# Documentación de la Arquitectura de la Aplicación

## Introducción

La aplicación está diseñada bajo una arquitectura basada en principios de \*\*Onion Architecture\*\* (Arquitectura en Cebolla), lo que permite una alta modularidad y separación de responsabilidades. Además, se implementaron principios SOLID para asegurar un código más flexible, mantenible y extensible. La aplicación está desarrollada en .NET y expone servicios RESTful para la gestión de máquinas virtuales.

## Componentes Principales de la Arquitectura

### Capa de Dominio

- Entidades: Representan los modelos centrales del negocio, como MaquinaVirtual.

- Interfaces: Definen contratos para repositorios y servicios, garantizando la independencia de la capa de aplicación respecto a implementaciones específicas.

- Lógica de Negocio: Reglas y validaciones específicas del dominio.

### Capa de Aplicación

- Servicios de Aplicación: Coordinan la ejecución de la lógica de negocio y la integración con otras capas.

- DTOs: Se utilizan para transferir datos entre la capa de aplicación y la presentación, asegurando que solo los datos necesarios se transmitan.

### Capa de Infraestructura

- Repositorios: Implementan las interfaces definidas en la capa de dominio utilizando Entity Framework y PostgreSQL.

- Autenticación y Seguridad: Implementa autenticación basada en tokens JWT.

- Configuración de la Base de Datos: Se realiza mediante DbContext configurado con Npgsql.

### Capa de Presentación

- Controladores RESTful: Utilizan ASP.NET Core para exponer servicios a través de endpoints.

- Swagger: Proporciona documentación interactiva para probar los servicios disponibles.

## Endpoints Principales

Los siguientes endpoints están disponibles para interactuar con la aplicación. A continuación, se detallan ejemplos y su funcionalidad:

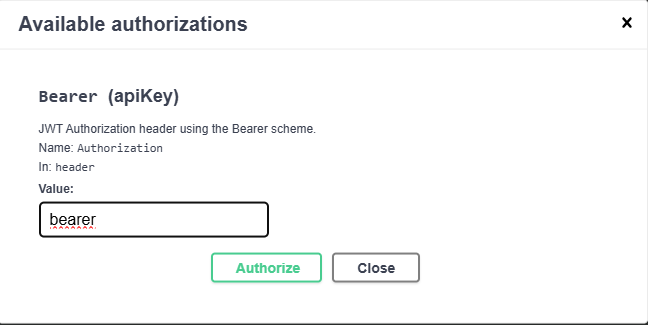
### ****Autenticación****

**Endpoint:** /Auth/login

* **Método:** POST
* **Descripción:** Permite obtener un token JWT para autenticarse en el sistema.
* **Ejemplo de Petición:**



**Nota: poner bearer antes del token**

****

### ****Obtener Máquina por ID****

**Endpoint:** /api/v1/MaquinaPorId

* **Método:** POST
* **Descripción:** Obtiene los detalles de una máquina virtual específica.
* **Ejemplo de Petición:**
* curl -X 'POST' \
* 'http://www.swmaquinas.somee.com/SWMAQUINA/api/v1/MaquinaPorId' \
* -H 'Authorization: bearer <token>' \
* -H 'Content-Type: application/json-patch+json' \

-d '{ "maquinatId": 1, "interactionId": "string" }'

### ****Crear Máquina Virtual****

**Endpoint:** /api/v1/CrearMaquinaVirtual

* **Método:** POST
* **Descripción:** Crea una nueva máquina virtual con los detalles proporcionados.
* **Ejemplo de Petición:**
* curl -X 'POST' \
* 'http://www.swmaquinas.somee.com/SWMAQUINA/api/v1/CrearMaquinaVirtual' \
* -H 'Authorization: bearer <token>' \
* -H 'Content-Type: application/json-patch+json' \

-d '{ "nombre": "string", "cores": 0, "ram": 0, "disco": 0, "sistemaOperativo": "string", "estado": "string", "interactionId": "string" }'

### ****Eliminar Máquina Virtual****

**Endpoint:** /api/v1/EliminarMaquinaVirtual

* **Método:** POST
* **Descripción:** Elimina una máquina virtual específica.
* **Ejemplo de Petición:**

curl -X 'POST' \

'http://www.swmaquinas.somee.com/SWMAQUINA/api/v1/EliminarMaquinaVirtual' \

-H 'Authorization: bearer <token>' \

-H 'Content-Type: application/json-patch+json' \

-d '{ "maquinatId": 0, "interactionId": "string" }'

### Usuarios para el Token JWT

La autenticación en la aplicación se realiza utilizando un token JWT que se obtiene mediante las credenciales de usuario. A continuación, se proporciona la lista de usuarios con sus respectivos correos electrónicos y contraseñas para realizar pruebas:

|  |  |
| --- | --- |
| Email | Password |
| admin@api.com | Admin123! |
| cliente1@api.com | Cliente123! |
| cliente2@api.com | Cliente123! |
| cliente3@api.com | Cliente123! |
| admin1@api.com | AdminExtra123! |

## Principales Clases y Métodos

### ****Controladores****

Los controladores se encuentran en la carpeta `Controllers` y gestionan la comunicación entre la aplicación y el cliente.  
- Ejemplo: `MaquinaController` maneja operaciones CRUD sobre máquinas virtuales.

### ****Servicios****

Los servicios implementan la lógica de aplicación.  
- Ejemplo: `MaquinaService` contiene las reglas de negocio para la creación y actualización de máquinas virtuales.

### ****Repositorios****

Los repositorios implementan la comunicación con la base de datos.  
- Ejemplo: `MaquinaRepository` utiliza Entity Framework para realizar operaciones CRUD en la base de datos.

## Tecnologías Utilizadas

- **Framework:** ASP.NET Core   
- **Base de Datos:** PostgreSQL con Entity Framework Core  
- **Autenticación:** Tokens JWT  
- **Documentación:**Swagger  
- **Principios:**Arquitectura en Cebolla, SOLID

## URL de Documentación Interactiva

La documentación interactiva de los endpoints está disponible en:  
[http://www.swmaquinas.somee.com/SWMAQUINA/swagger/index.html]

## Conclusión

La arquitectura basada en Onion y los principios SOLID garantizan que la aplicación sea mantenible y escalable. Los endpoints están diseñados para ser intuitivos y seguros, utilizando JWT para la autenticación. La integración con PostgreSQL y la utilización de Entity Framework facilitan la gestión de datos y aseguran un rendimiento óptimo.