Sistemas Embebidos

Trabajo práctico Nº 3

Sistemas Operativos II
DC - FCEFyN - UNC

Mayo de 2020

1 Introducción

Los *sistemas embebidos* suelen ser accedidos de manera remota, una forma común, suelen ser las *RESTful APIs*. Estas, brindan una interfaz definida y robusta para la comunicación y manipulación del *sistema embebido* de manera remota. Definidas para un esquema *Cliente-Servidor* se utilizan en todas las verticales de la industria tecnológica, desde aplicaciones de *IoT* hasta juegos multijugador.

2 Objetivo

El objetivo del presente trabajo practico es que el estudiante tenga un visión *end to end* de una implementación básica de una *RESTful API* sobre un *sistema enbedido*. El estudiante deberá implementarlo interactuando con todas las capas del procesos. Desde el *testing* funcional (alto nivel) hasta el código en C del servicio (bajo nivel).

3 Desarrollo

3.1 Requerimientos

Para realizar el presente trabajo practico, es necesario una computadora con *kernel* Linux, ya que usaremos *SystemD* para implementar el manejo de nuestro servicios.

3.2 Desarrollo

Se deberá implementar dos servicios en lenguaje C, estos son el *servicio de usuarios* y el *servicio de status*. Ambos servicios, deberán exponer una *REST API*. Con el objetivo de acelerar el proceso de desarrollo vamos a utilizar un *framework*: se propone utilizar https://github.com/babelouest/ulfius. El estudiante puede seleccionar otro, justificando la selección, o implementar el propio

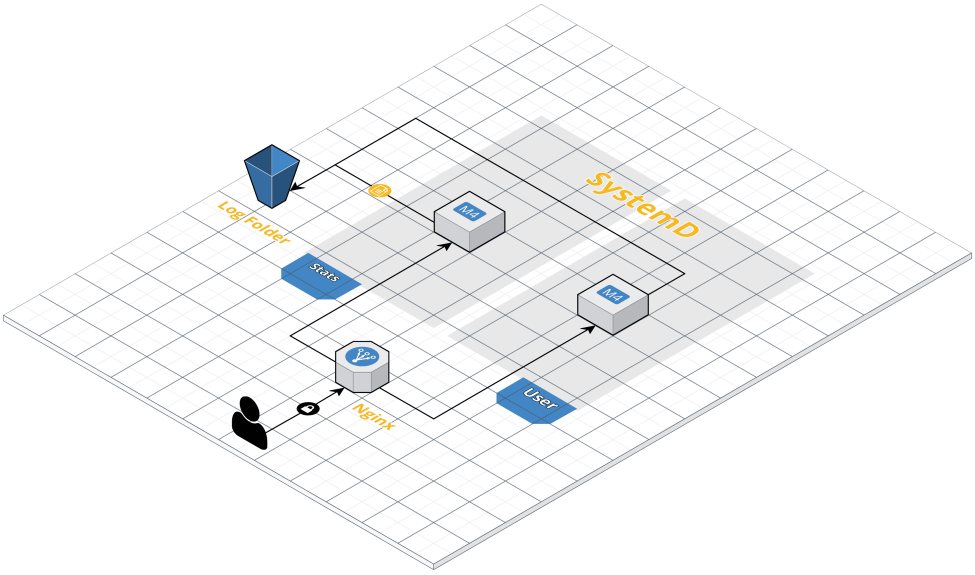

Figure 1: Arquitectura del sistema

(no recomendado). El servicio debe tener configurado un *nginx* por delante para poder direccionar el *request* al servicio correspondiente. El web server, deberá autenticar el *request* por medio de de un usuario y password enviado en el *request*, definido donde el estudiante crea conveniente. Las credenciales no deberán ser enviadas a los servicios.

El web server deberá retornar *404 Not Found* para cualquier otro *path* no existente.

A modo de simplificación, usaremos sólo *HTTP*, pero aclarando que esto posee **grabes problemas de seguridad**. Ambos servicios deben estar configurados con *SystemD* para soportar los comandos, *restart*, *reload*, *stop*, *start* y deberán ser inicializados de manera automática cuando el sistema operativo *botee*.

Ambos servicios deber *logear* todas sus peticiones con el siguiente formato:

<Timestamp>|<Nombre Del Servicio >| <Mensaje>

El `<Mensaje>` sera definido por cada una de las acciones de los servicios.

El gráfico 3.2 se describe la arquitectura requerida.

A continuación, detallaremos los dos servicios a crear y las funcionalidades de cada uno.

3.3 Servicio de Usuarios

Este servicio se encargará de crear usuarios y listarlos. Estos usuarios deberán poder *logearse* vía *SSH* luego de su creación.

3.3.1 POST /api/users

Endpoints para la creación de usuario en el sistema operativo:

POST http://{{server}}/api/users

Request

```
curl --request POST \
    --url http:// {server}}/api/users \
    -u USER:SECRET \
    --header 'accept: application/json' \
    --header 'content-type: application/json' \
    --data '{"username": "myuser", "password": "mypassword"}'
```

Respuesta

```
{
    "id": 142,
    "username": "myuser",
    "created_at": "2019-06-22 02:19:59"
}
```

El `<Mensaje>` para el log será: `Usuario <Id> creado`

3.3.2 GET /api/users

Endpoint para obtener todos los usuario del sistema operativo y sus identificadores.

```
curl --request GET \
    --url http://{{server}}/api/users \
    -u USER:SECRET \
    --header 'accept: application/json' \
    --header 'content-type: application/json'
```

Respuesta

```
{
  "data": [
      {
          "user_id": 2,
          "username": "user1",
      },
      {
          "user_id": 1,
```

```
            "username": "user2"
        },
        ...
    ]
}
```

El `<Mensaje>` para el log será: `Usuario listados: <cantidad de usuario del SO>`

3.4 Servicio de estado

Este servicio devolverá el estado actual del sistema

REQUEST /api/servers/hardwareinfo

```
curl --request GET \
    --url http://{{server}}/api/servers/hardwareinfo \
    -u USER:SECRET \
    --header 'accept: application/json' \
    --header 'content-type: application/json'
```

Respuesta

```
{
    "kernelVersion": "4.4.0-87-generic",
    "processorName": "Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz",
    "totalCPUCore": 2,
    "totalMemory": 0.9520301818847656,
    "freeMemory": 0.3449363708496094,
    "diskTotal": 9.376751616,
    "diskFree": 5.573582848,
    "loadAvg": 0,
    "uptime": "74h 54m 7s"
}
```

El `<Mensaje>` para el log será: `Estadísticas requeridas desde el <host>`. El `<host>` aquí, debe ser la dirección IP del cliente que realice el *request*. Para obtenerlo, se puede pasar por parámetro desde web server al servicio mediante un *header*.

4 Entrega

Se deberá proveer los archivos fuente, así como cualquier otro archivo asociado a la compilación, archivos de proyecto "Makefile" y el código correctamente documentado, todo en el repositorio de `https://classroom.github.com/a/8GnemVno`, donde le Estudiante debe demostrar avances semana a semana mediante *commits*.

También se debe entregar un informe, con el formato adjunto. El informe además debe contener el diseño de la solución y la comparativa de profilers. Se debe asumir que las pruebas de compilación se realizarán en un equipo que

cuenta con las herramientas típicas de consola para el desarrollo de programas (Ejemplo: gcc, make), y NO se cuenta con herramientas "GUI" para la compilación de los mismos (Ej: eclipse).

El install del makefile deberá copiar los archivos de configuración de systemd para poder luego ser habilitados y ejecutados por linea de comando. El script debe copiar los archivos necesarios para el servicio Nginx systemd para poder luego ser habilitados y ejecutados por linea de comando. Los servicios deberán pasar una batería de test escritas en *postman* provistas. TBD.

5 Evaluación

El presente trabajo práctico es individual deberá entregarse antes del jueves 4 de junio de 2020 a las 23:55 mediante el LEV. Será corregido y luego deberá coordinar una fecha para la defensa oral del mismo.

6 Referencias y ayudas

- `https://www.freedesktop.org/wiki/Software/systemd/`
- `https://docs.nginx.com/`
- `https://github.com/babelouest/ulfius`
- `https://kore.io/`