Escuela Politécnica Nacional

Construcción y evolución de software

Proyecto 1er. Bimestre

Tema: ENCANTO EA- Tienda Virtual de Arreglos Florales

Integrantes:

Chicaiza Tipantuña Andrea Maite

Rocha Rocha Evelin Ximena

2025-2026

Proyecto: ENCANTO EA — Tienda Virtual de Arreglos Florales

Proyecto: ENCANTO EA
Integrantes: *Andrea Chicaiza, Evelin Rocha*
Curso: 2025-B SW Constr GR2

Tecnologías: HTML, CSS (Proyecto Web Estático)
Tipo de Proyecto: Página web de tienda virtual donde los usuarios pueden explorar arreglos florales, visualizar precios e información detallada y realizar compras simuladas.

Descripción General del Proyecto
ENCANTO EA es una aplicación web desarrollada en HTML y CSS cuyo objetivo es ofrecer una experiencia moderna, estética y funcional para la compra de arreglos florales. El sistema permite a los usuarios navegar por un catálogo de productos, personalizar arreglos florales, iniciar sesión, gestionar un carrito de compras y procesar pedidos. Para los administradores, la plataforma integra paneles de control para gestionar productos, clientes y pedidos.
El sitio está diseñado como una tienda virtual intuitiva donde compradores y administradores pueden interactuar con las diferentes funcionalidades mediante una interfaz clara, agradable y enfocada en la experiencia del usuario.

PLAN MAESTRO DE SCM — Proyecto 001

Abarca la planificación, identificación, control, monitoreo y entrega ordenada de los artefactos del proyecto. El objetivo es garantizar que todos los componentes (páginas HTML, hojas de estilo, imágenes, documentación y versiones) se mantengan bajo control y evolucionen sin afectar la estabilidad del sistema.

FASE 1 — ANTES DE ESCRIBIR LA PRIMERA LÍNEA DE CÓDIGO

(Planificación y Diseño)

1. Objetivos del SCM

- Controlar versiones de todas las páginas HTML, CSS e imágenes.
- Mantener trazabilidad entre cambios, autores y fechas.
- Garantizar que los cambios no rompan la funcionalidad del sitio.
- Centralizar los artefactos del proyecto en un repositorio confiable.
- Establecer flujos claros de trabajo para programadores, diseñadores y administradores del contenido.

2. Alcance del SCM

El SCM cubre los siguientes elementos del proyecto:

- Archivos HTML (módulos de comprador y administrador).
- Archivos CSS (styles.css, framework.css, inicio.css).
- Carpeta de imágenes (productos, logos, banners).
- Documentación del proyecto (README, requerimientos, manuales).
- Versiones y ramas del repositorio Git.

3. Herramientas Seleccionadas

Control de versiones:

- Git
- GitHub (repositorio remoto)

Gestión de tareas:

- GitHub Projects (tablero Kanban u otras AzureDevops)

Comunicación del equipo:

- WhatsApp
- GitHub Issues para seguimiento técnico

Diseño UI / Recursos:

- Figma para wireframes e interfaz visual
- Carpeta `/assets` para imágenes optimizadas

4. Estructura del Repositorio

Nombre del repositorio:

```
encanto-ea-web
```

Tipo:
Monorepo (todo dentro de un solo repositorio)

Carpetas iniciales:

```
/docs            → Requisitos, manuales, arquitectura
/src             → Código HTML/CSS
/assets          → Imágenes, íconos, logos
/design          → Exportaciones desde Figma
/tests           → (Opcional) Validaciones de accesibilidad / W3C
```

```
encanto-ea/
├── src/
│   ├── pages/
│   │   ├── Inicio.html
│   │   ├── Login.html
│   │   ├── Carrito.html
│   │   ├── PanelAdministrador.html
│   │   ├── PanelExplorarProductos.html
│   │   ├── PanelPrincipal_Comprador.html
│   │   ├── GestionarClientes.html
│   │   ├── GestionarProductos.html
│   │   ├── Personalizar-Producto.html
│   │   ├── Procesar-Compra.html
│   │   ├── Procesar-Pedido.html
│   │   └── Agradecimiento.html
│   ├── css/
│   │   ├── styles.css
│   │   ├── framework.css
│   │   └── inicio.css
│   └── imagenes/
│       ├── cajaFloral.jpeg
│       ├── cajaPremium.jpg
│       ├── imagenDespedida.avif
│       ├── logoEncantoEA.png
│       ├── ramo.jpeg
│       ├── standarCatalogo.jpg
│       └── standarCliente.jpg
├── docs/
│   └── requerimientos.md
└── README.md
```

3. Política de Ramas (Branching Strategy)

Rama principal: `main`

- Estará **protegida**:
  - No se permite push directo
  - Solo se puede fusionar mediante Pull Request (PR)
  - Requiere revisión de al menos 1 compañero

Ramas secundarias:

- `feature/nombre-funcion` → Nuevas características
  Ej: `feature/carrusel-productos`
- `bugfix/descripcion` → Correcciones
  Ej: `bugfix/corregir-footer`
- `hotfix/urgente` → Bugs en producción
  Ej: `hotfix/filtro-precios`

4. Workflow del Proyecto (Flujo de trabajo)

1. Se crea un Issue en GitHub o AzureDevops describiendo el requerimiento

2. Se crea una rama desde `main` con naming convention
3. Se desarrolla el cambio en la rama feature
4. Se realiza commit siguiendo el estándar
5. Se abre un Pull Request (PR)
6. Se revisa por un compañero
7. Se aprueba y se fusiona a `main`

5. Gestión de Artefactos (No código)

Requisitos:

- Guardados dentro de `/docs/requisitos.md`

Diseños de interfaz:

- Enlazados desde el README
- Exportaciones dentro de `/design/`

Manual del sitio web:

- `/docs/manual-usuario.pdf`

Inventario de productos florales:

- `/docs/catalogo-productos.xlsx u otro diferente.`

FASE 2 — DURANTE EL DESARROLLO (Codificación y Pruebas)

- ## Gestión de Cambios y Pull Requests (PR)

Los siguientes elementos se consideran **ítems de configuración (CI)**:

- **CI-01:** Página Inicio (Inicio.html)
- **CI-02:** Página Login
- **CI-03:** Carrito
- **CI-04:** Panel de Administrador
- **CI-05:** Gestión de Clientes
- **CI-06:** Gestión de Productos
- **CI-07:** Hojas de estilo CSS
- **CI-08:** Imágenes
- **CI-09:** Documentación
- **CI-10:** README del proyecto

Cada CI deberá tener un estado: *En desarrollo*, *En revisión*, *Aprobado*, *Deprecado*.

1. Gestión de Cambios y Pull Requests (PR)

1. El desarrollador crea una rama para su cambio.

2. Realiza commit siguiendo formato:

- `feat: agregar buscador en PanelExplorarProductos`
- `fix: corregir tamaño de imagen en Inicio`

3. Realiza **Pull Request (PR)** hacia `main`.

4. El administrador SCM revisa:

- coherencia,
- validación visual (HTML/CSS),
- no ruptura de estilo.

5. Si es aprobado → se hace *merge*.
6. Si requiere ajustes → se devuelve al desarrollador.

Un PR solo se fusiona si cumple la **Definition of Done (DoD):**

Definition of Done

El PR debe incluir:

- Código limpio en HTML/CSS y sin errores W3C
- Imágenes optimizadas (peso máximo: 300 KB por imagen)
- Cumplimiento del diseño de Figma
- Incluye un mensaje de commit que referencia el Issue
  Ejemplo:
  `git commit -m "Agrega sección de catálogo floral (closes #12)"`
- Aprobación de al menos un revisor
- No rompe la estructura visual del sitio (validado con Mobile View)

2. Trazabilidad

Todos los cambios deben estar vinculados a un Issue:

- `closes #ID`
- `fixes #ID`
- `resolves #ID`

Esto permite rastrear quién hizo qué y por qué.

3. Integración Continua (CI)

Aunque el proyecto es HTML/CSS, igual se puede aplicar CI.

Se configurará GitHub Actions para ejecutar:

- Validación automática de HTML con `htmlhint`
- Validación de CSS con `stylelint`
- Comprobación de tamaños de imágenes
- Construcción automática y despliegue a GitHub Pages

4. Gestión de Línea Base (Baselining)

Cuando se complete un módulo importante se generará un **tag Git**.

Ejemplos:

- `v0.1-catalogo` → cuando se completa el catálogo de productos
- `v1.0-beta` → sitio web completo sin pasarela de pagos
- `v1.0.0` → primera versión de producción

FASE 3 — DESPUÉS DEL LANZAMIENTO

(Despliegue y Mantenimiento)

El producto final deberá incluir:

- Todo el código HTML/CSS validado.
- Imágenes optimizadas.
- Documentación actualizada (requerimientos, manual técnico, manual de usuario).
- Estructura ordenada según el árbol de carpetas.

1. Gestión de Releases y Despliegue (CD)

El sitio se desplegará mediante **GitHub Pages**.
Cada vez que se haga un merge en `main`, GitHub Actions generará la nueva versión automáticamente.

Esquema de Versionamiento

Usaremos **Versionamiento Semántico**:

- **MAJOR** — Cambios que afectan toda la estructura
- **MINOR** — Nuevas secciones o componentes
- **PATCH** — Correcciones pequeñas

Ejemplo:

- `v1.0.1` → se arregla un error en el menú
- `v1.1.0` → se añade sección “Ramos Personalizados”
- `v2.0.0` → nuevo rediseño completo del sitio

2. Plan de Mantenimiento Proactivo (Los 4 Tipos)

1. Mantenimiento Correctivo

Proceso para bugs:

1. Registrar Issue con etiqueta **bug**
2. Crear rama `hotfix/` desde `main`
3. Corregir
4. Crear PR
5. Fusionar y generar parche (PATCH)
   Ejemplo: `v1.0.2`

2. Mantenimiento Adaptativo

Cada vez que:

- Cambie el navegador (Chrome/Firefox/Edge)
- Se actualicen librerías
- Se agregue compatibilidad móvil

Se abre un Issue cada **3 semana** para revisar compatibilidad.

3. Mantenimiento Perfectivo

Orientado a mejorar calidad del proyecto:

- Optimizar CSS
- Reorganizar carpetas
- Mejorar accesibilidad
- Reducir tamaño de imágenes

Se dedicará **1 día cada semana** a refactorización.

4. Mantenimiento Preventivo

Para evitar problemas futuros:

- Activar **Dependabot** para librerías externas
- Revisar Build de GitHub Pages
- Validar enlaces rotos automáticamente
- Monitorear tiempos de carga (Lighthouse)