

# Final de Laboratorio III – 2022

## Aclaración:

Las partes se corregirán de manera secuencial (ascendentemente). Si están bien todos los puntos de una parte, habilita la corrección de la parte posterior.

Se pueden usar templates o reutilizar código hecho, pero en ningún caso, se debe incluir código obsoleto o que no cumpla ninguna función dentro del parcial.

Toda comunicación con el backend se realizará con AJAX. Todo pasaje de datos se hará con JSON.

Las vistas (páginas .php o .html) deben respetar fielmente las formas, colores, iconos, etc. Utilizar Bootstrap 4.

Se deben respetar los nombres de los archivos, de las clases, de los métodos y de los parámetros de las peticiones.

Respetar la estructura de directorios (`index.php` → `./public`; `fotos` → `./src/fotos`; `clases` en `./src/poo`; `vistas` en `./src/views` ;).

La Api Rest que se utilizará en el parcial la proveerá el alumno → API Rest (Slim 4).

Agregar a la Api los siguientes verbos GET (a nivel de ruta): `/front-end-login`, `/front-end-registro` y `/front-end-principal`, para acceder a `login.html`, `registro.html` y `principal.php` respectivamente.

## PARTE 1 (hasta un 6)

login.html – login.ts

Asociar al evento click del botón **btnEnviar** una función que recupere el correo y la clave para luego invocar al verbo POST de la ruta **/login** (de la Api Rest).

(POST) /login

Se envía un JSON → **user** (correo y clave) y retorna un JSON (éxito: true/false; jwt: JWT (con todos los datos del usuario, a excepción de la clave) / null; status: 200/403) El JWT, tendrá una duración de 60 segundos.

Si el atributo éxito del json de retorno es false, se mostrará (en un alert de BOOTSTRAP - danger) un mensaje que indique lo acontecido.

Si es true, se guardará en el **LocalStorage** el JWT obtenido y se redireccionará hacia **principal.php**.

A login form UI mockup with a pink border. At the top, the word "LOGIN" is written in blue. Below it, there are two input fields: "Correo" with an envelope icon and "Clave" with a key icon. Below the inputs are two buttons: "Enviar" in blue and "Limpiar" in yellow. At the bottom, there is a green button that says "Quiero registrarme!".

El botón 'Quiero Registrarme!' llevará al usuario hacia la página **registro.html**.

**Colores e iconos:** lightpink; lightgrey – fas fa-envelope; fas fa-key;

registro.html – registro.ts

Se registrarán los datos de un nuevo usuario.

Asociar al evento click del botón **btnRegistrar** una función que recupere el correo, la clave, el nombre, el apellido, el perfil y la foto. Invocar al verbo POST de la ruta **/usuario** (de la Api Rest).

(POST) /usuario

Alta de usuarios. Se agregará un nuevo registro en la tabla usuarios \*.

Se envía un JSON → **usuario** (correo, clave, nombre, apellido y id\_perfil \*\*) y **foto**.

La foto se guardará en ./src/fotos, con el siguiente formato: **id\_apellido.extension**.

Ejemplo: ./src/fotos/24\_perez.jpg

\*\* ID auto-incremental. 1.- propietario, 2.- encargado, 3.- empleado.

Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

Si el atributo éxito del json de retorno es false, se mostrará (en un alert de BOOTSTRAP - danger) un mensaje que indique lo acontecido.

Si es true, se redirigirá hacia **login.html**.

El formulario de registro tiene un encabezado con el título "REGISTRO" en letras azules. El cuerpo del formulario contiene los siguientes elementos:

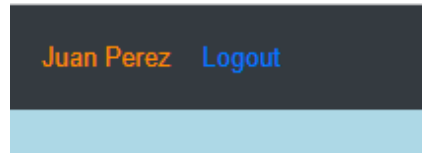
- Un campo de texto para "Correo" con un ícono de correo electrónico a la izquierda.
- Un campo de texto para "Clave" con un ícono de llave a la izquierda.
- Un campo de texto para "Nombre" con un ícono de usuario a la izquierda.
- Un campo de texto para "Apellido" con un ícono de usuario a la izquierda.
- Un menú desplegable para "Seleccionar perfil" con un ícono de tarjeta de identificación a la izquierda.
- Un campo de texto para "Seleccionar archivo" con un ícono de cámara a la izquierda. Al lado del campo, hay un texto que dice "No se eligió archivo".
- En la parte inferior, hay dos botones: "Registrar" (verde) y "Limpiar" (amarillo).

**Colores e iconos:** rgb(153, 167,184); lightgrey – fas fa-envelope; fas fa-key; fas fa-user; far fa-id-card; fas fa-camera;

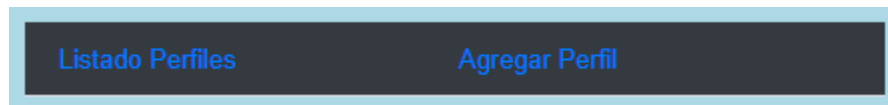
principal.php – principal.ts

Debe tener el nombre y apellido del usuario 'logueado'.

La única opción de menú (por el momento) será el Logout, que al pulsarlo redigirá hacia *login.html* (limpiando el JWT).



En el cuerpo, se tendrá un tab con las siguientes solapas.



**NOTA:** en cada interacción, se debe verificar la autenticidad del usuario, para ello, se tiene que invocar al verbo GET (ruta **/login**, de la ApiRest).

(GET) /login

Se envía el JWT → **token** (en el header) y se verifica. En caso exitoso, retorna un JSON (éxito: true/false; mensaje: string; status: 200/403).

Si el **JWT** no es válido, redirigir al **login.html**.

Al pulsar la solapa *Listado Perfiles* del tab (dejarla como activa), se mostrará el listado de los perfiles en una tabla de BOOTSTRAP (elegir un estilo).

Invocar al verbo GET (ruta **/perfil**, de la Api Rest).











(GET) /perfil

Listado de perfiles. Mostrará el listado completo de los perfiles (array JSON).

Retorna un JSON (éxito: true/false; mensaje: string; dato: stringJSON; status: 200/424)

Si el atributo éxito del json de retorno es false, se mostrará (en un alert de BOOTSTRAP - danger) el mensaje recibido.

Si es true, se armará (en el frontend) el listado que proviene del atributo **dato** del json de retorno.

Listado Perfiles		Agregar Perfil	
ID	DESCRIPCIÓN	ESTADO	ACCIONES
1	propietario	1	 
2	supervisor	1	 
3	invitado	1	 
4	empleado	1	 
5	gerente	1	 

Agregar al listado de perfiles, una columna extra (Acciones) con dos botones.

El primer botón (btn-danger) permitirá el borrado del perfil, previa confirmación del usuario, preguntando si el perfil con tal descripción y estado se quiere borrar de la base de datos. Para ello, se cargarán todos los datos del perfil en el formulario (formulario baja / modificación). Mostrarlo en una **ventana modal** (de tipo pop-up).

Si se confirma, se eliminará el perfil, invocando al verbo DELETE (del grupo /perfiles).

(DELETE) /perfiles

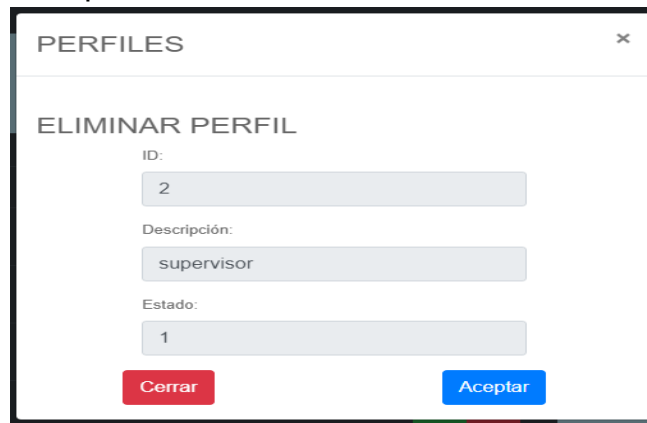
*Borrado de perfiles por ID.*

*Recibe el ID del perfil a ser borrado (**id\_perfil**, cómo parámetro de ruta) más el JWT → **token** (en el header).*

*Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)*

Si el atributo éxito del json de retorno es false, se mostrará (en un alert de BOOTSTRAP - warning) el mensaje recibido.

Si es true, refrescar el listado de perfiles.



El segundo botón (btn-info) permitirá la modificación del perfil seleccionado. Para ello, se cargarán todos los datos del perfil en el formulario (formulario baja / modificación). Mostrarlo en una **ventana modal** (de tipo pop-up).

Al pulsar el botón **btnModificar** (cambiarlo en el formulario) se invocará al verbo PUT (del grupo /perfiles).

(PUT) /perfiles

*Modificación de perfiles por ID.*

*Recibe el JSON del perfil a ser modificado → **perfil** (descripcion y estado), el ID → **id\_perfil** (id del perfil a ser modificado) (cómo parámetros de ruta) y el JWT → **token** (en el header).*

*Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)*

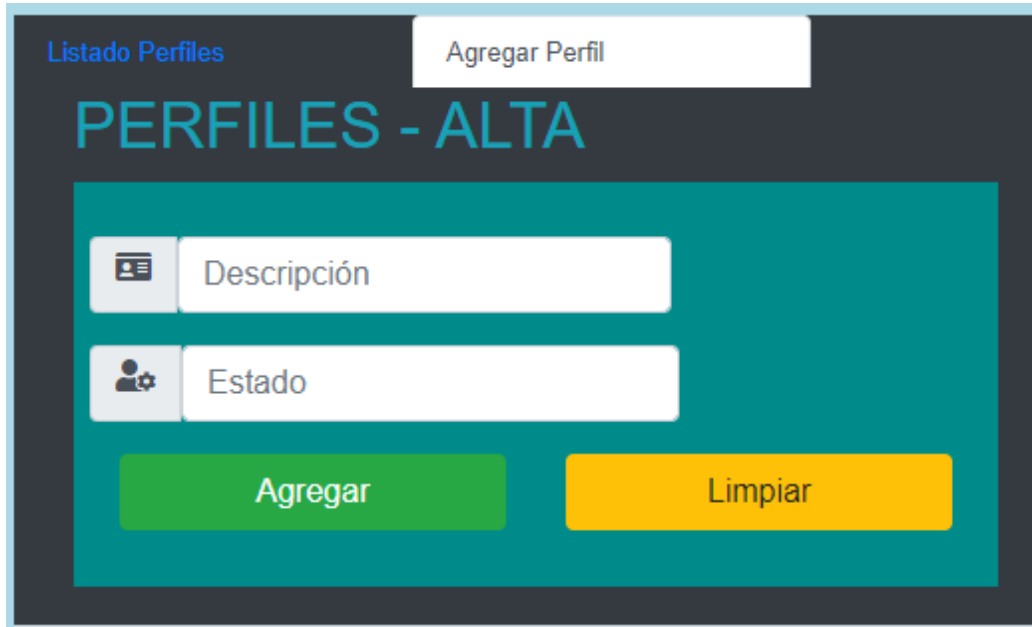
Si el atributo éxito del json de retorno es false, se mostrará (en un alert de BOOTSTRAP - warning) el mensaje recibido.

Si es true, refrescar el listado de perfiles.



## Parte 2 (hasta un 7)

Al pulsar la solapa *Agregar Perfil*, se mostrará el formulario de alta de perfiles (cómo muestra la imagen) en el cuerpo del tab.



**Colores e iconos:** darkcyan – fas fa-id-card; fas fa-user-cog;

Asociar al evento click del botón **btnAgregar** una función que recupere la descripción y el estado. Generar las validaciones correspondientes (campos NO vacíos) para TODOS los campos del formulario.

Invocar al verbo POST (nivel de aplicación de la Api Rest).

(POST) Alta de perfiles. Se agregará un nuevo registro en la tabla perfiles \*.

Se envía un JSON → **perfil** (descripción y estado \*\*).

\* ID auto-incremental. \*\* 1 → Activo y 0 → Inactivo.

Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

Si el atributo éxito del json de retorno es false, se mostrará (en un alert de BOOTSTRAP - danger) el mensaje recibido.

Si es true, se mostrará un mensaje (alert de BOOTSTRAP – success) indicando lo sucedido.

**NOTA:** en cada interacción, se debe verificar la autenticidad del usuario, para ello, se tiene que invocar al verbo GET (ruta **/login**, de la ApiRest).

(GET) /login

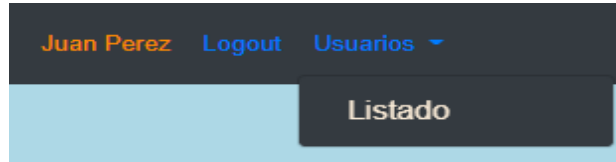
Se envía el JWT → **token** (en el header) y se verifica. En caso exitoso, retorna un JSON (éxito: true/false; mensaje: string; status: 200/403).

Si el **JWT** no es válido, redirigir al **login.html**.

## Parte 3

Agregar una opción al menú, *Usuarios*.

Al pulsar el submenú *Listado* del menú Usuarios, se mostrará el listado de los usuarios en una tabla de BOOTSTRAP (elegir un estilo). Mostrarlo debajo del tab.



Invocar al verbo GET (nivel de aplicación, de la Api Rest).

(GET) *Listado de usuarios. Mostrará el listado completo de los usuarios (array JSON).  
Retorna un JSON (éxito: true/false; mensaje: string; dato: stringJSON; status: 200/424)*

Si el atributo éxito del json de retorno es false, se mostrará (en un alert de BOOTSTRAP - danger) el mensaje recibido.

Si es true, se armara (en el frontend) el listado que proviene del atributo **dato** del json de retorno. Aplicar un filtro para que se muestre solamente el ID, nombre, apellido y perfil de los usuarios.

ID	NOMBRE	APELLIDO	PERFIL
1	propietario	gonzalez	1
2	empleado	perez	4
3	supervisor	martinez	2

**NOTA:** en cada interacción, se debe verificar la autenticidad del usuario, para ello, se tiene que invocar al verbo GET (ruta **/login**, de la ApiRest).

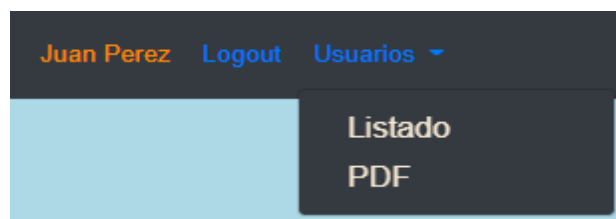
(GET) */login*

*Se envía el JWT → **token** (en el header) y se verifica. En caso exitoso, retorna un JSON (éxito: true/false; mensaje: string; status: 200/403).*

*Si el **JWT** no es válido, redirigir al **login.html**.*

Agregar el submenú *PDF* a la opción al menú Usuarios.

Al pulsar este submenú, se generará la visualización de un archivo en formato .pdf con el listado de los usuarios.



Invocar al verbo GET (de la ruta /pdf).

(GET) /pdf

Listado en formato **.pdf**.

Se envía el JWT → **token** (en el header) y mostrará:

\*-**Encabezado** (apellido y nombre del alumno a la izquierda y número de página a la derecha)

\*-**Cuerpo** (Título del listado, listado completo de los usuarios con su respectiva foto)

\*-**Pie de página** (fecha actual, centrada).

**NOTA:** El archivo **.pdf** contendrá clave, si el perfil es empleado, será el apellido del usuario logueado y si es distinto de empleado será el nombre del usuario logueado.

Agregar el Middleware de la parte anterior para que verifique el jwt.