UCP

UNIVERSIDAD DE LA CUENCA DEL PLATA

Trabajo Práctico 2

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Paradigmas y Lenguajes de la Programación III

Alumna: Arguello Kiara

Docente: Mgter. Ing. Agustín Encina

Año: 2025

Indice:

Resumen Trabajo Practico 1

En el TP1 el objetivo principal fue diseñar la estructura base del sitio web de Alondra, un sistema de gestión hotelera. Todo se trabajó con HTML y CSS, buscando crear una maqueta funcional que mostrara cómo sería la navegación, el diseño general y la distribución de los contenidos.
El proyecto incluyó varias secciones: una landing page con la propuesta del servicio, páginas de login, dashboard con métricas simuladas, y un formulario de contacto. En esta primera etapa todavía no había lógica de programación, por lo que los datos eran estáticos y las validaciones mínimas.

Durante el desarrollo aprendí a manejar Flexbox y Grid para organizar los elementos de forma adaptable, creando una estructura responsive básica. También apliqué transiciones, efectos hover y una buena jerarquía visual. El foco estuvo en construir una base sólida a nivel de maquetado y estilo, dejando preparada la pagina para incorporar JavaScript y mejorar las correcciones del TP1 en el TP2.

Introducción

En este segundo trabajo práctico enfoqué el desarrollo en incorporar JavaScript y diseño responsive para darle vida al sitio y convertirlo en una experiencia interactiva. A diferencia del TP1, donde la estructura era completamente estática, en este trabajo el objetivo fue aplicar lógica y dinamismo a cada página: animaciones, validaciones, modales, filtrado de datos y almacenamiento local.
También trabajé en que la web se adapte correctamente a distintos tamaños de pantalla, manteniendo una buena usabilidad tanto en desktop como en móvil. El uso de SASS me permitió mantener los estilos ordenados y consistentes entre todas las vistas, facilitando el mantenimiento del proyecto.
En conjunto, este trabajo representa el salto de un prototipo visual a una aplicación web funcional y adaptable, con un enfoque más profesional y orientado a la experiencia del usuario.

Mejoras al proyecto

Index

En esta primera parte de la web me enfoqué en que todo cargue suave y sin errores. Usé el evento DOMContentLoaded para que el JavaScript empiece a funcionar solo cuando el contenido ya está listo. Esto evita fallos y hace que las animaciones se vean bien sincronizadas con el render.

Sección “Landing”

La parte del encabezado la hice para que el texto aparezca con una animación sutil. Quise que el usuario tenga una primera impresión fluida, sin movimientos bruscos. La idea fue darle una entrada progresiva al texto principal apenas entra a la página. Visualmente mejora mucho y transmite prolijidad. Además agregue una animación de tipo fade-in + slide-up, aplicada mediante transiciones CSS que se activan con clases controladas por JavaScript, logrando una entrada visual limpia y fluida

Sección “Beneficios”

Acá implementé un carrusel automático que se mueve solo, pero también se puede pausar con el mouse. Lo diseñé para que se repita de forma infinita y no quede cortado al final. Además, cada tarjeta tiene una descripción que se genera desde el JS a partir de un atributo del HTML, así no tuve que escribir todo manualmente. Esto me permitió mantener el código más limpio y dinámico.

Sección "Planes"

En esta parte quise simplificar la acción del usuario. Si alguien hace clic en un plan, el JS detecta si hay una sección de contacto en la misma página. Si está, hace scroll suave hasta ahí; si no, lo manda directo a la página de contacto. Es una forma práctica de guiar al usuario sin que tenga que pensar mucho.

Login

En esta página el JavaScript cumple una función puntual: manejar el inicio de sesión.

Cuando el DOM termina de cargar, el script selecciona el formulario y los campos de correo y contraseña. Al hacer clic en el botón de login, el JS toma los valores ingresados y los compara con un usuario y contraseña específicos (ucp123 y 123).
Si coinciden, redirige automáticamente al dashboard; si no, muestra un mensaje de error creado dinámicamente debajo del formulario.

La validación es local y controlada totalmente por JS, sin conexión a base de datos, ideal para simular el flujo de autenticación en una maqueta. También usa manipulación del DOM (creación de nodos, asignación de clases y estilos inline) para mantener el código simple y sin dependencias externas.

dashboard

En esta página el JavaScript tiene todo el manejo lógico del panel de gestión hotelera, sin servidor ni base de datos externa —todo se guarda y se lee desde localStorage.

Al cargarse el DOMContentLoaded, el script inicializa la base de datos local si está vacía y genera automáticamente algunos hoteles y reservas de ejemplo. Desde ahí maneja todos los filtros, indicadores, tablas y modales.

Estructura general

Usé funciones utilitarias ($, $$) para seleccionar elementos rápido y formatters para fechas y montos. Esto mantiene el código más limpio y evita repetir funciones comunes.
El estado de los filtros se guarda en un objeto global state, que luego se usa para filtrar las reservas y renderizar solo lo necesario, sin recargar la página.

Indicadores y tablas

El JS calcula ocupación, huéspedes alojados, check-ins, check-outs e ingresos diarios. Todo se actualiza en tiempo real cada vez que cambian los filtros o se agregan nuevas reservas.
Los datos se muestran en el HTML mediante inyecciones directas (innerHTML) con formato visual y etiquetas de estado.

Modal "Nueva Reserva"

El modal se abre y cierra con animaciones simples controladas por clases CSS y se enfoca automáticamente al abrir. Al guardar, valida campos, genera un ID automático, y guarda la nueva reserva en localStorage. Luego actualiza todo el dashboard (filtros, indicadores, tablas y resumen).

Resumen de hoteles

También se calcula la ocupación de cada hotel y se representa con una barra de progreso. Es un resumen visual rápido del estado general, todo generado dinámicamente.

Contactanos

Manejo completo del formulario de contacto del lado del cliente: selecciona el form y el contenedor de alertas cuando el DOM está listo, y desde ahí controla mensajes, validación y reset. Es a modo de validar, no de almacenado de datos.

Mensajes dinámicos accesibles

Definí un helper showAlert que escribe el texto, setea clases (ok / error) y muestra el bloque de alerta (que en HTML está con aria-live="polite" para lectores). Resultado: feedback inmediato sin recargar.

Marcar requeridos y UX de errores

A todos los required les agrego aria-required, y escucho invalid/input para pintar/quitar .is-invalid y ocultar la alerta cuando el usuario corrige. Es simple y claro visualmente.

Submit con validación nativa + control propio

En el submit hago preventDefault(). Si el form no es válido, llamo reportValidity() (UI nativa del navegador) y muestro alerta de error. Si está ok, muestro mensaje de éxito y hago reset() suave. No hay backend: es un envío simulado.

Sass

trabajé los estilos usando SASS, principalmente para mantener una estructura más limpia y escalable. Lo que hice fue aprovechar sus variables, anidaciones y mixins para organizar cada sección (landing, login, dashboard y contacto) de forma separada, pero con una estética coherente entre todas. Gracias a SASS pude evitar repetir código, manejar los colores y tipografías desde un solo lugar, y mantener una jerarquía clara en los selectores. Además, usé anidación para reflejar la estructura real del HTML, lo que hizo que el mantenimiento sea mucho más simple. En general, aplicar SASS me permitió tener un estilo más modular, ordenado y profesional, y trabajar con una metodología parecida a la de proyectos reales.

Conclusión

Con este trabajo logré consolidar todo lo aprendido en las etapas anteriores, sumando la lógica y el comportamiento dinámico que le faltaba a la versión inicial. Gracias a JavaScript pude hacer que la web reaccione a las acciones del usuario, con animaciones suaves, validaciones inteligentes y almacenamiento de datos locales para simular la gestión real de un sistema hotelero.
El uso de SASS me permitió mantener el código mucho más ordenado y fácil de escalar, mientras que el enfoque responsive garantizó que la página se vea bien en distintos dispositivos. En conjunto, este proyecto me ayudó a integrar el diseño, la estructura y la programación en una misma solución, dando como resultado un sistema más profesional, completo y coherente, listo para seguir evolucionando hacia una versión con base de datos y backend real.