



DOCUMENTACIÓN PRÁCTICA SOA

- Tutor Virtual Inteligente -

Carlos Martín García
Iván Mateos Arias
Silvia Río Río
Oliver Urones García

4º GIISI
EPSZ - USAL



ÍNDICE

ENUNCIADO.....	2
IMPLEMENTACIÓN.....	2
DIAGRAMA ENTIDAD/RELACIÓN DE LA BASE DE DATOS	2
DIAGRAMA DE LOS SERVICIOS	3
PETICIONES POR POST	3
PETICIONES POR GET.....	3
DESCRIPCIÓN DE LOS SERVICIOS	4
ESB.PHP	4
LOGIN.PHP	4
MATERIA.PHP	4
TEST.PHP.....	5
RESULTADO.PHP	5
DESPLIEGUE Y DESCUBRIMIENTO	5
PRUEBAS.....	6
LOGIN.....	6
MATERIA.....	7
TEST	9
RESULTADO	10

ENUNCIADO

Realizar varios servicios que sirvan como un tutor virtual inteligente. Estos servicios seguirán una estructura definida por el profesor. La idea principal es que a partir de consultas desde cualquier cliente la aplicación tenga la capacidad de guiar a un alumno en la resolución de cuestiones específicas en materia de Tecnologías de la Información y de la Comunicación. Estos servicios almacenarán en una base de datos sus respuestas.

En ambas opciones se requiere autenticación a partir de tokens.

IMPLEMENTACIÓN

Se ha realizado una arquitectura orientada a servicios mediante REST, JSON y una base de datos relacional para la implementación del problema.

Diagrama Entidad/Relación de la Base de Datos

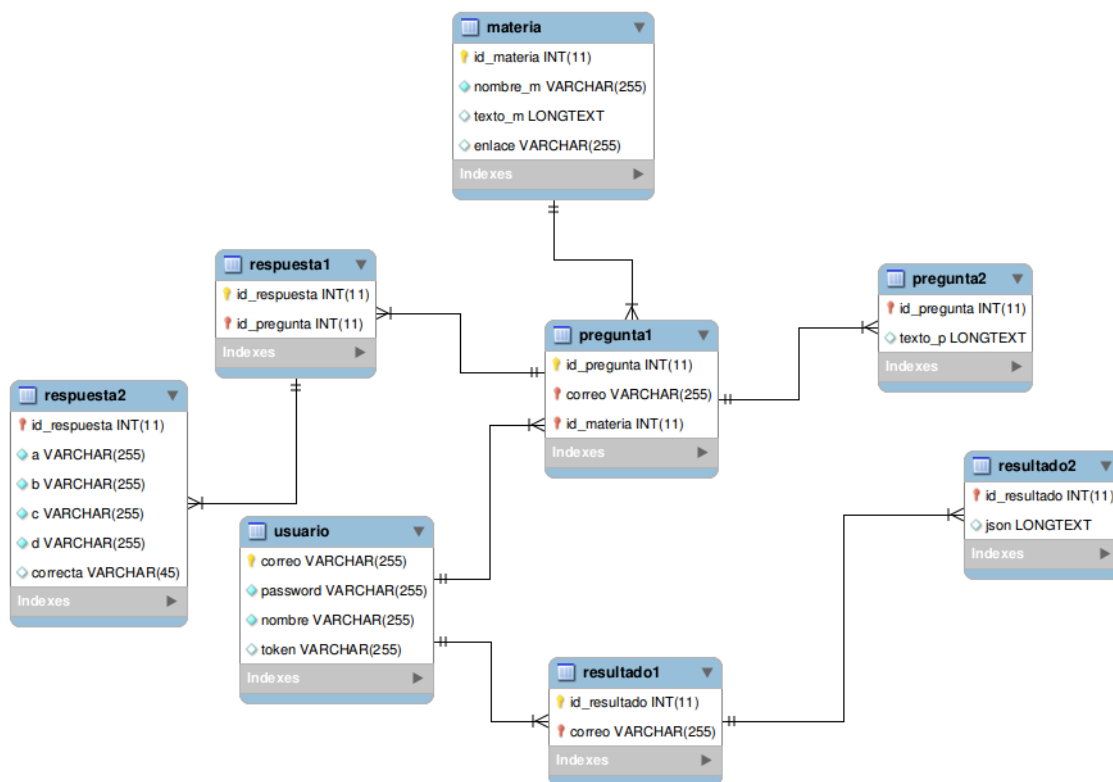


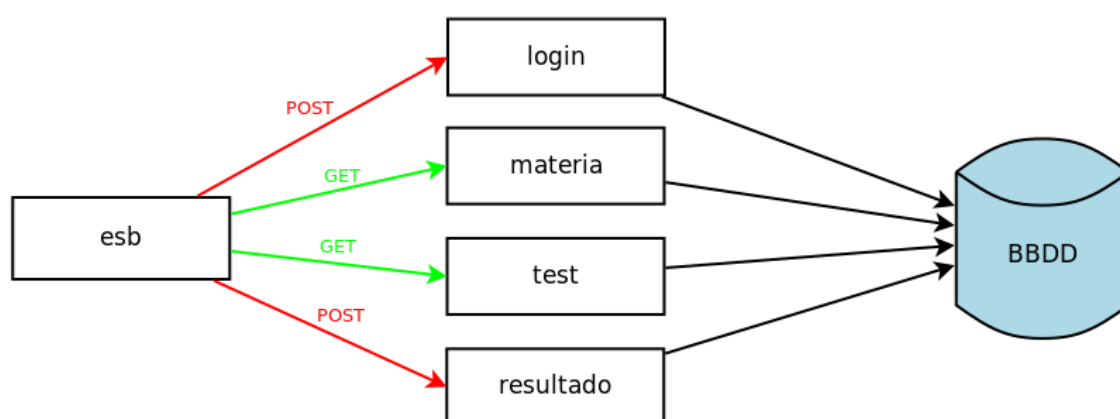
Diagrama de los servicios

El acceso a los servicios se ha implementado a través del ESB (esb.php) el cual filtra las peticiones por POST y GET; también se pueden usar los servicios de forma individual teniendo en cuenta que el servicio de resultado necesita de una autenticación de un usuario a través de token para que funcione correctamente.

Los servicios realizados son los siguientes:

- Servicio de login (login.php)
- Servicio de materia (materia.php)
- Servicio de test (test.php)
- Servicio de resultado (resultado.php)

Los servicios mencionados anteriormente son los que acceden a la base de datos como se muestra en el siguiente esquema.



Peticiones por POST

Los servicios de login y resultado van a través de peticiones POST ya que se van a crear recursos en la base de datos; en el caso del login, se creará el token del usuario y en el caso del resultado se guardarán los resultados de un test hecho por un usuario.

Peticiones por GET

Los servicios de materia y test van a través de peticiones GET ya que solamente se recuperan recursos de la base de datos; en el caso de materia, se recuperará la teoría de una materia dada y en el caso del test se recuperarán las preguntas asociadas a una materia dada.

DESCRIPCIÓN DE LOS SERVICIOS

En cada servicio se establece la cabecera de la siguiente forma:

```
header("HTTP/1.1 200 OK");
```

```
header("Content-Type: application/json");
```

La primera línea es para que se use el protocolo HTTP/1.1 en las peticiones.

La segunda línea es para que el tipo de contenido sea en formato JSON.

Esb.php

El esb es el archivo que dirige las peticiones de los servicios en función del tipo: GET o POST

1. Se recupera el método de petición a través de `$_SERVER['REQUEST_METHOD']`
2. En función de la petición se comprueban los parámetros que vienen y llama al servicio correspondiente a través de su URI

Login.php

El servicio de login permite la autenticación de un usuario y realiza los siguientes pasos:

1. Se comprueba que las variables “correo” y “pass” vengan por POST, sino es así, el servicio devuelve un JSON con el estado de la petición y un mensaje explicativo del error que ha sucedido
2. En caso contrario, se realizará la conexión a la base de datos, se recuperarán los datos del usuario y se actualizará el campo token en la base de datos para poder realizar la autenticación por token.
3. Por último lugar, el script devuelve el JSON con los datos del usuario autenticado.

Materia.php

El servicio de materia devuelve la teoría asociada a una materia y realiza los siguientes pasos:

1. Se comprueba si viene la materia por GET
2. Se obtiene la materia a recuperar de la base de datos y devuelve el JSON con la teoría de la materia recuperada.

Test.php

El servicio de test devuelve las preguntas de una materia asociada y realiza los siguientes pasos:

1. Comprueba si vienen los parámetros correctos por GET
2. Se realiza la consulta a la base de datos para recuperar el test de la materia correspondiente
3. Se van guardando los resultados obtenidos de la base de datos en un array de arrays y por último se devuelve el JSON correspondiente.

Resultado.php

El servicio de resultado comprueba las repuestas que ha realizado un usuario, muestra los resultados y los guarda en la base de datos; para ello realiza los siguientes pasos:

1. Conexión a la base de datos.
2. Se comprueba que el usuario se ha autenticado anteriormente, sino no podría contestar a las preguntas del test.
3. Se recuperan las respuestas que están guardadas en la base de datos y se va comprobando una por una con las respuestas que el usuario ha realizado; se van acumulando las respuestas correctas y las incorrectas.
4. El resultado se guarda en la base de datos, en el caso de que ese usuario ya hubiera contestado anteriormente al test se actualizarán los resultados, guardando así el JSON completo en la base de datos.

DESPLIEGUE Y DESCUBRIMIENTO

Para saber más acerca de cómo utilizar estos servicios ver archivo [API.html](#).

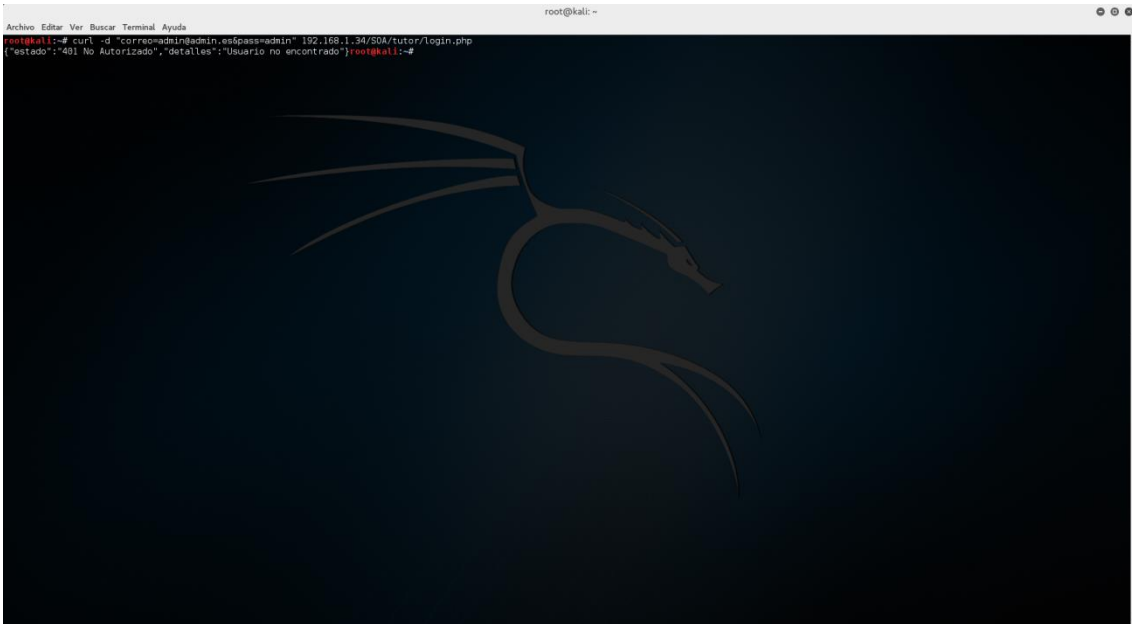
PRUEBAS

Las pruebas que se muestran a continuación se han hecho mediante curl desde una consola de comandos contra los servicios, tanto individualmente como a través del esb.

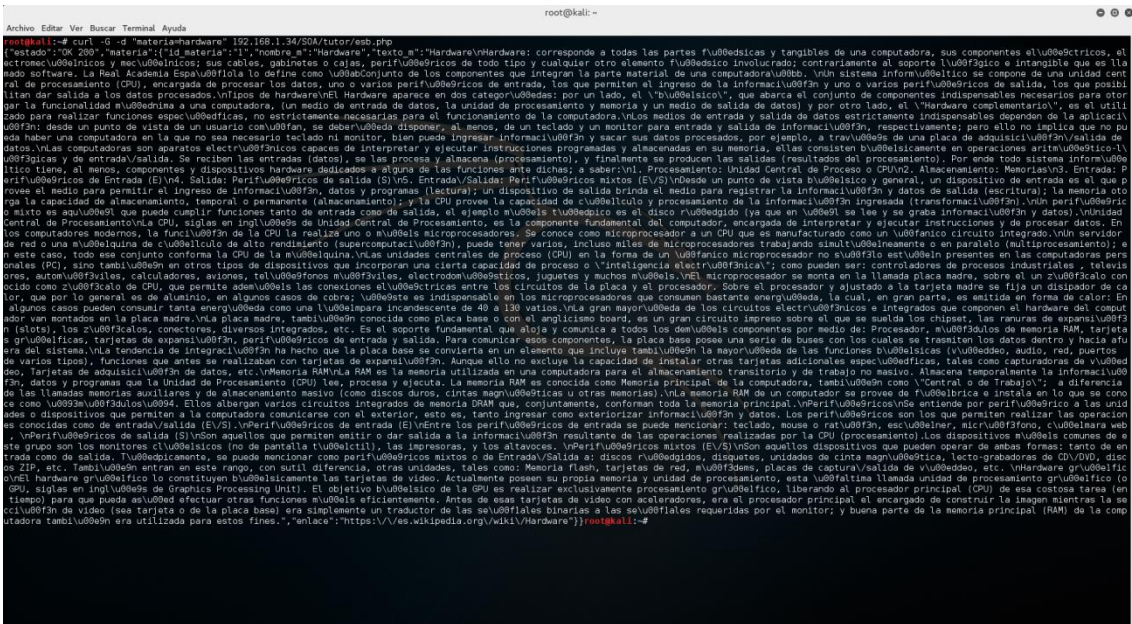
Login

```
root@kali:~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@kali:~# curl -d "correo=admin@admin.com&pass=admin" 192.168.1.34/50A/tutor/esb.php  
{\"correo\":\"admin@admin.com\",\"nombre\":\"nombre\",\"token\":\"bkwgqkn7uvnl2sljv8loipf14\",\"estado\":\"OK 200\"}root@kali:~#
```

```
root@kali:~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@kali:~# curl -d "correo=admin@admin.com&pass=admin" 192.168.1.34/50A/tutor/login.php  
{\"correo\":\"admin@admin.com\",\"nombre\":\"nombre\",\"token\":\"268lu4vl96qr8q19h2ccta7ac7\",\"estado\":\"OK 200\"}root@kali:~#
```




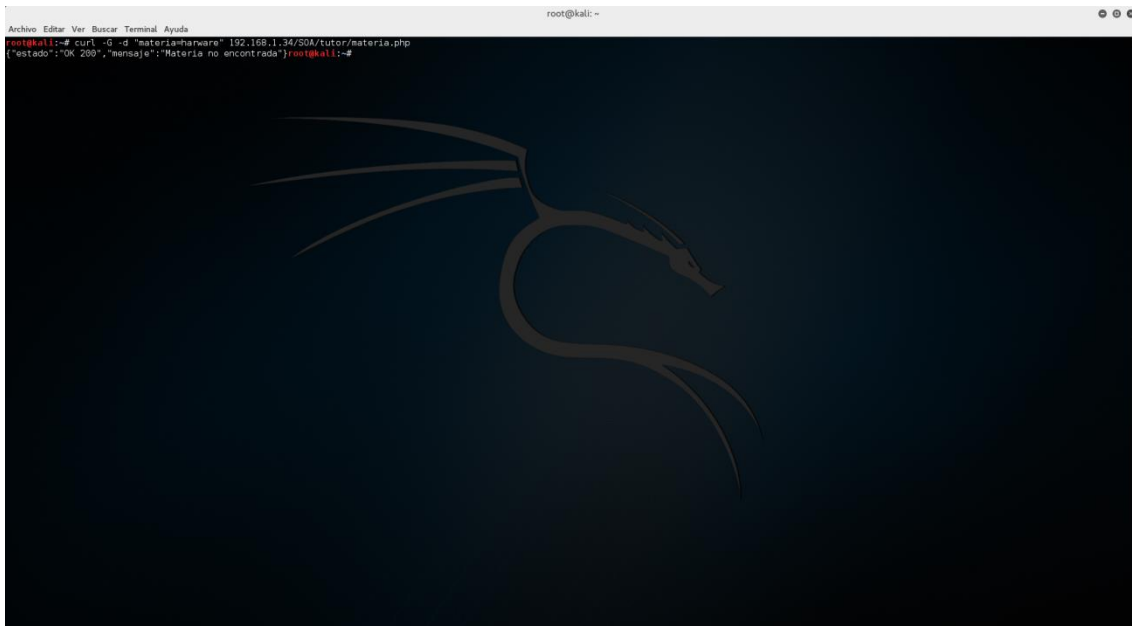
Materia



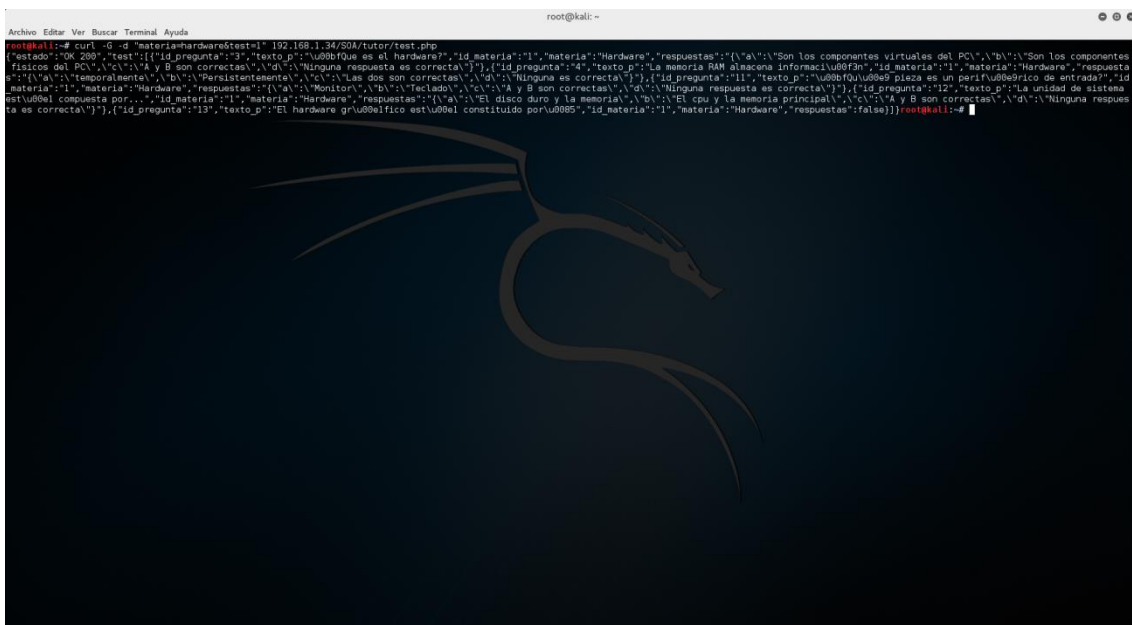
[illegible]

```
root@kali: ~  
root@kali:~# curl -G -d "materia=hardware" 192.168.1.34/soa/tutor/esb.php  
(*estado: "OK 200", "mensaje": "Materia no encontrada")root@kali:~#
```





Test



```
root@kali:~  
root@kali:~# curl -d "material=hardware&test=0" 192.168.1.34/504/tutor/test.php  
{\"estado\": \"401 No Autorizado\", \"detalles\": \"No se ha especificado la materia y/o test=1 o no viene por POST\"}root@kali:~#
```

Resultado

```
root@kali:~  
root@kali:~# curl -d "material=hardware&test={\"token\":\"md5:fn7uwl2zjv6ioipf46j-b62-e63-c64-d65-a\"} 192.168.1.34/504/tutor/esb.php  
{\"estado\": \"OK 200\", \"resultados\": {\"correctas\": 3, \"incorrectas\": 2, \"totales\": 5}}root@kali:~#
```

```
root@kali:~# curl -d "material=hardware&test=1&token=b62n7uwm2sljy8loipf46l-b62n63-c64-d65n" 192.168.1.34/SOA/tutor/resultado.php
{"estado":"OK 200","resultados":{"correctas":3,"incorrectas":2,"totales":5}}root@kali:~#
```

```
root@kali:~# curl -d "material=hardware&test=1&token=b62n7uwm2sljy8loipf46l-b62n63-c64-d65n" 192.168.1.34/SOA/tutor/resultado.php
{"estado":"401 No Autorizado","detalles":"NO autenticado"}root@kali:~#
```