

Materia:	Programación III 🕶		
Nivel:	1º Cuatrimestre →		
Tipo de Examen:	Primer Parcial 🕶		
Apellido ⁽¹⁾ :		Fecha:	22 de may 2025
Nombre/s ⁽¹⁾ :		Docente a cargo ⁽²⁾ :	Plazas, Ricardo Gaston
División ⁽¹⁾ :		Nota ⁽²⁾ :	
DNI ⁽¹⁾ :		Firma ⁽²⁾ :	

- (1) Campos a completar solo por el estudiante en caso de imprimir este enunciado en papel.
- (2) Campos a completar solo por el docente en caso de imprimir este enunciado en papel.

▲ Consignas Generales

- Este parcial está dividido en varias partes que se corregirán de forma secuencial. Solo si todos los puntos de una parte están bien resueltos, se habilitará la corrección de la siguiente.
- Toda interacción con el backend debe realizarse mediante fetch (AJAX),
 y la transmisión de datos debe ser en formato JSON.
- La comunicación se realizará con el API ubicado en: https://utnfra-api-pinturas.onrender.com
 Rutas disponibles:
 - GET /pinturas
 - GET /pinturas/:id
 - POST /pinturas
 - PUT /pinturas/:id
 - DELETE /pinturas/:id
- Para realizar pruebas del API, se debe utilizar la colección Postman provista en la tarea del aula virtual.
- El backend lo provee el docente, y se deben respetar **estrictamente** los nombres de los endpoints, parámetros y tipos de datos.



- Todas las referencias a archivos o recursos deben ser relativas.
- El proyecto debe desarrollarse a partir del starter proporcionado en la tarea del aula virtual (renombrar la carpeta con apellido y nombre).
 Estructura inicial esperada:
 - ./css/style.css
 - ./img/utnLogo.png
 - o ./js/manejadora.js
 - ./pintureria.html
- No se deben modificar nombres de archivos ni estructura de carpetas.
- El archivo principal es pintureria.html junto con los archivos manejadora.js y style.css.
- El frontend debe tener un diseño moderno y atractivo, utilizando
 Bootstrap 5 o superior.
- Se evaluará el uso correcto y semántico de:
 - HTML5 (estructura)
 - CSS personalizado
 - Componentes de Bootstrap
- Todas las validaciones deben estar implementadas tanto para agregar como para modificar registros.
- Incluir un archivo README.md con:
 - o Instrucciones para ejecutar el proyecto
 - Funcionalidades desarrolladas
 - Referencia a qué parte del examen responde cada funcionalidad



Parte 1 – Listado inicial y alta básica

- 1. Al cargar pintureria.html, obtener el listado de pinturas con GET /pinturas.
- 2. Mostrar los datos en una tabla con:
 - o Columnas: ID, MARCA, PRECIO, COLOR, CANTIDAD
 - Campo color renderizado con <input type="color" disabled>
- Asociar btnAgregar con una función que envíe el nuevo objeto con POST /pinturas.
- 4. Usar clases de Bootstrap para estilos y colores de feedback.

Extra HTML/CSS

- Asegurarse de que el formulario tenga estructura semántica.
- Utilizar un layout responsive con Bootstrap.

Parte 2 – Acciones de selección, modificación y eliminación

- 1. Agregar columna **ACCIONES** con botones:
 - Seleccionar (para cargar en el formulario)
 - Eliminar (con confirmación)
- 2. Al seleccionar una pintura, completar los inputs (incluyendo el ID).
- 3. Asociar btnModificar con una función que use PUT /pinturas/:id.
- 4. Confirmar y eliminar con DELETE /pinturas/:id.

🦴 Extra JavaScript y UX/UI

- Reemplazar los botones por íconos de Bootstrap o Fontawesome.
- Mostrar mensajes de error o éxito con alertas Bootstrap.



Parte 3 – Validaciones con Bootstrap y JS

- 1. Validar:
 - Todos los campos obligatorios
 - o Precio entre 50 y 500
 - o Cantidad entre 1 y 400
- 2. Mostrar errores debajo del campo con clases invalid-feedback y is-invalid.

Extra Validaciones HTML/CSS

- Utilizar pattern, min, max, type="number" correctamente en los inputs.
- Agregar estilos visuales adicionales en style.css para destacar campos válidos (.is-valid).

Parte 4 – UX mejorado y filtros

- 1. Vaciar el formulario al agregar/modificar.
- 2. Agregar un **spinner** Bootstrap mientras se espera respuesta del API.
- 3. Agregar botón **Filtros**:
 - Mostrar solo las pinturas cuya marca coincida con el input inputMarca
- 4. Agregar botón **Promedio**:
 - o Calcular y mostrar el precio promedio con alert.

🦴 Extra Interacción y Dinamismo

- Implementar animaciones con CSS.
- Permitir ordenar la tabla por precio (JS puro).



Parte 5 – Mejoras visuales y experiencia de usuario (UX/UI)

Lograr un diseño profesional, visualmente atractivo y responsive.

- 1. Reemplazar la tabla simple por una tabla responsive y accesible.
- 2. Incluir una **barra de navegación fija** arriba con logo UTN, nombre de la app y botones de navegación (Inicio, Alta, Listado).
- 3. Incorporar un sistema de tabs o acordeón (Bootstrap) para dividir:
 - o Formulario de carga/modificación
 - Listado general
 - Estadísticas
- 4. El diseño debe ser 100% responsive, usando container-fluid, row, col-md-6, d-flex, flex-wrap, etc.

🐪 Extra:

- Agregar íconos en botones con Bootstrap Icons
- Usar colores coherentes y buena tipografía (Roboto, Open Sans, etc.)

Parte 6 - Funcionalidades estadísticas y exportación

Agregar análisis y exportación de datos.

- 1. Agregar botón: **Mostrar estadísticas** (nuevo div o modal que muestre):
 - o Total de pinturas cargadas
 - Marca más común
 - Pintura con mayor precio
 - Promedio general y por marca (usando funciones reduce, map, etc.)
- 2. Agregar botón: Exportar a CSV



- Convierte el listado actual a CSV y permite descargar el archivo.
- o Tip: Blob, URL.createObjectURL(), y a.download
- Incluir una opción de modo oscuro/claro (botón toggle que cambie el tema).

🐪 Extra:

 Mostrar estadísticas en gráficos simples usando solo CSS (barras con width proporcional) o HTML puro.

Bonus - Validaciones accesibles y documentación técnica

- Agregar atributos aria-*, role, y etiquetas label for correctamente en el formulario.
- Incluir mensajes accesibles para lectores de pantalla.
- Redactar un archivo README.md con:
 - Estructura del proyecto
 - Explicación de partes y funcionalidades
 - Capturas de pantalla
 - Qué aprendió el alumno al hacerlo

Requisitos Técnicos Finales

- Usar solo JavaScript puro (sin frameworks).
- No usar JQuery ni otras librerías externas, salvo Bootstrap.
- Código limpio y comentado. Evitar duplicación.
- No subir carpetas ni archivos no utilizados.



•

■ Sistema de Calificación del Parcial – Programación III (2025)

Parte	Descripción	Puntos	
1	Listado inicial, carga de datos con GET, y alta con POST		
	- Estructura de tabla y campos mostrados correctamente (5 pts)		
-	- Uso de Bootstrap en tabla y diseño general (5 pts)	15	
	- Función agregar implementada correctamente (5 pts)		
2	Modificación (PUT) y eliminación (DELETE)		
	- Función de selección de fila y carga en el formulario (5 pts)		
	- Eliminación con confirmación y feedback (5 pts)	20	
	Modificación correctamente realizada y listado actualizado (5 pts)		
	- Reemplazo de botones por íconos, alertas de éxito/error (5 pts)		
3	Validaciones con JS + Bootstrap		
	- Validaciones numéricas (precio y cantidad) (5 pts)		
	- Estilos de campos inválidos con Bootstrap (is-invalid) (5 pts)		
	- Mensajes de error visibles y personalizados (5 pts)		
4	UX mejorado: Spinner, filtros, promedio		
	- Spinner visible mientras se interactúa con el API (5 pts)		
	- Función de filtro por marca implementada correctamente (5 pts)		
	- Promedio calculado y mostrado con alert (5 pts)		
5	Mejora visual: responsive, tabs/navbar, estética general	15	



Parte	Descripción	Puntos		
	- Navbar funcional con navegación (5 pts)			
	- Diseño responsive con Bootstrap (row, col, etc.) (5 pts)			
	- Organización visual clara (uso de tabs o acordeón) (5 pts)			
6	Estadísticas, exportación CSV, modo oscuro			
	- Estadísticas calculadas (marca más común, promedio, etc.) (5 pts)	15		
	- Generación y descarga de CSV funcional (5 pts)			
	- Implementación del modo oscuro/claro con persistencia (5 pts)			
	Accesibilidad, documentación técnica (README, capturas, etc.)			
Bonus	- Atributos aria-*, label for, accesibilidad básica (2 pts)	5		
	- Documento README explicativo con estructura y capturas (3 pts)			