UNIVERSIDAD CENTRAL DEL ECUADOR

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

PROGRAMACIÓN DISTRIBUIDA

Nombre: Steven Ortiz **Fecha:** 31/01/2022

Semestre: 8º **Paralelo:** 1

Documento de configuraciones

App-web

Config.java

```java
package com.programacion.config;

;
import ...


no usages  Steven-code1
@ApplicationScoped
public class Config {

    1 usage
    private final String BASE_URL="http://traefik";


    no usages  Steven-code1
    @ApplicationScoped
    @Produces
    public WebTarget webTarget(){
        Client client=ClientBuilder.newClient();
        return client.target(BASE_URL);
    }

}
```

Este código es una clase Java que está utilizando la especificación Java EE para crear un objeto configurable que permite hacer peticiones HTTP a un servidor con una dirección base específica, que se establece como "http://traefik".

WebServer.java

```
6 usages
private AuthorService authorService;
6 usages
private BookService bookService;

1 usage  Steven-code1
public WebServer(AuthorService authorService,
                 BookService bookService){

    this.authorService = authorService;
    this.bookService = bookService;
    port(8080);
    routes();
}
1 usage  Steven-code1
public void routes(){
    get( path: "/",(request, response) -> authorService.getAuthors(request,response));
    get( path: "/addAuthor",(request, response) -> authorService.showAddAuthor(request,response));
    post( path: "/addAuthor",(request, response) -> authorService.addAuthor(request,response));
    get( path: "/updateAuthor/:id",(request, response) -> authorService.showUpdateAuthor(request,response));
    get( path: "/deleteAuthor/:id",(request, response) -> authorService.deleteAuthor(request,response));
    get( path: "/books",(request, response) -> bookService.getBooks(request,response));
    get( path: "/addBook",(request, response) -> bookService.showAddBook(request,response));
    post( path: "/addBook",(request, response) -> bookService.addBook(request,response));
    get( path: "/updateBook/:id",(request, response) -> bookService.showUpdateBook(request,response));
    get( path: "/deleteBook/:id",(request, response) -> bookService.deleteBook(request,response));
```

Este código define una clase llamada WebServer que actúa como un servidor web. La clase tiene un constructor que recibe dos servicios, AuthorService y BookService, que se usan para manejar acciones relacionadas con los autores y los libros. La clase también tiene un método llamado routes() que establece las rutas URL para diferentes acciones como obtener autores, agregar autores, actualizar autores, eliminar autores, etc. Estas rutas están asociadas con métodos específicos en los servicios AuthorService y BookService. La clase usa la biblioteca Spark Java para crear el servidor web y establecer las rutas URL.

App-books

DbConfig.java

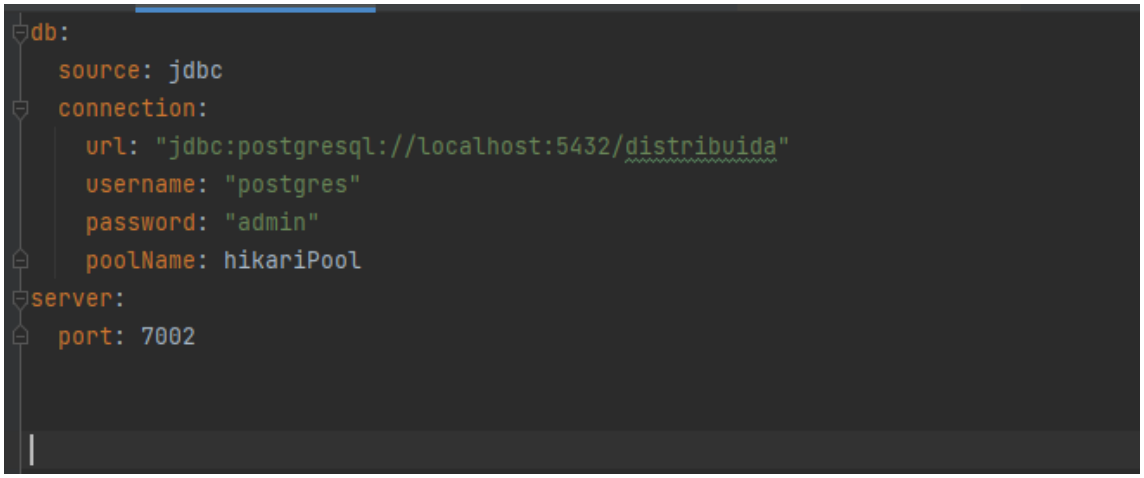
```
no usages  Steven-code1
@ApplicationScoped
public class DbConfig {

    no usages  Steven-code1
    @Produces
    @ApplicationScoped
    public DbClient dbClient(){
        Config config=Config.create();
        return DbClient.builder()
                .config(config.get("db"))
                .build();
    }


}
```

La función "dbClient" utiliza la clase Config para crear una configuración y luego utiliza esta configuración para construir una instancia de DbClient. La anotación @Produces indica que esta función produce un recurso compartido que puede ser inyectado en otras partes de la aplicación.

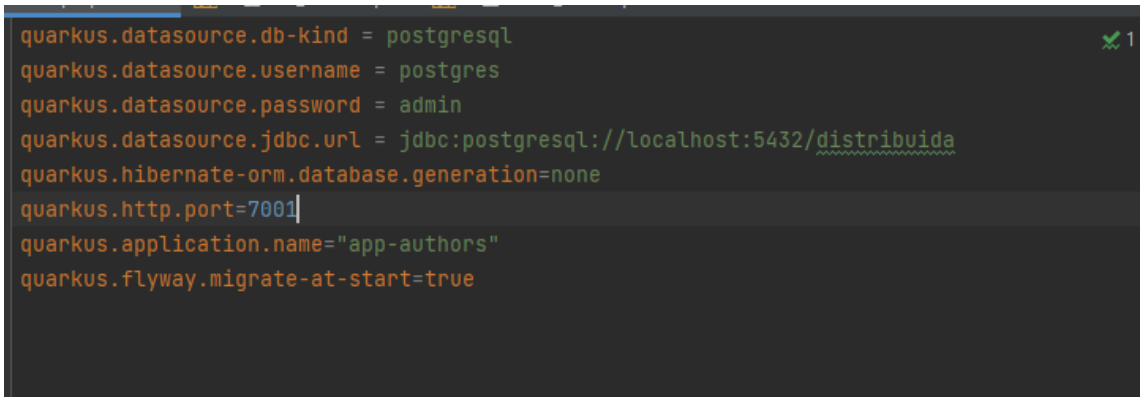
Aplicatio.yaml



Especifica los detalles de conexión para una base de datos PostgreSQL local, incluyendo la URL de la base de datos, el nombre de usuario y la contraseña para acceder a la base de datos. También especifica un nombre de pool de conexiones de base de datos "hikariPool" y el puerto del servidor "7002".

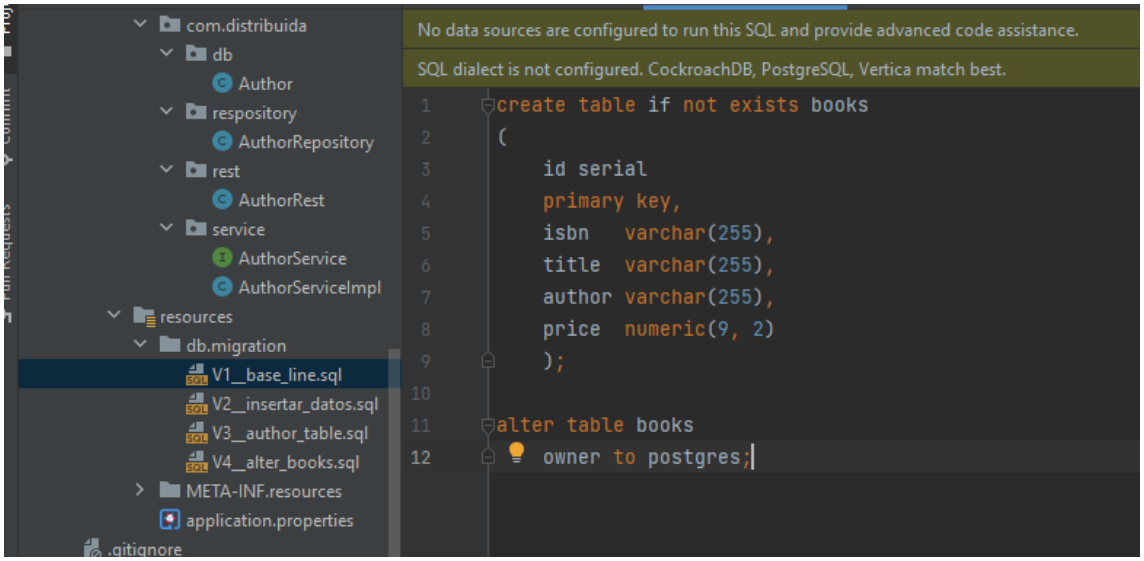
App-authors

Aplication.properties



Este es un archivo de configuración de la aplicación hecha con Quarkus. Contiene las configuraciones para la conexión a una base de datos PostgreSQL, el usuario y contraseña para conectarse, la URL de la base de datos, la configuración de Hibernate ORM, el puerto en el que se ejecutará la aplicación y el nombre de la aplicación. También incluye la configuración de Flyway para migrar las tablas en la base de datos en el inicio de la aplicación.

V1__base_line.sql



El archivo "V1__base_line.sql" es un archivo de migración que se utiliza para crear una tabla "books" y establecer el propietario de la tabla en "postgres" en una base de datos relacional.

La tabla "books" tiene los siguientes campos:

"id" que es una clave primaria y una columna autoincrementable.

"isbn" una columna de tipo varchar con una longitud máxima de 255 caracteres.

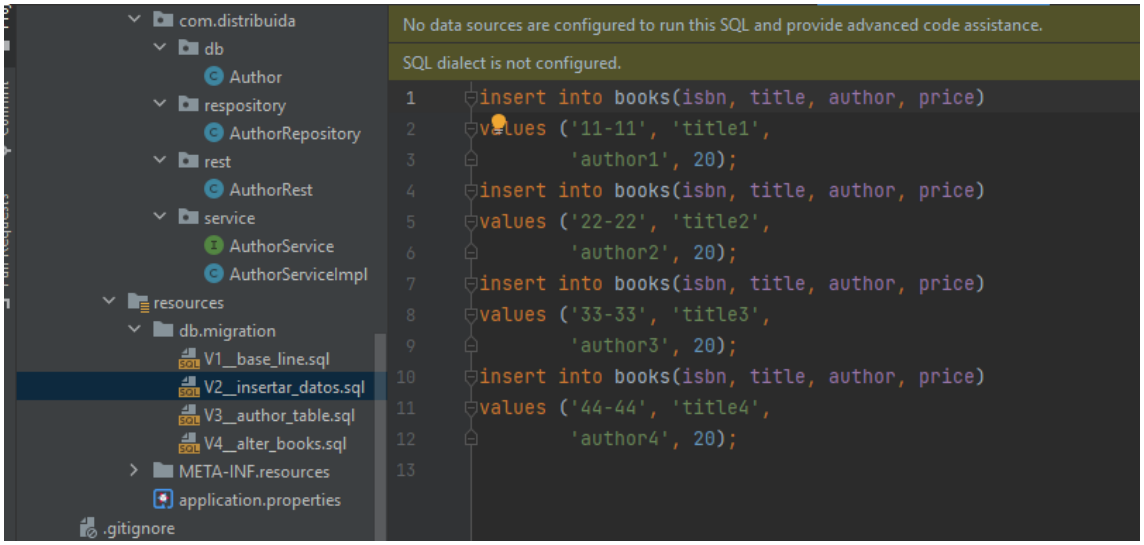
"title" una columna de tipo varchar con una longitud máxima de 255 caracteres.

"author" una columna de tipo varchar con una longitud máxima de 255 caracteres.

"price" una columna de tipo numérico con una precisión de 9 dígitos y 2 decimales.

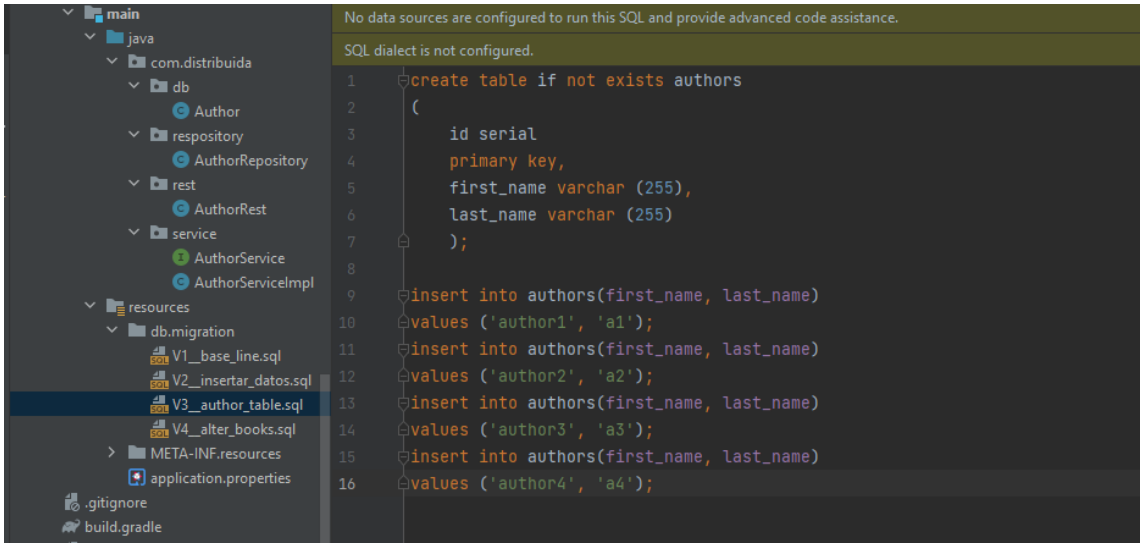
La migración se realiza utilizando Flyway.

V2__insertar_datos.sql



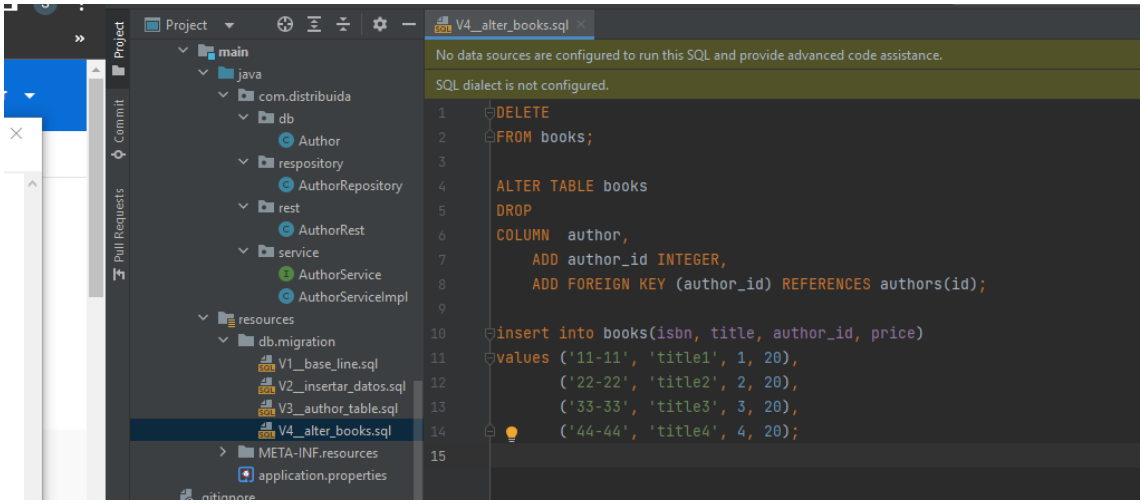
El archivo V2__insertar_datos.sql es un script de migración que agrega registros a la tabla "books". Este script agrega cuatro registros a la tabla con información de ISBN, título, autor y precio de los libros. La ejecución de este script de migración actualiza la base de datos y agrega los datos especificados.

V3__author_table.sql



El archivo V3__author_table.sql es un script de migración que crea la tabla “authors” en caso de no existir y agrega registros a la tabla "authors". Este script agrega cuatro registros a la tabla con información del id del autor, su nombre y su apellido.

V4__alter_books.sql

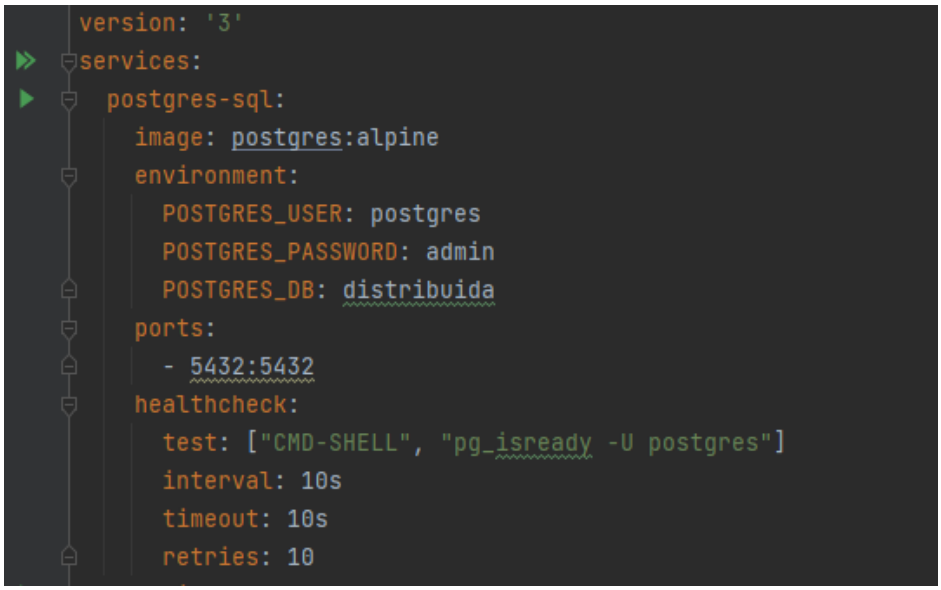


El script de migración V4__alter_books.sql modifica la estructura de la tabla de libros y la rellena con nuevos datos. Primero, elimina todos los registros existentes en la tabla books. Luego, se elimina la columna "author" y se agrega la columna "author_id". Además, se agrega una clave foránea que hace referencia a la tabla "authors" en el campo "id". Finalmente, se insertan nuevos registros en la tabla de libros con los valores correspondientes a "isbn", "title", "author_id" y "price".

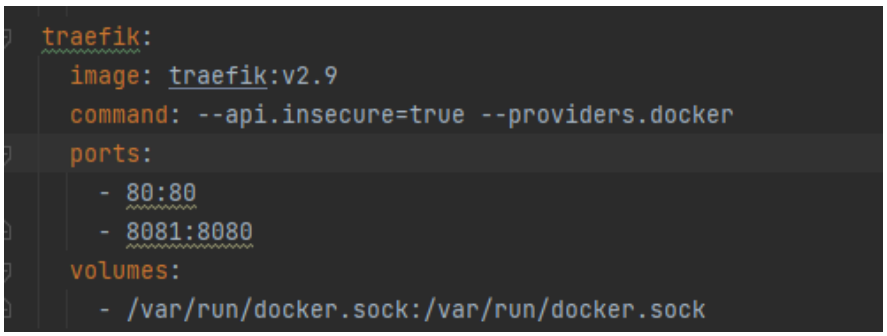
Docker

Docker-compose.yml

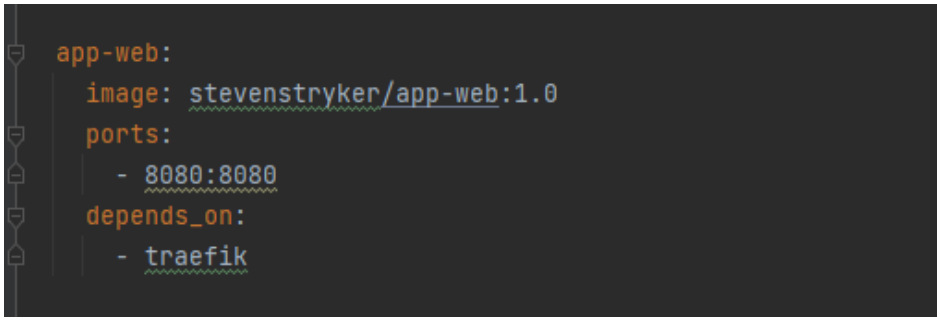
Este código es un archivo Docker Compose que describe una infraestructura compuesta por cuatro servicios: PostgreSQL, Traefik, una aplicación web (app-web), una aplicación de autores (app-authors) y una aplicación de libros (app-books).



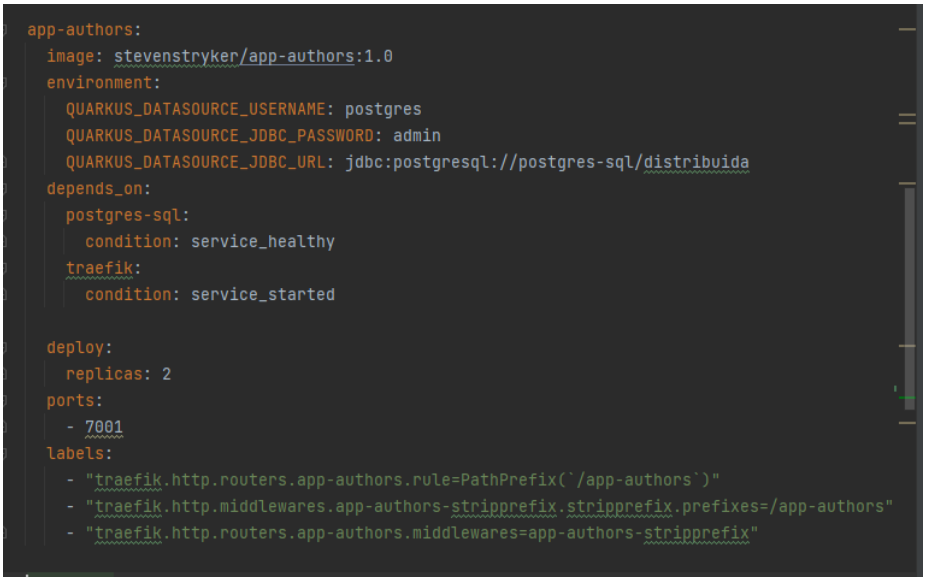
El servicio "postgres-sql" es un contenedor que ejecuta una imagen de PostgreSQL. Se establecen las variables de entorno para el usuario y la base de datos a utilizar, y se abre un puerto para permitir conexiones externas. También se establece una healthcheck para verificar que el contenedor esté funcionando correctamente.



El servicio "traefik" es un contenedor que ejecuta una imagen de Traefik, una plataforma de enrutamiento de proxy inverso. Se establecen las opciones de la línea de comandos para habilitar la API y la detección de proveedores, se abren dos puertos para permitir el acceso a la interfaz de usuario y a la API, y se monta el socket Docker para que Traefik pueda detectar los contenedores en ejecución.



El contenedor app-web utiliza una imagen llamada "stevenstryker/app-web:1.0" y expone el puerto 8080 del contenedor en el puerto 8080 del host. Depende del servicio "traefik".



El contenedor app-authors utiliza una imagen llamada "stevenstryker/app-authors:1.0". El entorno de este contenedor está configurado para utilizar una base de datos PostgreSQL con credenciales específicas. Depende del servicio "postgres-sql" y "traefik", con una condición de que ambos servicios deben estar saludables y arrancados antes de iniciar este contenedor.

```yaml
app-books:
  image: stevenstryker/app-books:1.0
  ports:
    - 7002
  deploy:
    replicas: 3
  environment:
    DB_CONNECTION_URL: jdbc:postgresql://postgres-sql/distribuida
    DB_CONNECTION_USERNAME: postgres
    DB_CONNECTION_PASSWORD: admin
  depends_on:
    traefik:
      condition: service_started
    postgres-sql:
      condition: service_started
  labels:
    - "traefik.http.routers.app-books.rule=PathPrefix(`/app-books`)"
    - "traefik.http.middlewares.app-books-stripprefix.stripprefix.prefixes=/app-books"
    - "traefik.http.routers.app-books.middlewares=app-books-stripprefix"
```

Este servicio utiliza la imagen "stevenstryker/app-books:1.0" y se asigna el puerto 7002. La cantidad de réplicas del servicio será de 3. Las variables de entorno DB_CONNECTION_URL, DB_CONNECTION_USERNAME y DB_CONNECTION_PASSWORD se usan para establecer la conexión con la base de datos. Los servicios "postgres-sql" y "traefik" son dependencias necesarias para que el servicio "app-books" funcione correctamente. Las etiquetas de "labels" se usan para configurar la ruta de acceso a la aplicación a través de Traefik, un proxy inverso.