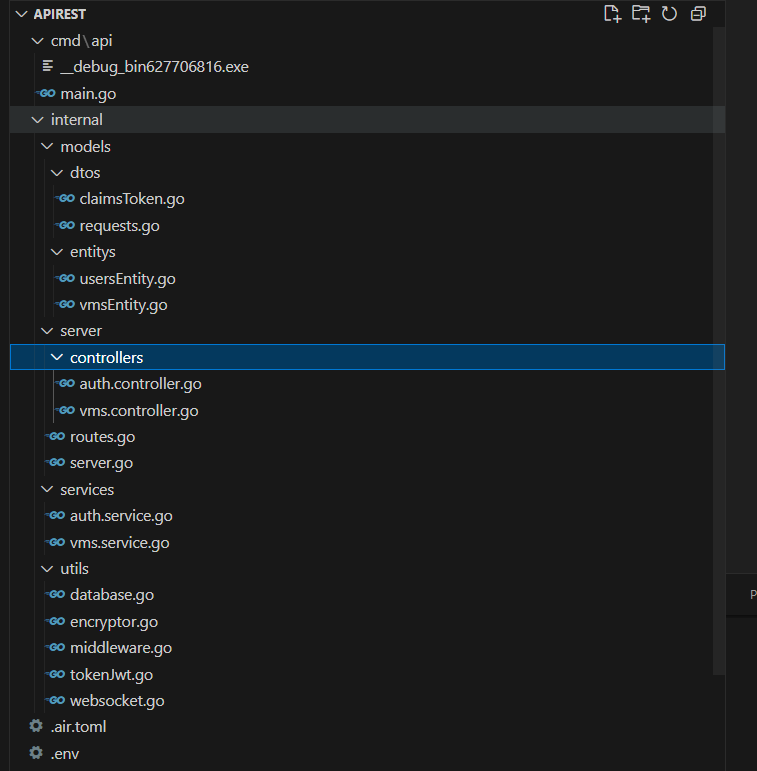
**Documentación API REST**

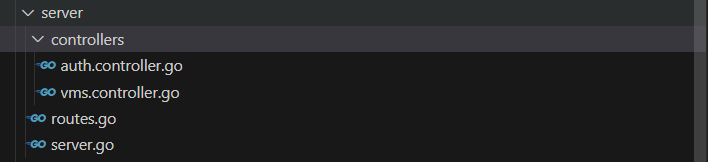
URL: [localhost:8080/](http://localhost:52395/api/usuarios)

* Api REST versión go1.24.1 “GOLANG”

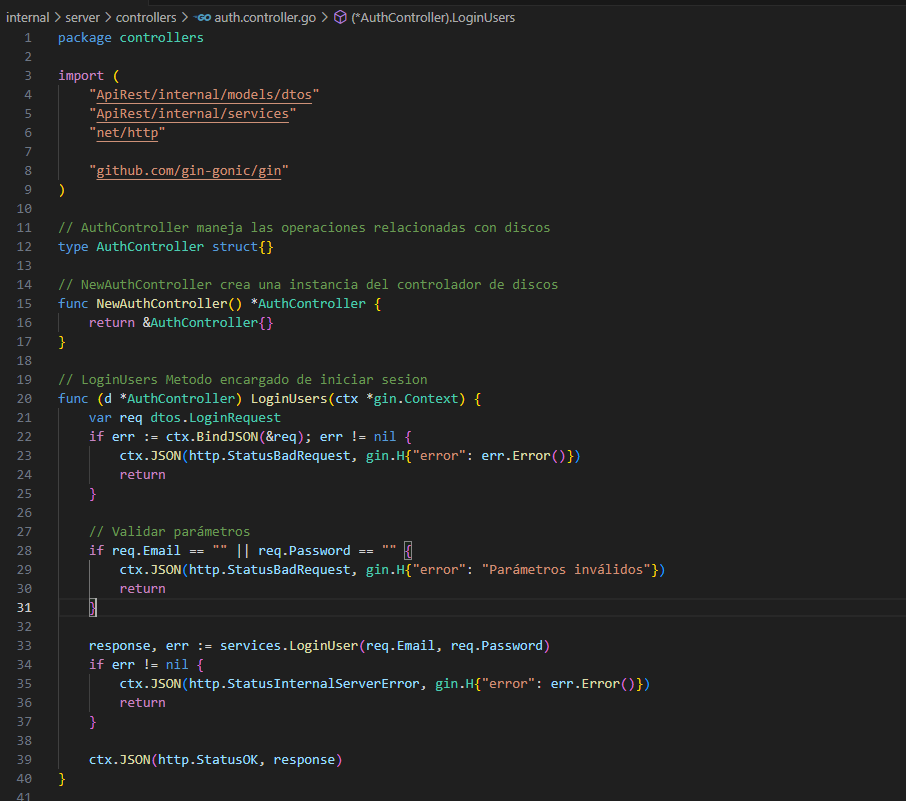
Estructura del proyecto



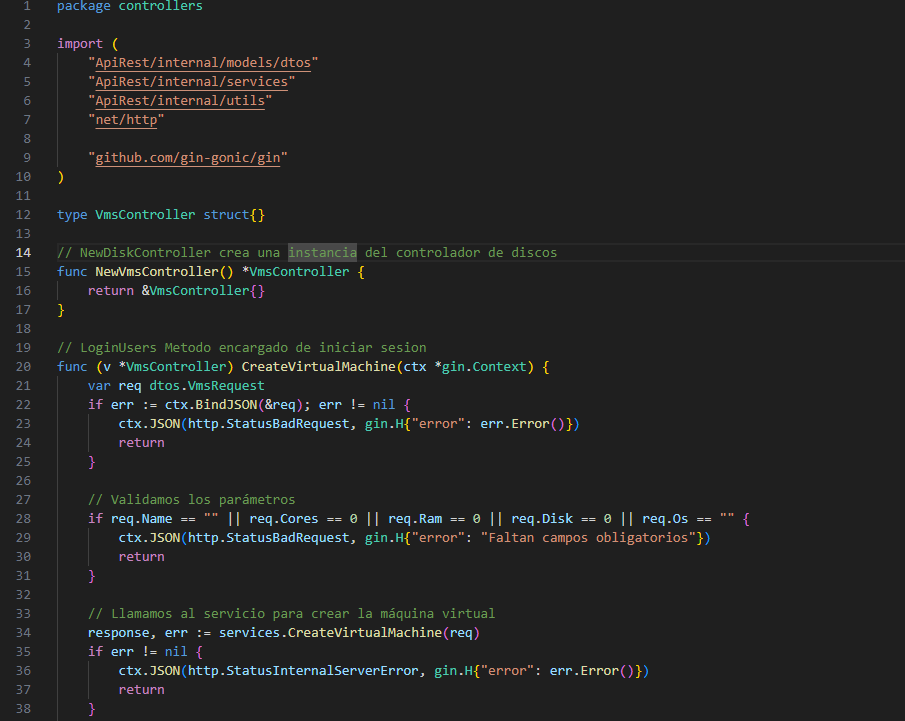
* **Controllers**



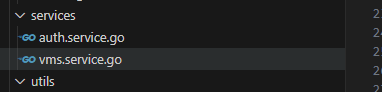
* auth.controller y vms.controller

Se crea el controlador correspondiente para verificar si el usuario existe y asi poder iniciar sesión y validar el TokenJWT

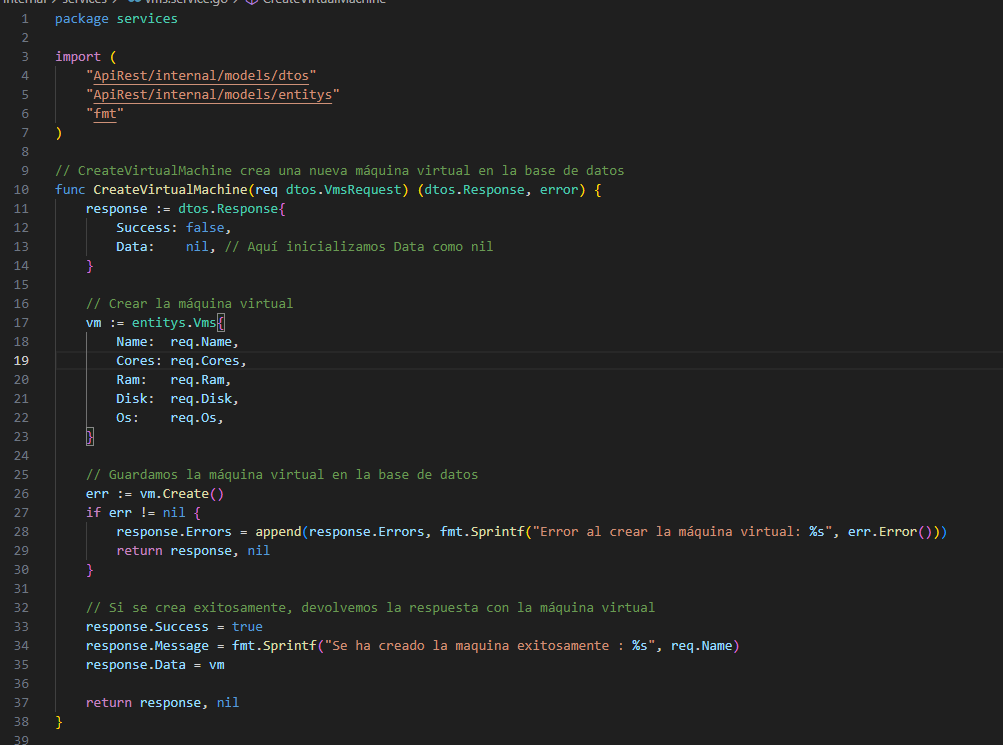
* Se implementa como extra la función de tokenJwt en la versión. “Se implementa en peticiones CRUD para poder validar que el usuario se encuentra autenticado” adicional el método de login ni los métodos GET y GETID tienen validación de token



* **Services**

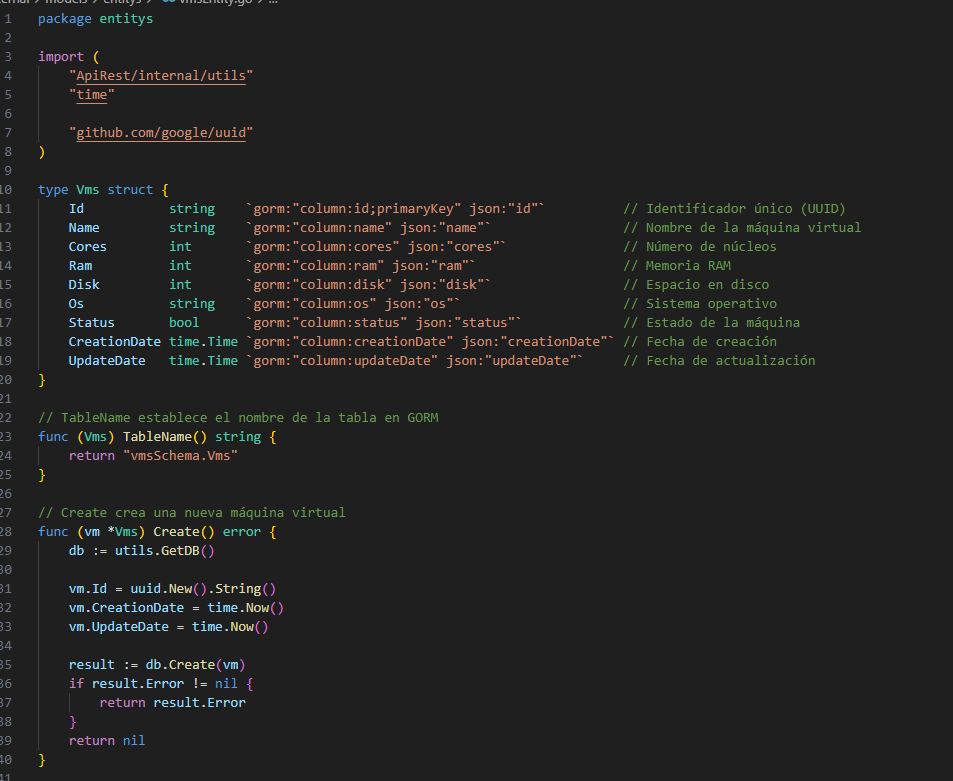


vms.service.go



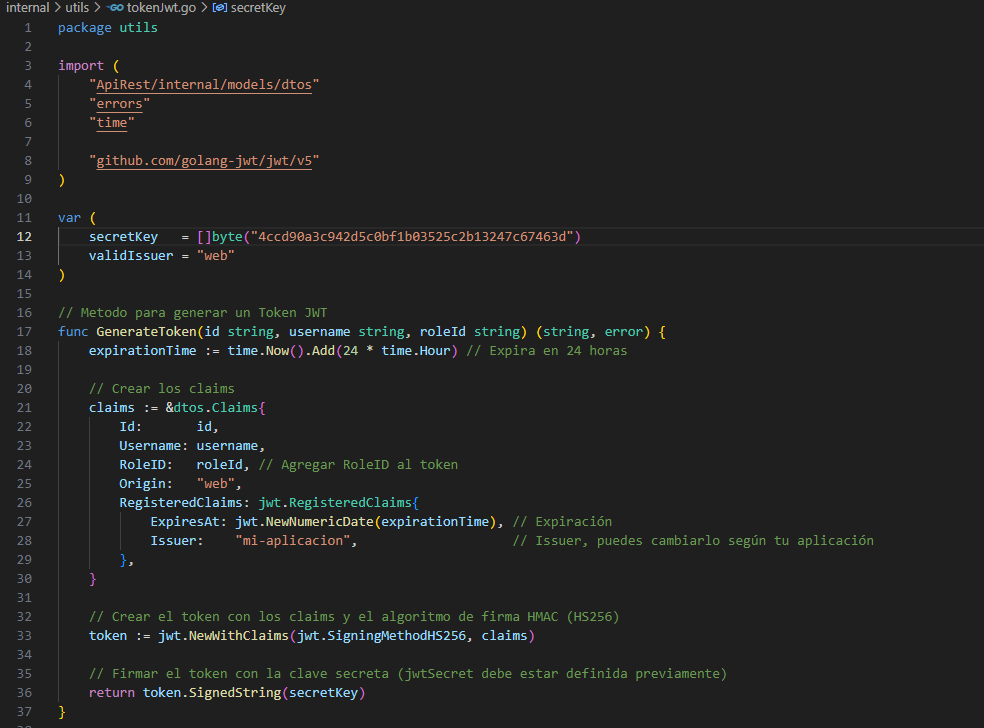
Se crean los correspondientes CRUD para el servicio VMS, como se puede observar se llama la interfaz para poder Crear, insertar eliminar o actualizar, adicional método de obtener todas las VMS y VM por id.

vmsEntity.go



Se utiliza la liberaría de “gorm” oficial, esta me permite conectar con la base de datos y la estructura de las tablas, mediante esto se realiza el CRUD necesario para poder gestionar las máquinas virtuales

tokenJwt.go



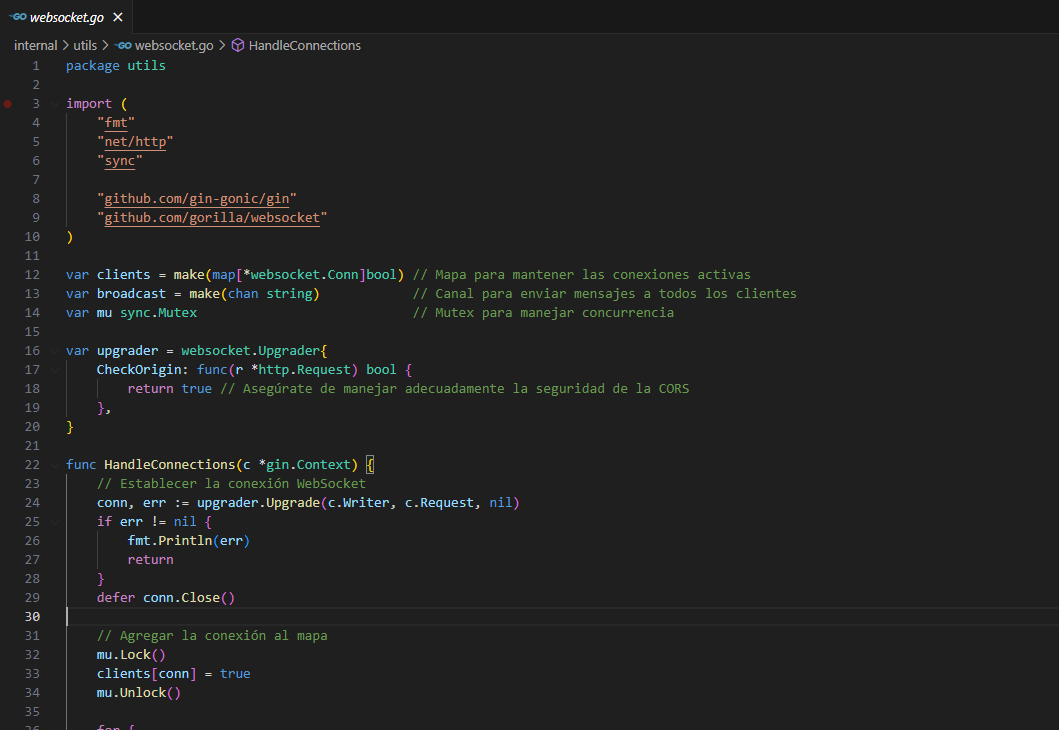
Método encargado de generar el token JWT, asigna las llaves de acceso y crea un token con una expiración definida

routes.go

Mediante este podemos crear todas las rutas de acceso al api Rest asi como las rutas y tipos de peticiones, adicional tenemos la validación de middleware para poder proteger las rutas que necesitan el token, y se agrega la petición que da inicio al canal de comunicación por WebSocket



Websocket.go



.env

Variables de entorno para conexiones a la DB y puertos locales, así como las demás que se necesiten

