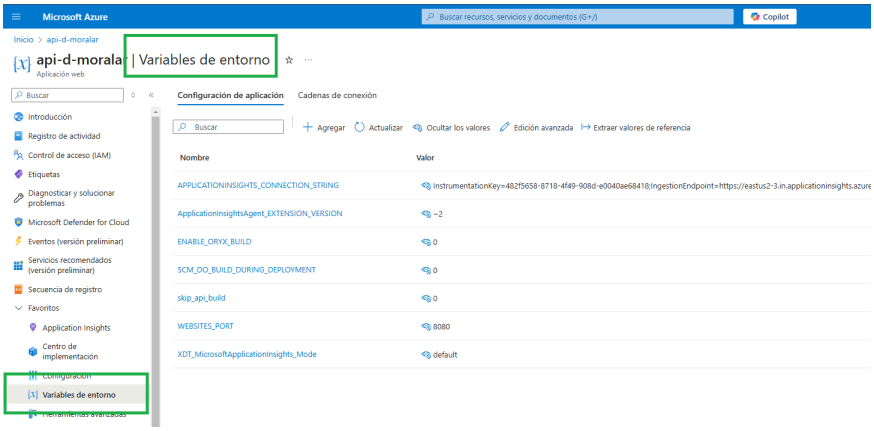


- Configuración de plataforma:

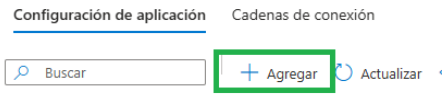
- Paso final: Verificar que si quede guardado la configuración de forma correcta.

B. Configurar variables de entorno en azure

- Paso 1: Ingresar a la sección de **Variables de entorno** en azure.



- Paso 2: Añadir estas opciones dado el caso que no las tenga:

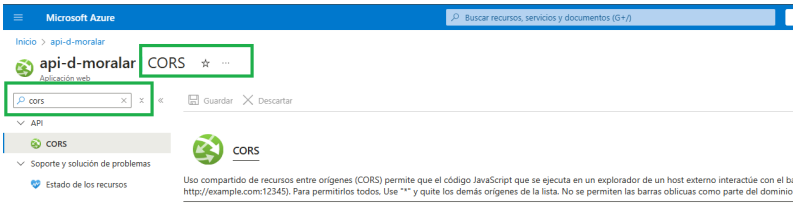


  - ENABLE_ORYX_BUILD: 0
  - SCM_DO_BUILD_DURING_DEPLOYMENT: 0
  - skip_api_build: 0
  - WEBSITES_PORT: 8080

- Paso final: Chequear que sí quede guardado la configuración.

C. Configurar Cors

- Paso 1: Ingresar a la sección de **CORS** en azure.



- Paso 2: Ir al apartado de **Orígenes permitidos** y agregar los siguientes valores y reemplazar las variables de corchetes por los valores reales:
    - http://localhost:4200
    - http://localhost
    - *
    - {url del sitio web administrativo de moralar}
    - {ip del cluster de mongoDb donde se encuentra la base de datos de moralar}
    - {https://{usuario-cluster-mongo-db}:{contraseña}@{host-del-closter} ejemplo: "https://nicolasviviescas:AZdKh1p4U9xyudm5@cluster-sinapsis.eegug.mongodb.net"

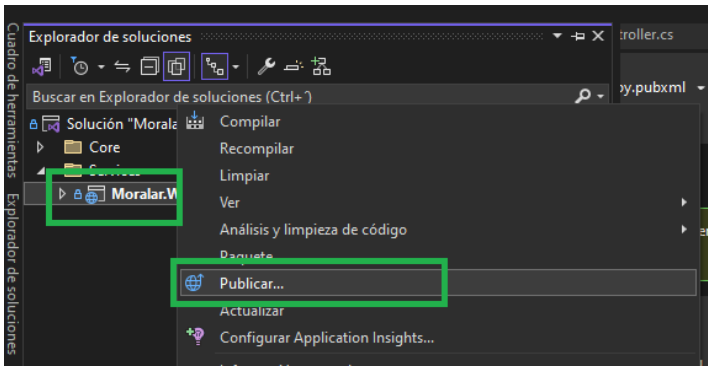
Nota: la más importante es que esté el asterisco porque el cluster va a estar cambiando las ip y no queremos que nos falle la conexión a la base de datos desde la api.

- Paso final: Asegurarse de guardar los valores correctamente.

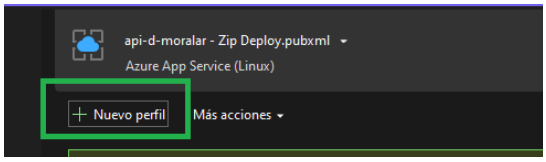
5.2 Desplegar a la nube de azure

Ya teniendo compilado nuestro aplicativo y de haber configurado bien las variables de entorno dentro de nuestro proyecto, ya podríamos hacer despliegue de nuestros cambios a la nube.

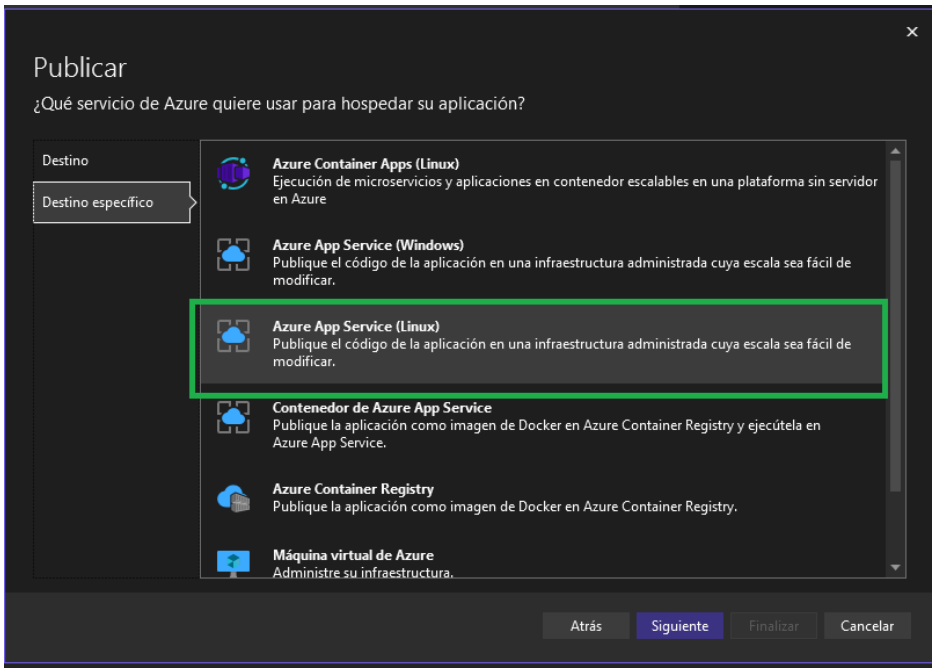
- Paso 1: Estar dentro de la raíz del proyecto en visual studio, luego hacer clic derecho al paquete **Moralar.WebApi ->** Publicar.



- Paso 2: Vamos a añadir un nuevo perfil en caso de que no tengamos configurado hacia donde desplegar desde el publicar.

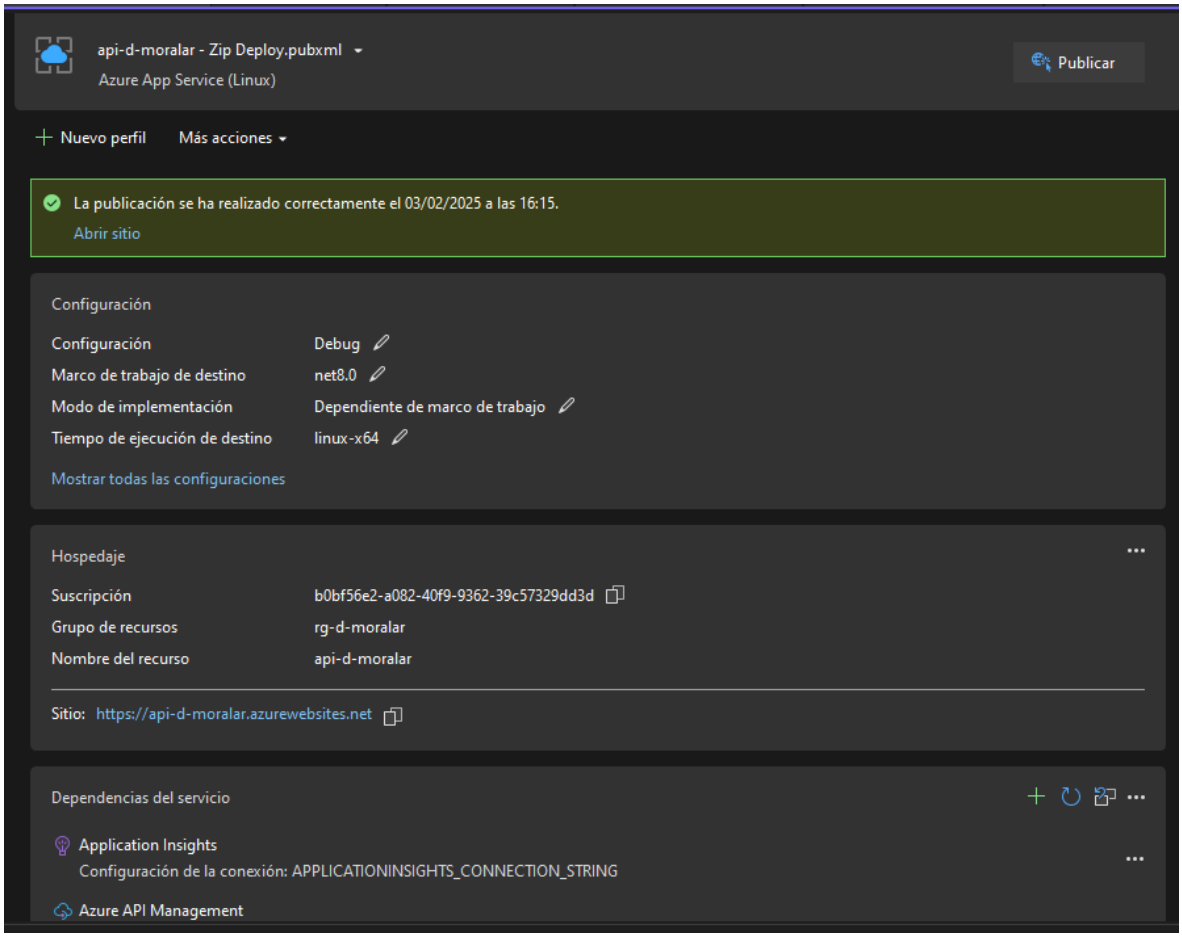


- Paso 3: Seleccionar la opción de azure, luego siguiente, después escoger esta opción dependiendo si el servidor del app service es linux o windows:

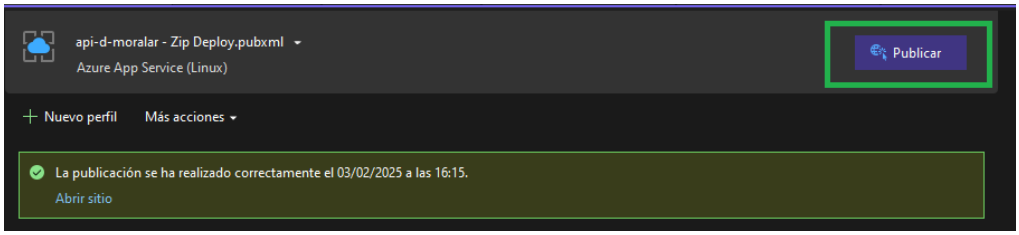


- Paso 4: Iniciar sesión con tus credenciales de azure y seleccionar tu app service al que quieres desplegar.

- Paso 5: Das finalizar configuración y ya estarías listo para poder publicar, se vería algo similar a esto.



- Paso 6: Hacer clic en el botón publicar, luego de finalizar te abrirá una ventana emergente del sitio.



- Paso final: Luego de que te lleva a esta página, le agregamos la ruta “swagger/index.html” algo como “{url de la api}/swagger/index.html” y ya con que te salga que utilizamos swagger es que todo salió muy bien.