

# Segundo parcial de Laboratorio III

## 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.

POO: obligatorio.

Toda comunicación con el backend se realizará con AJAX.

Todo pasaje de datos se hará con JSON.

La Api Rest que se utilizará en el parcial la proveerá el alumno. Se sugiere utilizar como base, la ya realizada en el segundo parcial de programación III -> *API Rest* para la concesionaria de autos Citroneta, que interactúe con la clase *Auto*, la clase *Usuario* y la base de datos *concesionaria\_bd* (autos - usuarios).

## PARTE 1 (hasta un 5)

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) / null; status: 200/403).

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**.

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

A login form UI mockup with a light pink background. 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. Both fields have placeholder text. Below the inputs are two buttons: "Enviar" in blue and "Limpiar" in yellow. At the bottom, there is a green button with the text "Quiero registrarme!".

**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 **/usuarios** (de la Api Rest).

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

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

\* ID auto-incremental. \*\* propietario, encargado y 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**.

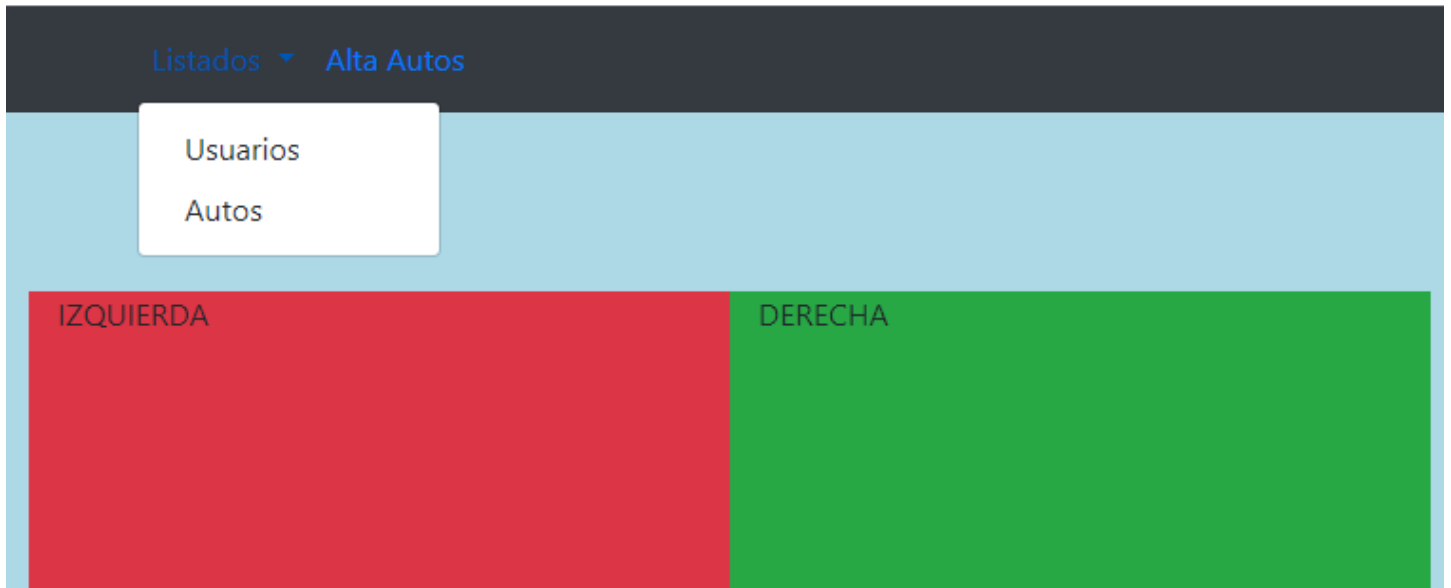
El formulario de registro, titulado "REGISTRO", contiene los siguientes elementos:

- Campo de texto "Correo" con un ícono de envelope.
- Campo de texto "Clave" con un ícono de key.
- Campo de texto "Nombre" con un ícono de user.
- Campo de texto "Apellido" con un ícono de user.
- Lista desplegable "Seleccionar perfil" con un ícono de id-card.
- Campo de selección de archivo "Seleccionar archivo" con un ícono de camera, que muestra el mensaje "No se eligió archivo".
- Botón "Registrar" de color verde.
- Botón "Limpiar" de color 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 un menú (cómo muestra la imagen) y tener dividido el cuerpo en dos secciones (izquierda y derecha).








Al pulsar el submenú *Usuarios* del menú Listados, se mostrará el listado de los usuarios en una tabla de BOOTSTRAP (elegir un estilo). Las fotos tendrán un ancho y un largo de 50px.  
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; tabla: 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 tabla del json de retorno. Mostrarlo en la sección derecha.

CORREO	NOMBRE	APELLIDO	PERFIL	FOTO
a@a.com	a	a	propietario	
b@b.com	b	b	empleado	
c@c.com	c	c	supervisor	
d@d.com	d	d	empleado	
e@e.com	e	e	empleado	

Al pulsar el submenú *Autos* del menú Listados, se mostrará el listado de los autos en una tabla de BOOTSTRAP (elegir otro estilo).

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

*(GET) Listado de autos. Mostrará el listado completo de los autos (array JSON).*

*Retorna un JSON (éxito: true/false; mensaje: string; tabla: 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 tabla del json de retorno. Mostrarlo en la sección izquierda.

MARCA	COLOR	MODELO	PRECIO
citroen	blanco	c4	123456
renault	gris	r9	50500
ford	rojo	fiesta	95300

## PARTE 2 (hasta un 6)

Agregar al listado de autos, dos columnas extras, cada una con un botón.

El primero con un botón (btn-danger) que permitirá el borrado del auto, previa confirmación del usuario.

Si se confirma, se eliminará el auto, invocando al verbo DELETE (de la Api Rest).

*(DELETE) Borrado de autos por ID.*

*Recibe el ID del auto a ser borrado más el **JWT** (en el header).*

*Si el perfil es '**propietario**' se borrará de la base de datos. Caso contrario, se mostrará el mensaje correspondiente (indicando que usuario intentó realizar la acción).*

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

Si es true, refrescar el listado de autos.

El segundo con botón (btn-info) que permitirá la modificación del auto seleccionado. Para ello, se cargarán todos los datos del auto en el formulario (mostrarlo en la sección izquierda).

Al pulsar el botón **btnModificar** se invocará al verbo PUT (de la Api Rest).

*(PUT) Modificar los autos por ID.*

*Recibe el JSON del auto a ser modificado (**auto**), el ID y el **JWT** (en el header).*

*Si el perfil es '**propietario**' o '**encargado**' se modificará de la base de datos. Caso contrario, se mostrará el*

*mensaje correspondiente (indicando que usuario intentó realizar la acción).  
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 el **JWT** no es válido, redirigir al **login.html**.  
Si es true, refrescar el listado de autos.

## PARTE 3

Al pulsar la opción de menú *Alta Autos*, se mostrará el formulario de alta de auto (cómo muestra la imagen) en la sección izquierda.

El formulario está contenido en un recuadro con fondo teal y una borde roja. Contiene cuatro campos de entrada de texto, cada uno con un icono en un botón gris a la izquierda: 'Marca' con el icono TM, 'Color' con el icono de una paleta, 'Modelo' con el icono de un coche, y 'Precio' con el icono de un dólar. Debajo de los campos hay dos botones rectangulares: uno verde con el texto 'Agregar' y uno amarillo con el texto 'Limpiar'.

**Colores e iconos:** darkcyan – fas fa-trademark; fas fa-palette; fas fa-car; fas fa-dollar-sign;

Asociar al evento click del botón **btnAgregar** una función que recupere el color, la marca, el precio y el modelo. Invocar al verbo POST (nivel de aplicación de la Api Rest).

*(POST) Alta de autos. Se agregará un nuevo registro en la tabla autos \*.  
Se envía un JSON → **auto** (color, marca, precio y modelo).*

*\* ID auto-incremental.  
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.

## PARTE 4

Utilizando los métodos MAP, REDUCE y FILTER, se pide:

Agregar una opción de menú que muestre el listado completo de los autos cuyo precio sea mayor a 250888. Invocar al verbo GET (ruta **/autos**, de la Api Rest). En el frontend, filtrar el listado obtenido, de acuerdo al precio indicado. Mostrar el listado en la sección derecha.

Agregar una opción de menú que muestre el promedio de precios. Invocar al verbo GET (ruta **/autos**, de la Api Rest).

Mostrar el promedio obtenido en un alert de BOOTSTRAP – info.

Agregar una opción de menú que muestre el listado de usuarios sólo con el nombre y la foto, de todos aquellos usuarios que sean **empleados**. Invocar al verbo GET (nivel de aplicación, de la Api Rest). Mostrar en la sección izquierda.