Proyecto de Software 2025

Trabajo Integrador (TI)

Contexto

El Programa Nacional de Registro y Preservación de Sitios Históricos es una iniciativa orientada a documentar, proteger y difundir el patrimonio cultural de las distintas regiones del país. A través del relevamiento y la sistematización de datos, se busca facilitar el acceso a información precisa sobre sitios históricos, monumentos y espacios de valor cultural, promoviendo su conservación y puesta en valor para las generaciones presentes y futuras.

La propuesta contempla la recopilación de información histórica, geográfica y fotográfica de cada sitio, así como la integración de datos sobre su accesibilidad y relevancia cultural. Esta información será administrada y centralizada para permitir consultas rápidas y seguras, tanto por parte de las autoridades competentes como por investigadores, guías turísticos y ciudadanos interesados.

La aplicación contará con tres tipos de usuarios:

- **Usuarios públicos**: podrán explorar el mapa de sitios históricos, publicar reseñas y marcar sitios como favoritos.
- **Editores**: serán responsables de administrar los sitios históricos y validar tanto la información como las reseñas.
- **Administradores**: además de las funciones de los editores, tendrán la capacidad de gestionar usuarios y asignar roles dentro del sistema.

Además existirá al menos un “system admin” que tendrá acceso completo a toda funcionalidad del sistema.

Objetivo general

El objetivo de este trabajo es desarrollar una **aplicación web** que centralice la información sobre sitios históricos de distintas ciudades del país, permitiendo su consulta y actualización de manera eficiente. La plataforma incluirá una interfaz de administración para la carga, edición y exportación de datos, así como la generación de mapas interactivos que muestren los sitios relevados a nivel local, regional y nacional.
La solución estará compuesta por una **aplicación interna de administración** (para editores y administradores) desarrollada en **Python** y **Flask**, y un **portal web público** desarrollado en **Vue.js** para la visualización de los sitios y mapas. Se empleará una base de datos **PostgreSQL** con soporte para datos geoespaciales, y se implementará una API con los endpoints necesarios para la integración entre la aplicación de administración y el portal público.

Funcionalidades de la aplicación (se dividirá en dos etapas)

El desarrollo de la aplicación se realizará en dos etapas. En líneas generales la funcionalidad mínima que se deberá incluir es:

- Mantener un registro detallado de los sitios históricos de distintas ciudades, incluyendo información geográfica, descripción, imágenes y estado de conservación.
- Permitir que los usuarios públicos propongan nuevos sitios históricos, sujetos a validación por editores o administradores antes de su publicación.
- Gestionar reseñas y valoraciones de los sitios por parte de los usuarios públicos.
- Administrar la información y el historial de ediciones realizadas por editores y administradores.
- Gestionar usuarios y roles.
- Mostrar mapas interactivos que muestren la distribución y características de los sitios históricos registrados.

Componentes de la aplicación

- **Aplicación de administración en Python y Flask:** permitirá a editores y administradores realizar altas, bajas y modificaciones de sitios históricos, reseñas y otros recursos del sistema. Incluirá registro y gestión de usuarios, autenticación y control de roles.
- **Base de datos PostgreSQL:** estructura optimizada para almacenar información georreferenciada de los sitios históricos, junto con datos de usuarios, reseñas y validaciones.
- **API para consultas e integración:** endpoints que proveerán la información necesaria al portal público y a la aplicación de administración, incluyendo datos para mapas interactivos y reportes estadísticos.
- **Portal en Vue.js (Mobile First):** interfaz pública interactiva optimizada para dispositivos móviles, que permitirá explorar el mapa de sitios históricos, consultar información, dejar reseñas y proponer nuevos sitios.

Dinámica de trabajo en equipo y evaluación

Es importante que el trabajo sea desarrollado en equipo, asignando roles y tareas a cada miembro para garantizar una implementación eficiente y exitosa de la aplicación web. Durante el cuatrimestre, **se realizarán reuniones periódicas para monitorear el progreso y resolver dudas**. Estas reuniones serán coordinadas con el docente asignado al grupo.
Al finalizar el cuatrimestre, cada grupo deberá presentar las aplicaciones web completas y funcionales, demostrando conocimiento sobre todas las funcionalidades requeridas y su correcto despliegue en un entorno de prueba.
Además, si bien las tareas podrán dividirse entre los integrantes para organizar el trabajo, **todos los miembros del equipo deben comprender el funcionamiento integral del sistema** y ser capaces de responder preguntas sobre cualquiera de sus partes. No será válido que un grupo separe

completamente las responsabilidades (por ejemplo, backend y frontend) sin que cada integrante tenga un conocimiento básico y compartido de lo realizado por los demás.

Etapa 1

Aplicación Privada

Esta aplicación estará destinada a editores y administradores para la gestión integral de los sitios históricos. Permitirá realizar altas, bajas y modificaciones de la información registrada, y administrar reseñas.

Se desarrollará utilizando el framework **Flask** en **Python**, conectado a una base de datos **PostgreSQL** con soporte para datos geoespaciales. Contará con un sistema de autenticación y control de roles para garantizar que cada usuario acceda únicamente a las funciones correspondientes a su perfil.

1 Layout

Descripción

Implementar un **layout base** para todas las vistas de la aplicación privada. Este layout funcionará como contenedor principal y garantizará una estructura general uniforme, permitiendo que cada vista reemplace únicamente el contenido central sin alterar el resto del diseño.

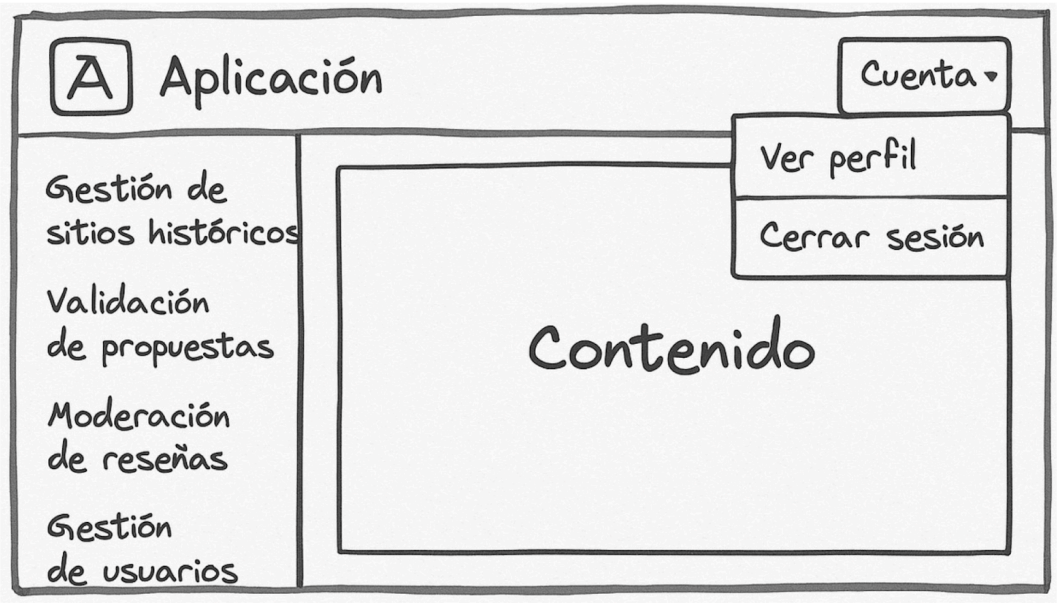


Figura 1. Posible visualización del layout de la aplicación privada.

Requerimientos funcionales

- Menú de navegación lateral o superior con enlaces a todos los módulos del sistema:
    - Gestión de sitios históricos
    - Moderación de reseñas
    - Gestión de usuarios
- Barra superior (top bar) con:
    - Logo de la aplicación visible en todas las secciones.
    - Dropdown de usuario/cuenta para ver perfil y cerrar sesión.
- El layout debe ser responsive para visualizarse correctamente en distintas resoluciones.

Notas

- No es necesario seguir un diseño fijo; se espera una propuesta funcional y coherente con la estética general de la aplicación.
- El logo debe mantenerse en una posición coherente y con un tamaño adecuado en todas las vistas.

2 Módulo de Usuarios

Descripción

Implementar el módulo de usuarios de la aplicación privada, accesible únicamente para usuarios con rol Administrador. Este módulo permitirá crear, leer, actualizar y eliminar usuarios del sistema.

Requerimientos funcionales

- Operaciones CRUD de usuarios.
- Validar que no existan dos usuarios con el mismo email.
- Campos mínimos para cada usuario:
    - Email
    - Nombre
    - Apellido
    - Password

    - Activo: SI | NO
    - Rol: Usuario público | Editor | Administrador
- Funcionalidad de búsqueda por:
    - Email
    - Activo (SI | NO)
    - Rol
- Resultados ordenables por:
    - Fecha de creación (Ascendente y descendente).
- Paginación del lado del servidor (máximo 25 registros por página).

Notas

- Validar en el cliente:
    - Campos requeridos.
    - Formato de campos.
- Validar en el servidor:
    - Campos requeridos.
    - Formato de campos.
    - Unicidad de email.
- Se debe mostrar un feedback claro al usuario en caso de errores de validación.

Permisos

- Usuarios con rol **Administrador** pueden acceder a este módulo.

3 Gestión de Roles, Permisos y Bloqueo de Usuarios

Descripción

Implementar las funcionalidades de asignación y gestión de roles, permisos y estado de los usuarios.

Requerimientos funcionales

- Asignar o desasignar roles a un usuario (Editor, Administrador).

- Bloquear/desbloquear usuarios:
    - Un usuario bloqueado no puede iniciar sesión.
    - Los usuarios con rol Administrador no pueden ser bloqueados.
- Implementar permisos siguiendo el patrón modulo_accion. Ejemplos para este módulo:
    - `user_index` → listar usuarios
    - `user_new` → crear usuario
    - `user_update` → actualizar usuario
    - `user_destroy` → eliminar usuario
    - `user_show` → ver detalle de usuario

Notas

- Los permisos y roles pueden administrarse directamente desde la base de datos en esta etapa del proyecto. Es decir, no hace falta hacer una interfaz gráfica para agregar permisos y roles, pero deberán existir en la base de datos.
- **Es importante entender el porqué del uso de esta solución para implementar la autorización y seguir el esquema de forma correcta. Este tema será explicado oportunamente en los horarios de práctica.**

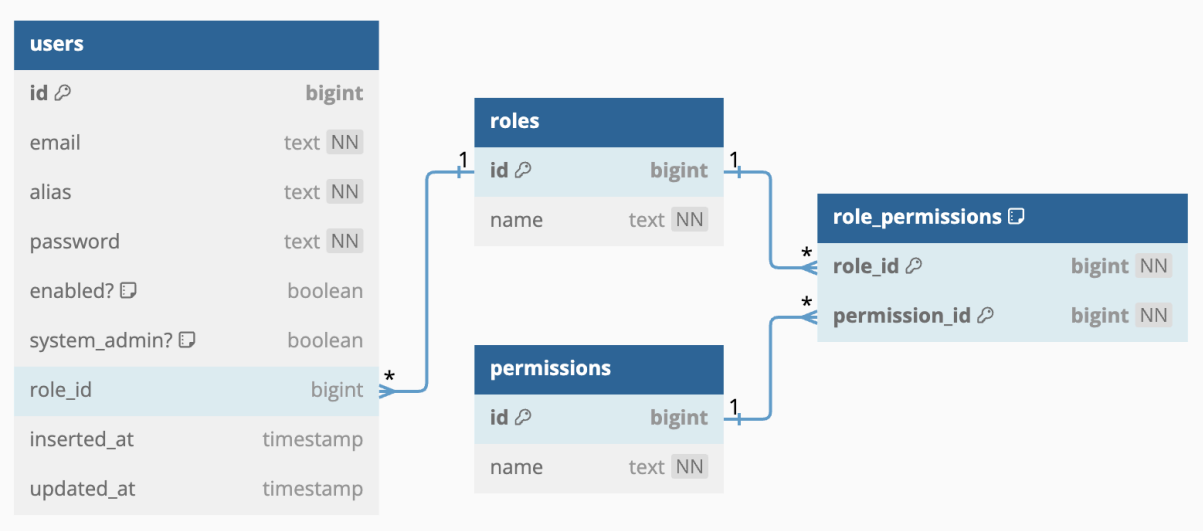


Figura 2. Posible esquema para el manejo de usuarios.

Permisos

- Todas las operaciones de roles y permisos sólo pueden ser realizadas por un usuario con rol **Administrador**.

4 Login y manejo de sesiones

Descripción

Implementar el sistema de autenticación y manejo de sesiones para la aplicación privada (Administración).

Requerimientos funcionales

- Formulario de login para la aplicación privada que permita iniciar sesión con email y contraseña.
- Validación de credenciales contra la base de datos.
- Manejo seguro de contraseñas (hash).
- Creación y manejo de sesiones para mantener el estado del usuario autenticado.
- Verificación de sesión y permisos en cada módulo.
- Implementar cierre de sesión.

Notas

- **No se permite el uso de librerías externas que abstraigan el proceso completo de login (por ejemplo, flask-login)**. La implementación debe ser manual para comprender y controlar cada paso del flujo de autenticación.
- Implementar verificación de permisos antes de acceder a funcionalidades restringidas. Esta verificación se debe realizar para todas las secciones del sistema de Administración.
- Las cookies de sesión deben ser seguras y tener fecha de expiración.
- **Deben utilizar** la librería **Flask-Session** para almacenar las sesiones del lado del servidor.
- La autenticación en la aplicación privada será obligatoria para acceder cualquier parte de la aplicación de administración.

5 Módulo de Sitios Históricos

Descripción

Implementar el módulo principal para la gestión de sitios históricos en la aplicación privada, accesible para editores y administradores. Este módulo

permitirá registrar, editar, eliminar y visualizar la información completa de cada sitio histórico.

Requerimientos funcionales

- Operaciones CRUD para sitios históricos.
- Campos mínimos para cada sitio:
    - Nombre del sitio
    - Descripción breve
    - Descripción completa
    - Ciudad
    - Provincia
    - Ubicación geográfica (latitud y longitud)
    - Estado de conservación (Bueno, Regular, Malo)
    - Año de inauguración
    - Categoría (Arquitectura, Infraestructura, Sitio arqueológico, etc)
    - Fecha de registro
    - Visible. Describe si el sitio está listo para ser visible o nó en el portal.
- Subida de una o más imágenes asociadas al sitio. **Esta funcionalidad se implementará en la etapa 2.**
- Implementar la posibilidad de agregar múltiples etiquetas que se administran en el siguiente módulo.
- Desarrollar: Búsqueda avanzada de sitios. La búsqueda debe implementarse como listado principal de este módulo.
- Agregar botón para Exportar sitios.

- Paginación del lado del servidor (máximo 25 registros por página).

- Para seleccionar las coordenadas de los sitios históricos se debe implementar un widget de mapa interactivo que permita al usuario elegir un punto sobre el mismo para obtener las coordenadas del sitio histórico. Esta funcionalidad debe estar disponible tanto en la creación como en la edición de un sitio.

Permisos

- **Editores** y **Administradores** pueden crear y modificar sitios.

- Los **administradores** pueden eliminar sitios.

Notas

- La validación de campos obligatorios debe realizarse tanto en el cliente como en el servidor.

- La gestión de imágenes se definirá con más detalle en otra tarea en la etapa 2 del trabajo.

- La búsqueda avanzada se describe con más detalle en otra tarea.

- La funcionalidad para exportar sitios se describe con más detalle en otra tarea.

- En este módulo se deben poder enlazar las etiquetas a los sitios. La gestión de etiquetas se describe en otra tarea.

6 Tags (Etiquetas) de Sitios Históricos

Descripción

Implementar la gestión de tags para clasificar y facilitar la búsqueda de sitios históricos. Los tags podrán asignarse en forma múltiple a cada sitio.

Requerimientos funcionales

- CRUD de tags.
- Campos mínimos del tag:
    - Nombre (obligatorio, único, 3–50 caracteres, case-insensitive)
    - Slug (autogenerado desde el nombre, único, solo minúsculas y guiones)
- Búsqueda por nombre.
- Orden por nombre y por fecha de creación (asc/desc).
- Paginación del lado del servidor (25 elementos por página).
- Asignación múltiple de tags a cada sitio desde el formulario de Sitios (selector multiselección). Tanto para crear como para editar un sitio.
- Restricción al eliminar: no permitir borrar un tag si está asignado a algún sitio (mostrar mensaje).

Permisos

- **Editores** y **Administradores** pueden gestionar tags.

Notas

- Generar slug automáticamente en backend al guardar (normalizar, quitar acentos, espacios → guiones).
- Validar unicidad de nombre y slug en base de datos.
- Validación de duplicados en el servidor con feedback claro al usuario.

7 Búsqueda avanzada de Sitios Históricos

Descripción

Agregar un buscador avanzado en la aplicación privada para localizar sitios históricos combinando múltiples filtros.

Requerimientos funcionales

- Filtros disponibles (combinables entre sí):
    - Ciudad (texto o selector)
    - Provincia (selector)
    - Tags (multiselección)
    - Estado de conservación (Bueno | Regular | Malo)
    - Fecha de registro (rango: desde / hasta)
    - Visibilidad (checkbox)
- Búsqueda por texto: El texto debe estar contenido en el nombre del sitio o la descripción breve.
- Orden de resultados: por fecha de registro, nombre o ciudad (asc/desc).
- Paginación del lado del servidor (25 elementos por página).
- Reset de filtros con un clic (botón “Limpiar”).
- Mostrar contador de resultados y estado “sin resultados” cuando corresponda.
- Lista de resultados con columnas clave (Nombre, Ciudad, Provincia, Estado de conservación y visibilidad).
- Mantener los filtros seleccionados al paginar o cambiar el orden (mismo estado de pantalla).

Permisos

- Esta funcionalidad se agrega al Listado de Sitios Históricos por lo que los permisos son los descritos ahí.

Notas

- Validaciones básicas: rango de fechas válidas.
- Mantener coherencia con el estilo del Layout y la paginación ya implementada en otros módulos.

8 Exportación de datos de Sitios Históricos a CSV

Descripción

Permitir exportar desde la aplicación privada un archivo CSV con información de los sitios históricos registrados, incluyendo sus tags asociados.

Requerimientos funcionales

- Botón “Exportar CSV” en el listado de sitios históricos del panel de administración.
- El archivo debe incluir, como mínimo, las siguientes columnas:
    - ID del sitio
    - Nombre
    - Descripción breve
    - Ciudad
    - Provincia

    - Estado de conservación
    - Fecha de registro
    - Coordenadas de geolocalización
    - Tags asociados (tener precaución para diferenciar los múltiples tags asociados con el resto de los valores de atributos de un sitio).

- El CSV debe respetar el filtro y orden actuales aplicados en el listado antes de exportar.
- Codificación UTF-8 con separador , (coma).
- Nombre del archivo: sitios_<YYYYMMDD_HHMM>.csv
- Botón visible en la vista de listado de sitios (en la barra de acciones).
- Al hacer clic, se descarga automáticamente el CSV.

Permisos

- Los **Administradores** pueden exportar datos.
- **Editores** no pueden exportar.

Notas

- Generar el archivo del lado del servidor no generar el CSV en el cliente.
- Manejar mensajes claros en caso de error (por ejemplo: “No hay datos para exportar”).
- No incluir imágenes ni descripciones completas para evitar archivos demasiado grandes.

9 Panel de visibilidad de funcionalidades (Feature Flags)

Descripción

Implementar en la aplicación privada un módulo para activar/desactivar funcionalidades del portal y del área de administración sin necesidad de despliegues. En esta versión se gestionan tres flags:

- **Modo mantenimiento de administración** (deshabilita temporalmente el sitio de administración).
- **Modo mantenimiento de portal web** (deshabilita temporalmente el portal público).
- **Permitir nuevas reseñas** (habilita/deshabilita creación y/o visualización).

Requerimientos funcionales

- Crear una sección **“Feature Flags”** accesible solo para **System Admin**.
- Mostrar un listado de flags con: nombre, descripción, estado (ON/OFF), última modificación (usuario y fecha/hora).
- Permitir cambiar el estado de cada flag mediante un toggle ON/OFF o checkbox con confirmación.
- Flags incluidos hasta el momento:
    - admin_maintenance_mode (boolean):
        - ON → bloquea todas las rutas de administración excepto login.
        - Los system admins podrán loguearse.
        - Campo adicional “Mensaje de mantenimiento” (texto corto) que se mostrará en el formulario de login del sistema.
        - Además el mensaje se mostrará para usuarios logueados si el estado cambia sin importar en qué sección se encuentren.
    - portal_maintenance_mode (boolean):
        - ON → Pone el portal en modo mantenimiento.
        - También tiene un campo adicional para el mensaje de mantenimiento que debe aparecer en el portal.
        - Este mensaje es diferente que el del sistema de administración.
        - Este flag se usará en la etapa 2.
    - reviews_enabled (boolean):
        - OFF → oculta/deshabilita la creación de reseñas en el

portal.

        - El backend dejará de procesar los reviews que se intenten insertar por la API.
        - Este flag se usará en la etapa 2.
- Auditoría básica: registrar fecha de última modificación y usuario que realizó la última modificación.
- Validaciones: longitud máxima del mensaje de mantenimiento; confirmación al activar mantenimiento.
- Los mensajes son obligatorios cuando se habilitan los modos de mantenimiento.

Permisos

- **System Admins** puede visualizar y modificar los flags.
- **Administradores** y **Editores**: afectados por los flags, pero no pueden modificarlos ni acceder al panel.
- El panel debe permanecer accesible para **System Admin** incluso con el Modo mantenimiento activado.

Notas

- Intentos de acceso al admin cuando admin_maintenance_mode = ON deben redirigir a una pantalla de mantenimiento con el mensaje configurado.
- Cuando reviews_enabled = OFF, la UI oculta la creación de reseñas y el backend rechaza operaciones no permitidas (403). **Esto se aplicará en la etapa 2.**
- Esta funcionalidad es exclusiva de **System Admins**.

10 Historial de modificaciones de un sitio

Descripción

Implementar en la aplicación privada un sub módulo para registrar y visualizar el **historial de modificaciones** realizadas sobre cada sitio histórico. Esto permitirá tener trazabilidad de los cambios y saber qué usuario los realizó.

Requerimientos funcionales

- Cada vez que se realice una acción sobre un sitio (crear, editar, eliminar, cambiar tags o imágenes), registrar un **evento** en el historial.
- Datos mínimos a registrar:
    - Sitio afectado
    - Usuario que realizó la acción
    - Fecha y hora
    - Tipo de acción (Creación, Edición, Eliminación, Cambio de estado, Cambio de tags, etc.)
- **Listado de historial** accesible desde la vista de detalle/edición de un sitio.
- **Filtros**: por usuario, por tipo de acción, por rango de fechas.
- **Orden**: cronológico (más reciente primero).
- Paginación del lado del servidor (25 registros por página).
- En la vista de detalle/edición de un sitio, sección **“Historial de cambios”** en pestaña o panel.

Permisos

- **Editores** y **Administradores** pueden visualizar el historial debe agregarse un permiso especial y agregarlo a los roles que tengan acceso a la sección.

Notas

- El registro del historial debe hacerse **automáticamente** desde el backend al modificar sitios.

- No se requiere edición ni eliminación de registros del historial (solo lectura).

Fecha límite de entrega TI (1/2): 08 de octubre 23:59

Etapa 2

Aplicación Privada

Moderación de Reseñas

Descripción

Implementar un módulo en la aplicación privada para listar, revisar y moderar reseñas creadas por usuarios públicos: aprobar, rechazar o eliminar.

Requerimientos funcionales

- Listado de reseñas con columnas: sitio, usuario (email/alias), calificación (1–5), contenido de la reseña, estado y fecha de creación.
- Estados de reseña: Pendiente, Aprobada, Rechazada (Pendiente por defecto).
- Acciones por reseña:
    - Aprobar (cambia estado a Aprobada).
    - Rechazar (cambia estado a Rechazada; motivo de rechazo obligatorio, máx. 200 caracteres).
    - Eliminar (confirmación previa).
- Filtros (combinables):
    - Estado (Pendiente/Aprobada/Rechazada)
    - Sitio (selector)

    - Calificación (rango 1–5)
    - Fecha (rango desde/hasta)
    - Usuario (email, texto)
- Orden por Fecha (asc/desc) y Calificación (asc/desc).
- Paginación del lado del servidor (25 elementos por página).
- Detalle de reseña accesible desde el listado (link o icono “ver”).
- Estados visibles con chips/etiquetas de color.
- Mensajes de confirmación y feedback claro en cada acción.

Permisos

- Moderadores, Editores y Administradores pueden moderar (aprobar/rechazar/eliminar).

Notas

- Al aprobar, la reseña pasa a estar visible para ser incluida en la respuesta de la API que luego consumirá el portal público.
- Al rechazar, la reseña no quedará disponible; guardar el motivo para auditoría.
- Eliminar: acción definitiva; exigir confirmación.
- Validaciones en cliente y servidor (longitud de motivo, estados válidos, permisos, etc.).

Gestión de imágenes de Sitios Históricos

Descripción

Permitir subir, reemplazar y eliminar imágenes asociadas a cada sitio histórico desde la aplicación privada.

Requerimientos funcionales

- Añadir imágenes a un sitio (múltiples).
- Marcar una imagen como “portada” del sitio (solo una).
- Eliminar imágenes (impedir eliminar si es la portada; exigir cambiar la portada antes).
- Ordenar imágenes.
- Campos por imagen: archivo, título/alt (obligatorio), descripción corta (opcional).

Restricciones/validaciones

- Formatos permitidos: JPG, PNG, WEBP.
- Tamaño máximo por archivo: 5 MB.
- Límite de 10 imágenes por sitio.
- Sólo una imagen puede ser marcada como portada.

Permisos

- Esta funcionalidad se agrega al Listado de Sitios Históricos por lo que los permisos son los descritos ahí.

Almacenamiento

- Las imágenes se almacenarán en un servicio MinIO configurado para el proyecto.
- En la base de datos se guardará: URL pública, título/alt, descripción, orden, indicador de portada y timestamps.

Notas

- Renombrar archivos al subir para evitar colisiones.
- Mostrar la portada en el listado de Sitios (miniatura).
- Mantener coherencia con la paginación y estilos del Layout existente.

API

Se deberán implementar en la aplicación privada una serie de endpoints que permitan realizar distintas operaciones u obtener contenido desde la aplicación pública.

Pueden acceder a la especificación de la API publicada en nuestra web en el siguiente enlace.

Nota: La especificación es una guía para el/la estudiante que podrá modificar lo definidos o agregar en caso que se considere.

Aplicación Pública (Portal)

Se deberá desarrollar **otra aplicación** a la cual accederán el público general para obtener información de interés sobre la Institución, actividades, noticias y contacto.

El código de esta nueva aplicación deberá compartir espacio dentro del mismo repositorio de código de la aplicación Flask. La aplicación pública la deberán realizar utilizando el framework VueJs 3. **Cómo requisito para el correcto funcionamiento en el servidor se deberá ubicar el código correspondiente a esta aplicación dentro del directorio web del proyecto**.

Página de inicio destacada

Descripción

Implementar la home del portal público (mobile first) con secciones destacadas de sitios: Más visitados, Mejor puntuados y Recientemente agregados. Debe facilitar el descubrimiento y la navegación hacia el listado y el detalle de cada sitio.

Requerimientos funcionales

- Hero (opcional): título del proyecto + buscador rápido (redirige al Listado con el texto aplicado).
- Secciones destacadas (cada una independiente):
    - Más visitados
    - Mejor puntuados
    - Favoritos
    - Recientemente agregados
- Cada sección muestra un carrusel o grid de tarjetas (imagen portada, nombre, ciudad/provincia, calificación si aplica).
- Botón “Ver todos” por sección que navega al Listado con el orden/filtro correspondiente aplicado.
- Las tarjetas enlazan al detalle del sitio.
- Tarjetas consistentes con el componente del listado de sitios.
- En caso de no haber contenido en las secciones mostrar un mensaje adecuado.

Permisos

- Público (no requiere autenticación).

Notas

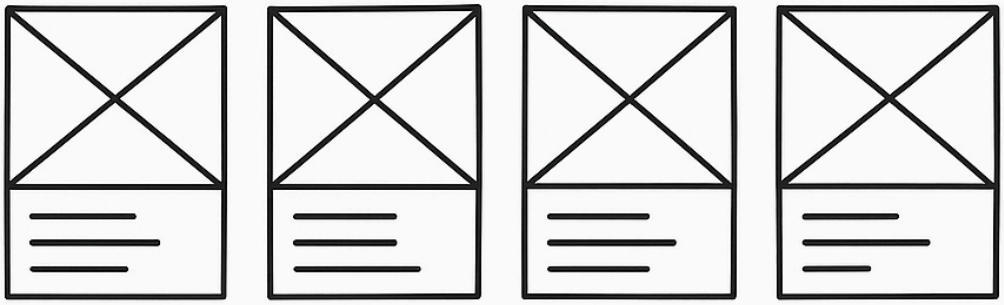
- Esta tarea consume datos de la API (definidos en la aplicación privada).
- Cargar cada sección de forma perezosa/independiente para mejorar la percepción de velocidad.
- Mantener componentes reutilizables.
- La sección de “Favoritos” sólo debería verse si el usuario inicia sesión.

Hero (opcional) Buscador rápido

TÍTULO DEL PROYECTO

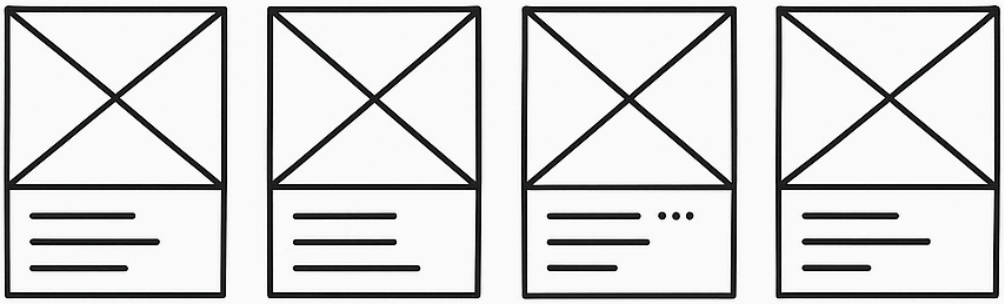
Más visitados

Ver todos ›



Mejor puntuados

Ver todos ›



Recientemente agregados

No hay contenido

Listado y Búsqueda de Sitios Históricos

Descripción

Implementar en la app pública una vista mobile first con el listado de sitios y un sistema de búsqueda por texto e integración de filtros (ciudad, provincia, tags, estado de conservación). La vista consumirá datos de la API (definida en la aplicación privada).

Requerimientos funcionales

- Listado en tarjetas con: imagen de portada, nombre, ciudad, provincia, estado y tags (mostrar 5 máximo).
- Búsqueda por texto: Aplica sobre nombre y descripción breve.
- Filtros combinables:
    - Ciudad (texto o selector)
    - Provincia (selector)
    - Tags (multiselección)
    - Favoritos (check)
- Los filtros descritos se deben poder combinar con una búsqueda visual en mapa. De forma tal que si se elige un punto en el mapa se puede buscar resultados cercanos a ese punto. Esta funcionalidad debería ser similar a la que se puede encontrar en https://www.zonaprop.com.ar/ al comenzar una búsqueda y hacer click en “Ver Mapa”.
- Orden: por fecha de registro, nombre, mejor rankeados. Ascendentes y descendentes para todos los casos.
- Paginación o carga infinita (elegir una y mantenerla consistente).
- Reset de búsqueda y filtros con un clic (“Limpiar”).
- Diseño **mobile first**: buscador y filtros en acordeón o panel plegable en móviles; vista lateral en desktop.

- Grid responsive: 1 columna en móvil, 2–4 en tablet/escritorio.
- Mantener estado de búsqueda/filtros/orden al paginar y al volver desde el detalle.
- Reflejar estado en la URL (query params) para compartir resultados.

Permisos

- Público (no requiere autenticación).

Notas

- Utilizar endpoints de la app privada.
- Reutilizar componentes de tags, tarjetas y estilos comunes.

Detalle de Sitio Histórico

Descripción

Implementar la vista mobile first de detalle de un sitio histórico en la aplicación pública. Debe mostrar información completa, galería de imágenes, tags, estado de conservación, mapa con la ubicación visible de inmediato, y el bloque de reseñas y calificaciones. Incluir botón “Escribir reseña” (requiere login).

Requerimientos funcionales

- Encabezado: nombre del sitio, ciudad/provincia, estado de conservación (chip/etiqueta).
- Imágenes: portada + carrusel/galería (con alt text).
- Descripción: breve y completa (plegable “ver más / ver menos”).

- Tags: listados y clicables (navegan al listado filtrado por ese tag).
- Mapa: componente siempre visible en la vista de detalle.
    - Mostrar marcador en la ubicación (lat/lon del sitio).
    - Tooltip con nombre y descripción breve.
    - Zoom inicial adecuado para visualizar la ciudad/localidad.
- Calificación promedio: estrellas (1–5) + cantidad de reseñas.
- Reseñas: listado paginado (25 por página) con alias del usuario, puntuación, fecha y texto. Mostrar sólo reseñas aprobadas.
- Botón “Escribir reseña”:
    - Si no está autenticado → redirigir al login con Google.
    - Si está autenticado → abrir formulario (puntuación + texto).
- Icono/Botón para marcar el sitio como favorito
    - Si no está autenticado → redirigir al login con Google.
    - Si está autenticado → habilita icono .
- Diseño mobile first, adaptado a tablet/escritorio.
- Mapa insertado como sección fija en la vista (debajo de la descripción, antes de reseñas).
- Componentes accesibles (alt en imágenes, etiquetas semánticas en controles).
- Botón de “Volver” al listado conservando búsqueda/filtros previos.

Permisos

- Público: ver toda la información.

- Autenticado: crear/eliminar sus propias reseñas.

Notas

- El mapa debe cargarse automáticamente con la vista, no mediante un botón.
- Usar librería de mapas estándar (ej. Leaflet) con tiles libres.
- Consistencia visual con chips, tarjetas y estilos ya definidos.

Registro / Login con Google

Descripción

Habilitar el inicio de sesión con Google en la app pública. La autenticación es obligatoria solo para:

- Dejar reseñas y calificaciones.
- Proponer nuevos sitios históricos.

El resto de la navegación permanece pública. Al autenticarse, se crea/actualiza un perfil básico (nombre, email y foto opcional).

Requerimientos funcionales

- Botón “Continuar con Google” visible cuando una acción requiera autenticación (escribir reseña, proponer sitio) y en el menú de usuario.
- Flujo de acceso condicional:
    - Si el usuario intenta una acción restringida sin sesión → redirigir a login con Google.
    - Tras iniciar sesión → volver a la acción original.

- Perfil básico (solo lectura en esta etapa): nombre, email, avatar (si está disponible).
- Indicadores de sesión en el encabezado (avatar/alias) y Cerrar sesión.
- Persistencia de sesión mientras el token sea válido.
- Menú de usuario (cuando hay sesión) con: Perfil, Mis reseñas, Sitios favoritos y Cerrar sesión.
- Mensajes claros en errores comunes (fallo de autenticación, intento cancelado).

Permisos

- Público (no autenticado): puede navegar, ver listados, detalles y mapa.
- Autenticado: además, puede crear reseñas y proponer sitios.

Notas

- Esta tarea consume los servicios de autenticación expuestos por la app privada (definir en issues de API aparte).
- Mantener el flujo mobile first y accesible.
- No se solicita edición del perfil en esta etapa (sólo visualización).

Perfil del Usuario

Descripción

Implementar la vista mobile first de Perfil del usuario autenticado. Debe mostrar la lista de sus reseñas y de sitios marcados como favoritos.

Requerimientos funcionales

- Cabecera de perfil: mostrar la información obtenida desde google.
- Pestañas o secciones:
    - Mis reseñas: listado con Sitio, Calificación (1–5), Fecha, Extracto.
    - Mis sitios favoritos: listado con sitios marcados como favoritos.
- Paginación server-side en ambos listados (25 por página).
- Orden: por fecha (asc/desc).
- Listados vacíos amigables:
    - “Aún no escribiste reseñas.”
    - “Aún no marcaste ningún sitio como favorito.”

Permisos

- Requiere autenticación (login con Google).
- Solo se muestran datos propios del usuario.

Reseñas y Calificaciones

Descripción

Permitir que usuarios autenticados creen y eliminen reseñas (texto + puntuación 1–5) sobre un sitio histórico.

Requerimientos funcionales

- Crear reseña (usuario autenticado):
    - Campos: puntuación (1–5) y texto.
    - Una reseña por usuario y por sitio (si existe, ofrecer Editar).
    - Validaciones: puntuación 1–5; texto 20–1000 caracteres.
- Eliminar reseña propia: con confirmación.
- Mostrar mensajes claros de si la acción se realizó o no.

Permisos

- Crear/eliminar: solo el autor autenticado.
- Navegación y lectura: público.

Notas

- Mantener coherencia con la moderación: sólo se listan reseñas aprobadas.
- Si una reseña editada vuelve a estado “pendiente”, mostrar aviso al usuario.
- Reutilizar componentes de tarjetas y formularios existentes.

Requisitos técnicos y pautas de trabajo

1 Requisitos técnicos

- El prototipo debe desarrollarse utilizando **Python**, **JavaScript**, **HTML5**, **CSS3** y **PostgreSQL**, respetando el modelo en capas **MVC**.

- Seguir las guías de estilo de Python y documentar todo el código Python con **docstrings**.

- Uso obligatorio de **Jinja** como motor de plantillas para la aplicación privada.

- La aplicación privada debe desarrollarse con el framework **Flask**.

- Validar todos los datos de entrada tanto en el cliente como en el servidor.

    - Las validaciones del lado del servidor deben implementarse en un módulo aparte que reciba los datos y devuelva el resultado.

    - Ejemplos de validaciones: solo números en un DNI, solo caracteres válidos para un nombre, opciones de lista que pertenezcan al conjunto permitido, etc.

- **Contraseñas**: deben almacenarse utilizando hash seguro (bcrypt, Argon2 o equivalente).
  Seguridad contra inyección SQL: toda interacción con la base de datos debe realizarse a través de **SQLAlchemy** para evitar inyecciones SQL.

- Se permite el uso de librerías que facilitan tareas (conexiones a servicios externos, parseo, patrones de buenas prácticas, validaciones de datos, etc.), siempre que todos los miembros del equipo puedan explicar su funcionamiento y propósito. Consultar con el ayudante si el uso de la misma está permitido.

- **Login:** no se permite el uso de librerías que abstraiga completamente el proceso (por ejemplo, Flask-Login), para asegurar comprensión de la implementación.

- Para la interacción con la base de datos, utilizar **SQLAlchemy** como ORM para mantener una capa de abstracción.

- Aplicar buenas prácticas de **semántica web en HTML5**, usando correctamente las etiquetas del lenguaje.

- Versionar todas las entregas con **git**, siguiendo **Versionado Semántico** (por ejemplo, etapa 1 → v1.x.x).

- Se permite el uso de frameworks CSS como **Bootstrap**, **Foundation**, **Bulma** o **Tailwind CSS**.

- **Manejo de logs**: implementar registro básico de eventos y errores en backend para facilitar depuración.

- **Control de dependencias**: mantener actualizados y controlados pyproject.toml (Poetry) y package.json con versiones definidas para evitar incompatibilidades.

- Todas las vistas deben ser **responsive** contemplando al menos tres resoluciones cómo mínimo (si quiere utilizar las sugeridas por el framework que utilice comuníquelo al ayudante):

    - < 767px (móviles)

    - 768px – 1024px (tablets)

    - > 1025px (portátiles y escritorio)

- Se debe utilizar VueJs (versión 3.5.X) como framework de desarrollo web para la aplicación pública/portal.

2 Pautas de trabajo y entrega

- La entrega es obligatoria. Todos los integrantes deben participar en la defensa.

- El ayudante evaluará el progreso y participación mediante consultas online y seguimiento en **GitLab**.

- El proyecto se deberá realizar en grupos de **cinco integrantes**.

- Deben ser visibles los aportes de cada integrante tanto en **GitLab** como en la defensa.

- La defensa será virtual, salvo que se acuerde presencial por falta de medios técnicos (**micrófono y cámara**).

- El trabajo será evaluado desde el **servidor de la cátedra**. No se aceptarán entregas fuera del tiempo y forma establecidos.

- Funcionalidades no terminadas en la etapa 1 podrán completarse en la etapa siguiente.

3 Información del servidor

- **Lenguaje:** Python 3.12.3
- **Servidor web:** nginx/1.24.0 (Ubuntu)
- **Dependencias Python:** Poetry 2.1.4
- **Base de datos:** PostgreSQL 16

Fecha límite de entrega TI (2/2): 20 de noviembre 23:59