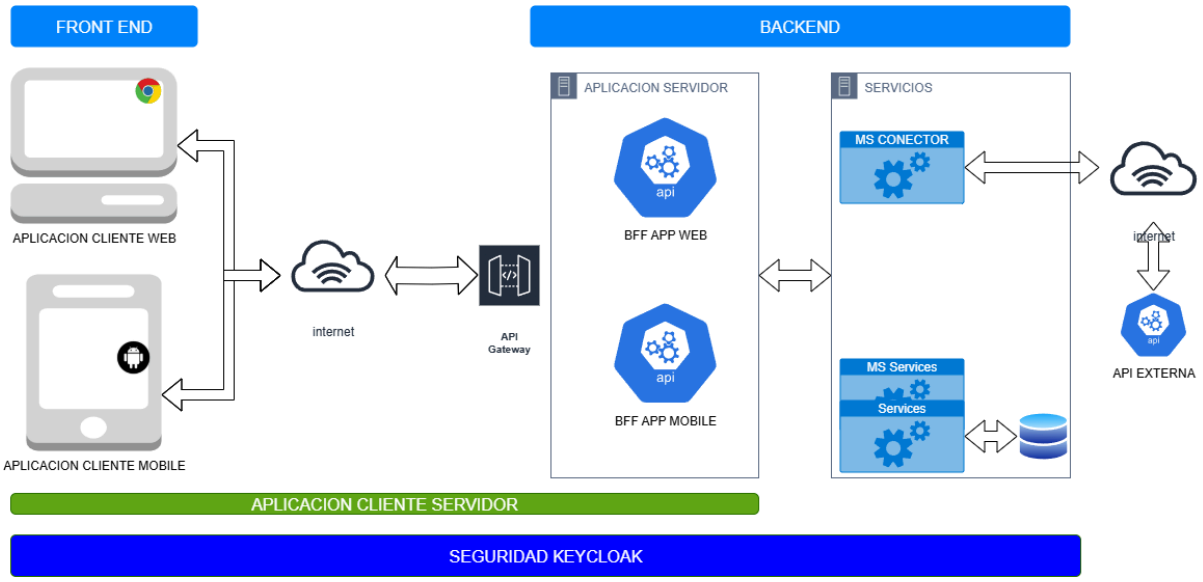
Arquitectura Cliente-Servidor



La arquitectura **cliente-servidor** es un modelo de diseño de software en el que las tareas se distribuyen entre dos componentes principales: el **cliente (solicita recursos y peticiones a servicios)** y el **servidor (Procesa y responde las peticiones)**

En nuestro esquema de referencia

Cliente: Front End

Servidor: Back End

Puntos clave de la arquitectura

1. **División de tareas**: Hay una clara separación de responsabilidades. El cliente se encarga de la interfaz de usuario y de la presentación de los datos, mientras que el servidor se encarga del procesamiento, el almacenamiento y la gestión de la lógica de negocio.
2. **Comunicación**: La interacción ocurre a través de un protocolo de comunicación (como HTTP, FTP, etc.).
3. **Centralización**: Los recursos, como las bases de datos y la lógica de negocio, están centralizados en el servidor.
4. **Escalabilidad**: Es relativamente fácil escalar la arquitectura. Se pueden añadir más clientes sin afectar al servidor o, si el servidor está sobrecargado, se puede mejorar su capacidad o añadir más servidores para manejar el aumento de la demanda.

Ejemplos comunes

- **Navegador y servidor web**
- **Aplicación de correo electrónico**: La aplicación en tu teléfono (cliente) se conecta a un servidor de correo para enviar y recibir mensajes.
- **Juegos en línea**: Tu consola o PC (cliente) se conecta a un servidor central para jugar con otras personas y para que se procese la lógica del juego.

Configuración de infraestructura de la arquitectura a aplicar en el proyecto

***Nota**: Se aplicará configuración de ejemplo utilizando sistema operativo windows,, quedando a criterio del alumno la utilización de linux y adaptación el ejemplo de configuración.*

1 Configuración de dominio Local. 127.0.0.1 dacs2025.local

DNS local para resolver el dominio de prueba
C:\Windows\System32\drivers\etc editar archivo hosts como administrador
agregar al final linea 127.0.0.1 dacs.local

Donde dacs.local puede ser el dominio local que prefieran para su proyecto

2 Configuración de Certificado SSL/TLS HTTPS para comunicación segura entre cliente y servidor

- **HTTPS** es el **protocolo de comunicación** seguro.
- **SSL/TLS** es el **certificado y la tecnología** de encriptación que permite que HTTPS funcione.

Para crear el certificado utilizaremos openssl y una serie de script que nos facilitaran la creación de certificado de extensión *.crt, la llave privada *.key y los archivos *.pem para ser configurados en el servidor web que implementaremos

- Se recomienda configurar en la variable de entorno PATH la ruta donde se encuentra el ejecutable openssl.exe
- Se puede instalar openssl siguiendo este how to https://www.ssldragon.com/es/how-to/openssl/install-opnessl-windows/

Los script que utilizaremos para crear los certificados son make-cert.bat para crear: este script les solicita el nombre de dominio a crear, por ejemplo dacs2025.local pueden completar los datos solicitados o dejar por defecto los que propones, (los valores propuestos se pueden configurar en cert.conf también provistos) se crearan server.crt y server.key
Luego utilizaremos export-to-pem.bat para crear los dos archivos *.pem que usaremos en nginx para configurar HTTPS estos archivos se deben copiar en la carpeta ssl que crearemos en el siguiente paso

3 Instalar Docker Desktop

recomendado crear cuenta y hacer login (utilizó SSO de gmail)
https://docs.docker.com/desktop/setup/install/windows-install/

4 Configurar nginx como servidor web

- Crear directorio donde tendremos el docker-composer D:\nginx-docker
- Crear archivo docker-compose.yml

```
services:
  web:
    image: nginx:latest
```

```
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
      - ./nginx/ssl:/etc/nginx/ssl
      - ./nginx/web:/usr/share/nginx/html
```

Crear subdirectorio nginx con subdirectorios web, ssl y archivo default.conf

```
server {
    listen 80;
    server_name sis.local;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;
    server_name sis.local;

    ssl_certificate /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;

    root /usr/share/nginx/html;
    index index.html;


       # Bloque 2: Gateway para la aplicación 'ms-test'
    # Las peticiones a https://misitio.dev/ms-test se redirigirán a
http://host.docker.internal:8989
    location ~ ^/ms-test(.*) {
             # Reescribe la URL, eliminando el prefijo /ms-test/
        # y agregando el nuevo prefijo /ms_seguridad_apis
        rewrite ^/ms-test/(.*)$ /ms_seguridad_apis/$1 break;

        proxy_pass http://host.docker.internal:9003;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    # Bloque 3: Gateway para la aplicación 'ms-prueba'
    # Las peticiones a https://misitio.dev/ms-prueba se redirigirán a
http://host.docker.internal:9090
    location /ms-prueba/ {
```

```
        rewrite ^/ms-prueba/(.*) /$1 break;
        proxy_pass http://host.docker.internal:9090;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

      # Bloque 1: Servir la aplicación de Angular desde la raíz del
dominio
    # Aquí Nginx buscará y servirá los archivos estáticos de la
aplicación Angular.
    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

}
```

Comandos para correr docker

Abrir la consola CMD y navegar al directorio creado con los archivos y carpetas antes mencionados. posicionarse donde esta el archivo docker-compose.yml

docker-compose up -d (arranca ya después se puede parar y arrancar desde interfaz de doker -desktop)

docker-compose down --volumes --rmi all (Para eliminar la imagen y reiniciar nuevamente desde cero)

docker-compose logs web (Para ver logs de ngnix, tambien se pueden ver desde docker desktop)

Crear un archivo index.html y cpiarlo en la carpeta “web”. Verificar funcionamiento index.html y https

```
<html>
<head></head>
<body>
<p>HOLA DESDE NGINX!!</p>
</body>
</html>
```

5 Instalar SDK JAVA

- Descargar segun su PC y sistema operativo JDK 21
  https://www.openlogic.com/openjdk-downloads
- Instalar
- Configurar variables de entorno de sistema, Configurar JAVA_HOME y PATH o verificar esté configurada. Ejemplo
  JAVA_HOME = C:\Program Files\OpenLogic\jdk-8.0.452.09-hotspot
  en PATH, %JAVA_HOME%\bin

- Verificar en consola CMD java -version y javac -version

6 Instalar certificado en JVM para que java confíe en nuestro certificado autofirmado

En los script para crear los certificados existe un readme donde se indica el comando para importar el certificado server.crt a la máquina virtual de java

7 Instalar NODE/NPM , AngularCli y Cliente GIT

https://git--scm-com.translate.goog/downloads/win?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

NODE/NPM

https://nodejs.org/en/download

ANGULAR CLI

```
npm install -g @angular/cli
ng version
```

Los pasos siguientes los vamos a ir desarrollando y sumando el documento

8 Instalar Postgres y PGAdmin4 con docker

9 Instalar keycloack con docker

10 Instalar visual studio code para front end

11 Instalar STS (eclipse) o intelliJ para backend