DOCUMENTO FUNCIONAL DEL PROYECTO

ALUMNO/GRUPO: Guillermo Tellez Cepeda

---

1. Introducción y contexto

- **El problema o necesidad que se quiere resolver.**

  La Agenda Cultural de Catalunya publica información pública valiosa sobre actividades culturales, pero los datos se ofrecen principalmente en formatos técnicos (JSON/datos abiertos) o a través de portales con navegación compleja y poco adaptada a móviles. Esta barrera técnica dificulta que el ciudadano medio pueda descubrir, filtrar y planificar su asistencia a eventos de forma ágil. La falta de una interfaz unificada, intuitiva y centrada en la experiencia de usuario limita el acceso real a la oferta cultural pública y dispersa la información en múltiples fuentes.

- **Quién será el usuario o cliente final.**

  El proyecto está dirigido a cualquier ciudadano o visitante interesado en la cultura en Cataluña, sin necesidad de conocimientos técnicos previos. La plataforma también incluye un perfil de administrador (gestor de contenidos) encargado de mantener la base de datos actualizada mediante la carga segura de archivos oficiales.

- **Qué solución se propone y con qué propósito.**

  Se propone el desarrollo de Moute, una aplicación web responsiva que transforma los datos abiertos de la Generalitat en una experiencia de navegación clara y accesible. La plataforma permite consultar eventos mediante un listado dinámico con scroll infinito o un calendario mensual interactivo, aplicar filtros combinados (fecha, precio y categoría), gestionar favoritos de forma local y visualizar información detallada de cada actividad. Incluye un panel de administración con estadísticas y un sistema seguro de actualización manual de la base de datos.

  Su propósito es democratizar el acceso a la cultura, centralizando información dispersa en una herramienta práctica, ligera y de bajo mantenimiento, demostrando cómo los datos públicos y las tecnologías de código abierto pueden convertirse en servicios digitales útiles para la ciudadanía.

---

2. Análisis de requisitos

2.1. Requisitos funcionales (RF)

Qué debe hacer el sistema.

Código	Descripción del requisito funcional
RF1	El sistema permitirá visualizar eventos culturales en formato de listado con tarjetas o calendario mensual interactivo.
RF2	El sistema permitirá buscar eventos mediante filtros combinados: texto (título/descripción/ubicación), rango de fechas, precio (gratuito/de pago) y categoría cultural.
RF3	El sistema permitirá a los usuarios marcar eventos como favoritos y gestionarlos desde una sección dedicada, con persistencia local en el navegador.
RF4	El sistema mostrará información detallada de cada evento: descripción completa, fechas, horarios, ubicación, precio, enlaces externos e imágenes.
RF5	El sistema permitirá al administrador iniciar sesión con credenciales seguras y acceder a un panel de control restringido.
RF6	El sistema permitirá al administrador actualizar la base de datos mediante la carga de archivos JSON procedentes de la Agenda Cultural de Catalunya.
RF7	El sistema mostrará al administrador estadísticas en tiempo real: total de eventos, distribución por año/comarca/categoría/precio y últimos eventos añadidos.
RF8	El sistema permitirá al administrador cambiar su contraseña de forma segura.
RF9	El sistema diferenciará entre usuarios administradores (con acceso completo) y usuarios normales (solo visualización con nombre de sesión).
RF10	El sistema cargará eventos de forma progresiva (scroll infinito) para optimizar el rendimiento.

2.2. Requisitos no funcionales (RNF)

Cómo debe comportarse el sistema.
Incluye aspectos como rendimiento, seguridad, compatibilidad o facilidad de uso.

Código	Descripción del requisito no funcional
RNF1	La interfaz será responsive y accesible desde dispositivos móviles, tablets y escritorio, adaptándose automáticamente al tamaño de pantalla.
RNF2	Las páginas deberán cargarse en menos de 3 segundos en condiciones normales de conexión.
RNF3	El sistema soportará al menos 100 usuarios concurrentes sin degradación del rendimiento.
RNF4	Las contraseñas de administrador se almacenarán con hash seguro (Werkzeug/SHA-256).
RNF5	La aplicación será compatible con navegadores modernos: Chrome, Firefox, Safari y Edge (últimas 2 versiones).
RNF6	La base de datos mantendrá la integridad de los datos y prevendrá duplicados durante la importación.
RNF7	El código seguirá buenas prácticas de desarrollo: estructura modular, comentarios básicos y separación clara entre frontend y backend.
RNF8	La aplicación estará disponible 24/7 con un tiempo de actividad superior al 95%.

2.3. Restricciones

Condiciones o limitaciones del proyecto.

Lenguajes o tecnologías obligatorias:

- Backend: Python 3 con Flask
- Frontend: HTML5, CSS3, JavaScript (Vanilla)
- Base de datos: SQLite3
- Servidor web: Apache 2.4 con mod_wsgi
- Framework CSS: Bootstrap 5
- Control de versiones: Git/GitHub

Recursos disponibles (tiempo, equipo, materiales):

- Tiempo: Limitado al período académico del proyecto
- Equipo: Desarrollo individual
- Infraestructura: Máquina virtual Ubuntu 24.04 con acceso mediante bastión
- Presupuesto: Cero (todas las tecnologías son open source)
- Datos: Fuente única - Agenda Cultural de Catalunya (datos abiertos en formato JSON)

Dependencias o limitaciones técnicas:

- La actualización de eventos depende de la disponibilidad y estructura del JSON oficial de la Generalitat
- No se requiere conexión a internet para el funcionamiento básico una vez desplegado
- El almacenamiento está limitado por el espacio disponible en la máquina virtual
- Las imágenes se cargan desde servidores externos (agenda.cultura.gencat.cat)
- No se implementará sistema de registro de usuarios (solo login visual para usuarios normales)

---

3. Análisis de usuarios y roles

Objetivo: identificar quién usará el sistema y qué podrá hacer.

Describe los distintos tipos de usuario, sus necesidades y sus permisos.

Rol	Descripción	Permisos principales
Administrador	Usuario autorizado responsable de mantener actualizada la base de datos de eventos y supervisar el funcionamiento del sistema. Requiere autenticación con credenciales seguras.	- Cargar archivos JSON para actualizar la base de datos - Visualizar estadísticas y métricas del sistema - Cambiar contraseña de acceso - Acceder al panel de administración completo - Gestionar el contenido de eventos (inserción masiva)

Usuario registrado	Persona que inicia sesión con un nombre de usuario (sin autenticación estricta) para personalizar su experiencia. Su identidad se almacena en sesión y sus preferencias en el navegador.	- Consultar y filtrar eventos - Marcar eventos como favoritos - Gestionar sus favoritos desde sección dedicada - Visualizar detalles completos de eventos - Navegación personalizada con nombre de sesión
Visitante	Usuario anónimo que accede a la aplicación sin iniciar sesión. Tiene acceso completo a la información pública pero sin persistencia de preferencias.	- Consultar listado de eventos - Aplicar filtros de búsqueda - Visualizar calendario interactivo - Acceder a detalles de eventos - Sin acceso a favoritos persistentes

Notas adicionales:

- El sistema implementa un modelo híbrido de autenticación: los administradores requieren validación contra base de datos, mientras que los usuarios normales solo necesitan proporcionar un nombre visual.
- Los favoritos se gestionan mediante localStorage del navegador, por lo que están vinculados al dispositivo, no a la cuenta de usuario.
- No existe un sistema de registro formal de usuarios para simplificar la arquitectura y respetar la privacidad.

4. Casos de uso / Escenarios de uso

Objetivo: mostrar cómo interactúan los usuarios con el sistema.

Selecciona de tres a cinco casos principales y descríbelos brevemente.

Código	Nombre del caso de uso	Actor principal	Descripción	Resultado esperado
CU1	Consultar eventos culturales	Visitante / Usuario	El usuario accede a la página principal y visualiza los eventos en formato de listado con tarjetas o calendario mensual. Puede navegar entre diferentes vistas y hacer scroll para cargar más eventos.	El sistema muestra los eventos disponibles de forma clara y organizada, con imágenes, fechas y ubicación.
CU2	Buscar y filtrar eventos	Visitante / Usuario	El usuario introduce términos de búsqueda y/o aplica filtros combinados: rango de fechas, precio (gratuito/de pago) y categoría cultural (teatre, música, exposicions, etc.).	El sistema muestra únicamente los eventos que coinciden con los criterios de búsqueda y filtros aplicados.
CU3	Gestionar favoritos	Usuario registrado	El usuario marca eventos como favoritos desde el listado o detalle, y puede consultarlos posteriormente en la sección "Mis Favoritos".	Los eventos seleccionados se guardan en el navegador y el usuario puede acceder a ellos rápidamente en cualquier momento.

CU4	Iniciar sesión como administrador	Administrador	El administrador introduce sus credenciales (usuario y contraseña) en el formulario de login. El sistema valida las credenciales contra la base de datos.	Si las credenciales son correctas, se concede acceso al panel de administración con todas las funcionalidades de gestión.
CU5	Actualizar base de datos de eventos	Administrador	El administrador accede a la sección de actualización, selecciona un archivo JSON procedente de la Agenda Cultural de Catalunya y lo sube al sistema.	El sistema procesa el archivo, inserta los nuevos eventos evitando duplicados, y muestra un resumen de eventos añadidos correctamente.

---

5. Modelo de datos o estructura de la información

Objetivo: representar la información que gestionará el sistema.*

Entidades principales

El sistema se sustenta en dos entidades (tablas) independientes dentro de la base de datos SQLite:

- events: Almacena la información cultural procesada a partir de los archivos JSON oficiales. Contiene metadatos, fechas, ubicación, categorías, precios y recursos multimedia.
- admins: Gestiona las credenciales de acceso al panel de administración. Centraliza la seguridad y el control de sesiones privilegiadas.

Relaciones entre entidades

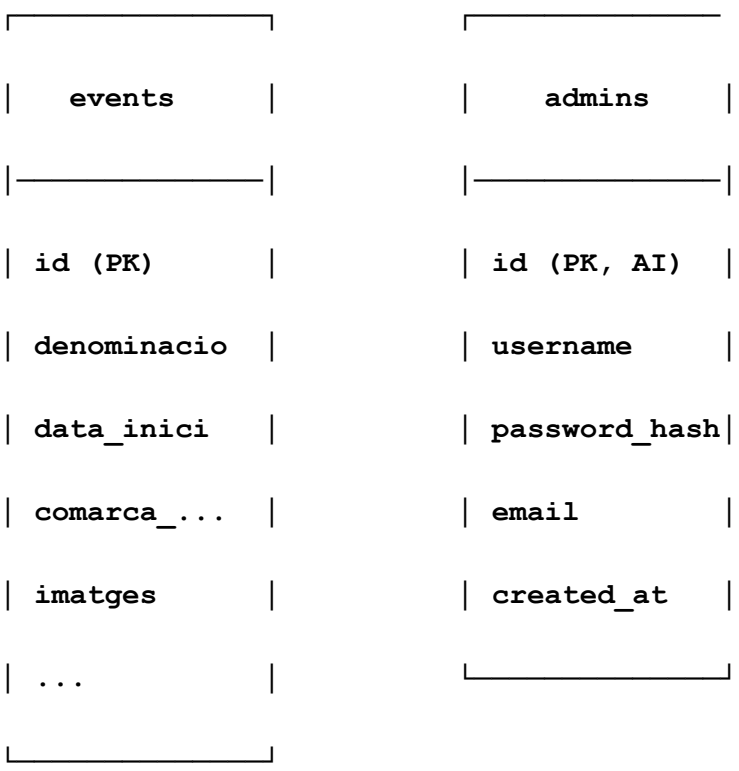
No existen claves foráneas ni relaciones directas entre ambas tablas, ya que cumplen funciones completamente independientes (contenido público vs. gestión interna del sistema).

- Interacción a nivel de aplicación: El administrador opera sobre la entidad events (inserción masiva y generación de estadísticas), pero la base de datos no registra propietarios ni vínculos relacionales.
- Nota sobre favoritos: La relación conceptual usuario ↔ evento favorito no se almacena en el servidor. Se gestiona mediante localStorage en el navegador, manteniendo un array de IDs (events.id) vinculado al dispositivo. Esto elimina la necesidad de una tercera tabla y simplifica la arquitectura.

Tabla resumen de campos principales

Entidad	Campo	Tipo	Clave	Descripción
events	id	TEXT	PK	Identificador único del evento (proporcionado por la fuente oficial)
events	denominacio	TEXT		Título o nombre oficial del evento
events	data_inici / data_fi	TEXT		Fechas de inicio y fin (formato ISO 8601)
events	tags_categories	TEXT		Clasificación temática (Teatre, Música, Exposicions, etc.)
events	comarca_i_municipi	TEXT		Ubicación geográfica y administrativa
events	imatges	TEXT		URL de la imagen principal (convertida a absoluta si era relativa)
events	entrades	TEXT		Información sobre precios, gratuidad o enlaces de compra
events	descripcio_html	TEXT		Contenido detallado en formato HTML seguro
admins	id	INTEGER	PK, AI	Identificador interno incremental del administrador

admins	username	TEXT	UNIQUE	Nombre de usuario para autenticación
admins	password_hash	TEXT		Contraseña cifrada mediante Werkzeug (PBKDF2/SHA-256)
admins	created_at	TIMESTAMP		Fecha de creación del perfil administrativo



Nota técnica: Ambas tablas son autónomas. La integración y el flujo de datos se resuelven en la capa de aplicación (Flask), no mediante restricciones relacionales en SQLite.

6. Diseño de la interfaz

Objetivo: visualizar la estructura y navegación del sistema antes de desarrollarlo.*

Pantalla 1: Listado de Eventos (Home)

Funcionalidad principal:

Página de inicio que muestra todos los eventos culturales disponibles en formato de tarjetas con scroll infinito. Incluye panel de filtros combinados (búsqueda textual, rango de fechas, precio y categoría) y botones para marcar favoritos. Cada tarjeta muestra imagen, título, fecha, ubicación y badge de gratuidad.

Casos de uso relacionados:

- CU1: Consultar eventos culturales
- CU2: Buscar y filtrar eventos
- CU3: Gestionar favoritos

Elementos visuales clave:

- Barra de navegación superior con menú responsive
- Panel de filtros colapsable en la parte superior
- Grid de 3 columnas (responsive: 1 columna en móvil)
- Spinner de carga en la parte inferior
- Botones de corazón para favoritos en cada tarjeta

Pantalla 2: Calendario Mensual

Funcionalidad principal:

Vista calendario que muestra los eventos distribuidos por días del mes. Permite navegación entre meses, visualización rápida de eventos por fecha y apertura de modal con lista detallada al hacer clic en un día. Incluye búsqueda contextual dentro del modal.

Casos de uso relacionados:

- CU1: Consultar eventos culturales
- CU2: Buscar y filtrar eventos

Elementos visuales clave:

- Cabecera con botones de navegación (mes anterior/siguiente)
- Grid de 7 columnas (días de la semana)
- Badges numéricos en días con eventos
- Modal emergente con lista de eventos del día seleccionado
- Resaltado del día actual

Pantalla 3: Detalle de Evento

Funcionalidad principal:

Muestra información completa de un evento específico: imagen destacada, descripción HTML enriquecida, fechas y horarios, ubicación con mapa embebido (texto), precio/entradas, enlaces externos y botón para añadir a favoritos.

Casos de uso relacionados:

- CU1: Consultar eventos culturales
- CU3: Gestionar favoritos

Elementos visuales clave:

- Layout de dos columnas (contenido principal + sidebar)
- Imagen principal en la parte superior
- Secciones diferenciadas con iconos (fecha, ubicación, descripción)
- Badges informativos (categoría, ámbito, modalidad)
- Botón de acción principal (visitar web oficial)
- Botón de favorito flotante

Pantalla 4: Mis Favoritos

Funcionalidad principal:

Listado personalizado de eventos marcados como favoritos por el usuario. Los datos se recuperan del localStorage del navegador y se muestran en el mismo formato de tarjetas que el listado principal. Incluye mensaje alternativo si no hay favoritos guardados.

Casos de uso relacionados:

- CU3: Gestionar favoritos

Elementos visuales clave:

- Grid de tarjetas similar al home
- Icono de corazón relleno en cada tarjeta
- Mensaje informativo si la lista está vacía
- Botón para eliminar de favoritos

Pantalla 5: Login / Iniciar Sesión

Funcionalidad principal:

Formulario de autenticación que permite el acceso como administrador (con validación de credenciales) o como usuario normal (solo nombre visual). Detecta automáticamente el tipo de usuario y redirige al panel correspondiente.

Casos de uso relacionados:

- CU4: Iniciar sesión como administrador

Elementos visuales clave:

- Formulario centrado con diseño de tarjeta
- Campos: nombre de usuario y contraseña
- Iconos en inputs (usuario y candado)
- Botón de acceso principal
- Mensajes de error/éxito con alertas dismissibles
- Diseño responsive y limpio

Pantalla 6: Panel de Administración

Funcionalidad principal:

Dashboard que muestra estadísticas en tiempo real de la base de datos: total de eventos, eventos próximos/pasados, distribución por año/comarca/categoría/precio, y últimos eventos añadidos. Incluye accesos directos a funcionalidades de gestión.

Casos de uso relacionados:

- CU4: Iniciar sesión como administrador
- CU5: Actualizar base de datos de eventos

Elementos visuales clave:

- Tarjetas de estadísticas principales (3 columnas)
- Tablas con scroll para distribuciones (año, comarca, categoría, precio)
- Tabla de últimos eventos añadidos
- Botones de acción rápida (actualizar BD, cambiar contraseña, volver)
- Menú dropdown en navbar con opciones de admin

Pantalla 7: Actualizar Base de Datos

Funcionalidad principal:

Interfaz para que el administrador cargue archivos JSON procedentes de la Agenda Cultural de Catalunya. Valida el archivo, lo procesa e inserta los nuevos eventos evitando duplicados.

Casos de uso relacionados:

- CU5: Actualizar base de datos de eventos

Elementos visuales clave:

- Formulario con input de tipo file
- Indicador de formatos aceptados (.json, .jsonld)

- Mensaje informativo sobre el proceso
- Botones de cancelar y subir archivo
- Alertas de éxito/error tras el procesamiento

Pantalla 8: Cambiar Contraseña

Funcionalidad principal:

Permite al administrador modificar su contraseña de acceso de forma segura. Valida la contraseña actual, verifica que la nueva cumpla los requisitos mínimos y confirma la coincidencia.

Casos de uso relacionados:

- CU4: Iniciar sesión como administrador

Elementos visuales clave:

- Tres campos: contraseña actual, nueva contraseña, confirmar nueva
- Indicador de longitud mínima (6 caracteres)
- Validación en tiempo real
- Botón de guardar cambios
- Mensajes de error específicos

Navegación global

Elementos comunes en todas las pantallas:

- Navbar superior: Logo, enlaces a Eventos/Calendario/Favoritos, menú de usuario/admin
- Footer: Información de copyright y enlaces básicos
- Sistema de alertas: Mensajes flash para feedback de acciones
- Diseño responsive: Adaptación automática a móvil/tablet/desktop

7. Planificación técnica

Objetivo: planificar el desarrollo del proyecto.*

- **Lenguajes y frameworks:**
  Backend: Python 3.12 con Flask 3.x (framework web ligero), Jinja2 (motor de plantillas) y Werkzeug (seguridad y utilidades WSGI).
  Frontend: HTML5, CSS3, JavaScript (Vanilla ES6+) y Bootstrap 5.3 (framework responsivo para maquetación y componentes UI).
  Infraestructura: Apache 2.4 con `mod_wsgi` para integración del servidor web con la aplicación Python.

- **Base de datos:**
  SQLite3: Sistema de gestión relacional embebido (archivo único `moute.db`). No requiere servidor independiente, lo que reduce la complejidad de despliegue y el consumo de recursos. Se estructura en dos tablas independientes (`events` para contenido cultural y `admins` para gestión de acceso), con consultas SQL estándar y manejo transaccional para garantizar integridad durante la importación masiva de JSON.

- **Herramientas de diseño o edición:**
  Entorno de desarrollo: Visual Studio Code con extensiones para Python, Git, SQLite y linting.

  Control de versiones: Git + GitHub (repositorio público, estrategia de commits atómicos y ramas por funcionalidad).

  Diagnóstico y pruebas: SQLite Browser (inspección de esquema y datos), logs de Apache (`/var/log/apache2/moute_error.log`), DevTools del navegador (Network, Console, Responsive Design Mode).

  Despliegue: Máquina virtual Ubuntu 24.04 (IsardVDI) con acceso vía bastión, terminal SSH, `systemctl` para gestión de servicios y `nano`/`vim` para edición remota de configuración.

- **Reparto de tareas (si es en grupo):**
  Al ser un desarrollo individual, el trabajo se ha organizado por módulos funcionales en lugar de por roles, priorizando dependencias técnicas y entregables incrementales:

1. Arquitectura y datos: Diseño del stack LASP, definición del esquema SQLite, estudio del formato JSON oficial y lógica de transformación.

2. Backend y seguridad: Implementación de rutas Flask, sistema de sesiones, autenticación de administrador, hashing de contraseñas y validación de entradas.

3. API interna y lógica de servidor: Desarrollo de endpoints REST (`/api/events`, `/api/search`, `/api/events-by-month`, `/api/filters`), paginación y construcción dinámica de consultas SQL.

4. Frontend e interactividad: Maquetación con Bootstrap, integración de Jinja2, desarrollo de lógica JS para filtros combinados, calendario mensual, scroll infinito y gestión de favoritos vía `localStorage`.

5. Despliegue y producción: Configuración de Virtual Host en Apache, integración `mod_wsgi`, ajuste de permisos (`www-data`), dominio bastión y optimización de carga estática.

6. Validación y documentación: Pruebas cross-device, depuración de errores, redacción de memoria técnica y documento funcional.

- **Cronograma (puede incluir un diagrama de Gantt):**

Desarrollo organizado en 6 fases secuenciales con solapamiento controlado entre backend y frontend. Representación aproximada (formato Gantt textual):

Fase	Actividad principal	Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Sem 6
F1	Investigación, análisis de requisitos y diseño de arquitectura	■■■■	■				
F2	Desarrollo backend (Flask, rutas, DB SQLite, autenticación)		■	■■■	■		
F3	API REST interna y procesamiento de datos JSON			■	■■■		
F4	Desarrollo frontend (Bootstrap, Jinja2, JS interactivo)				■	■■■	■
F5	Integración, despliegue en VM y configuración Apache/WSGI					■	■■■
F6	Pruebas finales, depuración y documentación						■■■

8. Análisis de riesgos

Objetivo: identificar posibles problemas y cómo se afrontarán.*

8.1. Identificación de riesgos

Ejemplos:

- Los principales riesgos detectados para el proyecto Moute incluyen:

- Cambios en la estructura de datos fuente: La Generalitat de Catalunya podría modificar el formato JSON de la Agenda Cultural, rompiendo el proceso de importación.

- Problemas técnicos en el despliegue: Configuración compleja de Apache/mod_wsgi en Ubuntu 24, incompatibilidades de versiones o caídas del servidor.

- Pérdida o corrupción de datos: Fallos durante la importación masiva, eliminación accidental de la base de datos o problemas de integridad.

- Dependencia de recursos externos: Las imágenes se cargan desde servidores de la Generalitat (agenda.cultura.gencat.cat); si cambian las URLs o eliminan imágenes, la interfaz mostrará errores.

- Rendimiento con grandes volúmenes de datos: La base de datos podría crecer significativamente, afectando tiempos de respuesta en consultas y filtros.

- Vulnerabilidades de seguridad: Acceso no autorizado al panel de administración, inyecciones SQL o exposición de credenciales.

- Falta de adopción por parte de usuarios: Dificultad de uso, interfaz poco intuitiva o competencia con la web oficial.

- Limitaciones de tiempo y recursos: Desarrollo individual con plazo académico limitado, curva de aprendizaje de tecnologías nuevas.

8.2. Valoración y respuesta

Clasifica cada riesgo según su probabilidad e impacto, e indica cómo se mitigará.*Clasificación de cada riesgo según su probabilidad e impacto, e indicación de cómo se mitigará.*

Riesgo	Probabilidad	Impacto	Plan de prevención o contingencia
Cambios en el formato JSON de la fuente oficial	Media	Alta	Implementar validación robusta del JSON con manejo de excepciones. Documentar la estructura esperada y mantener un archivo de ejemplo. Si cambia el formato, adaptar el parser en `update_db_from_file()` verificando campos obligatorios antes de insertar.
Problemas técnicos en configuración Apache/WSGI	Alta	Alta	Seguir documentación oficial de mod_wsgi y Ubuntu. Realizar pruebas en entorno controlado antes de producción. Mantener logs de error activos y configuraciones versionadas en Git. Tener un plan de rollback a configuración anterior funcional.
Pérdida o corrupción de la base de datos	Baja	Crítico	Realizar copias de seguridad automáticas semanales de `moute.db` mediante script cron. Almacenar backups en ubicación externa. Implementar transacciones SQL atómicas durante importaciones (commit solo si todo es correcto).
Imágenes externas no disponibles	Media	Media	Implementar manejo de errores en etiquetas `<img>` con atributo `onerror` que cargue imagen placeholder. Considerar futuro sistema de caché local de imágenes críticas.

Degradación del rendimiento con muchos eventos	Media	Media	Implementar índices SQL en columnas frecuentemente consultadas (`data_inici`, `tags_categories`, `comarca`). Mantener paginación estricta (20 eventos/página). Optimizar consultas con `EXPLAIN QUERY PLAN`. Considerar caché de resultados para peticiones repetidas.
Acceso no autorizado al panel de administración	Baja	Crítico	Usar hashing seguro con Werkzeug (PBKDF2/SHA-256). Implementar sesiones HTTPS-only en producción. Validar y sanitizar todas las entradas de usuario. Proteger rutas con decorador `@admin_required`. Cambiar contraseña por defecto inmediatamente tras despliegue.
Falta de usabilidad o adopción por usuarios	Media	Media	Realizar pruebas de usabilidad con dispositivos reales (móvil, tablet, desktop). Seguir estándares de Bootstrap para patrones de diseño conocidos. Incluir feedback visual inmediato (alertas, spinners, mensajes de confirmación). Mantener interfaz limpia y navegación intuitiva.
Limitaciones de tiempo por curva de aprendizaje	Alta	Media	Priorizar funcionalidades esenciales (RF1-RF4) antes de características avanzadas. Utilizar tecnologías con amplia documentación (Flask, Bootstrap). Aprovechar ejemplos y repositorios públicos como referencia. Mantener un registro de problemas y soluciones (debugging log).

Caída del servidor o indisponibilidad del servicio	Baja	Alta	Configurar reinicio automático de Apache (`systemctl enable apache2`). Monitorear logs de error regularmente. Tener acceso SSH de emergencia vía bastión. Documentar procedimiento de reinicio y verificación de servicios.
Incompatibilidad entre versiones de dependencias	Media	Media	Usar entorno virtual Python (`venv`) para aislar dependencias. Documentar versiones exactas en `requirements.txt`. Verificar compatibilidad entre Python 3.12, Flask 3.x y mod_wsgi antes de actualizar.

9. Validación y criterios de éxito

Objetivo: definir cómo sabremos que el proyecto funciona correctamente.*

- **Criterios de aceptación (qué debe cumplirse para darlo por válido).**

  - El proyecto se considerará validado y operativo cuando se cumplan los siguientes requisitos mínimos:

  - El sistema importa archivos JSON oficiales de la Agenda Cultural sin duplicar registros existentes y gestiona campos opcionales o mal formateados sin interrumpir el flujo.

  - La interfaz principal muestra eventos correctamente en dos modos: listado dinámico con scroll infinito y calendario mensual interactivo.

  - Los filtros combinados (búsqueda textual, rango de fechas, precio y categoría) devuelven resultados precisos y se actualizan sin recargar la página.

  - Los favoritos se marcan, desmarcan y persisten correctamente mediante `localStorage`, sincronizándose con la sección dedicada.

  - El panel de administración es accesible únicamente con credenciales válidas, genera estadísticas en tiempo real y permite el cambio seguro de contraseña.

  - La aplicación es completamente responsiva y se adapta visual y funcionalmente a móviles, tablets y escritorio.
  - El despliegue es accesible públicamente a través del dominio proporcionado por el servicio de bastión, sin errores 500 o 404 en rutas principales.

- **Pruebas previstas (funcionales, de usuario, de rendimiento).**
    - Pruebas funcionales: Verificación sistemática de cada requisito funcional (RF1-RF10). Se ejecutará un ciclo completo: carga de JSON de prueba → validación en SQLite → visualización en frontend → aplicación de filtros → login como admin → actualización de BD → cambio de contraseña → logout. Cada paso se documentará con capturas y registros de consola.

    - Pruebas de usabilidad: Navegación guiada por perfiles (visitante, usuario, administrador). Se evaluará la claridad de la interfaz, la retroalimentación visual (alertas, spinners, estados de carga), la intuitividad del calendario modal y la adaptación del menú en pantallas ≤ 768px.

    - Pruebas de rendimiento: Medición de tiempos de carga inicial y latencia de las peticiones AJAX (`/api/events`, `/api/search`). Se verificará que el scroll infinito no bloquee el hilo principal y que las consultas SQL con múltiples filtros respondan en menos de 2 segundos para conjuntos de hasta 1.000 registros.

- **Indicadores de calidad o resultados esperados.**
    - Tasa de éxito en importación: ≥ 95 % de registros válidos procesados; manejo *graceful* de excepciones JSON sin caída del servicio.

    - Rendimiento: Tiempo de carga inicial < 3 segundos; respuesta de la API < 1 segundo bajo condiciones de red estándar (4G/Wi-Fi).

    - Compatibilidad: Funcionamiento correcto en las últimas dos versiones de Chrome, Firefox, Safari y Edge; adaptación fluida desde 320px hasta resoluciones 4K.

    - Seguridad y estabilidad: Cero vulnerabilidades de inyección SQL o exposición de credenciales; contraseñas administrativas almacenadas con hash seguro (Werkzeug/PBKDF2); disponibilidad del servicio ≥ 95 % durante el período de evaluación académica.

    - Experiencia de usuario: Navegación asíncrona (sin recargas completas), feedback inmediato en acciones interactivas y coherencia visual con el sistema de diseño definido (Bootstrap 5 + tema corporativo púrpura).

---

10. Conclusión

Objetivo: cerrar el análisis y preparar la siguiente fase.*

Resume las decisiones principales tomadas durante el análisis:

- **Qué funciones tendrá el sistema.**
  El sistema integrará dos niveles de interacción claramente diferenciados: un acceso público para consultar eventos en listado o calendario, aplicar filtros combinados (fecha, precio, categoría, texto), visualizar detalles completos y gestionar favoritos con persistencia local; y un panel de administración protegido que permitirá iniciar sesión con credenciales seguras, cargar archivos JSON oficiales para actualizar la base de datos evitando duplicados, visualizar estadísticas en tiempo real (distribución por año, comarca, categoría y precio) y modificar la contraseña de acceso.

- **Qué tecnologías se utilizarán.**
  Se ha optado por un stack ligero, estable y de bajo mantenimiento: Python 3.12 con Flask 3.x para la lógica de servidor y enrutamiento; SQLite3 como base de datos relacional embebida; Apache 2.4 + mod_wsgi para el despliegue productivo; y Bootstrap 5 + JavaScript vanilla para una interfaz responsiva y asíncrona. El control de versiones se gestionará con Git/GitHub, y la infraestructura se desplegará en una máquina virtual Ubuntu 24.04 con acceso mediante bastión.

- **Qué valor aporta el proyecto.**
  Moute cierra la brecha entre los datos abiertos institucionales y la ciudadanía. Al transformar formatos técnicos (JSON) en una experiencia de navegación intuitiva, accesible desde cualquier dispositivo y centrada en la usabilidad, el proyecto democratiza el acceso a la programación cultural, reduce la fricción en la búsqueda de actividades y demuestra cómo las tecnologías de código abierto pueden servir al interés público sin depender de infraestructuras complejas o licencias comerciales.

Indica los próximos pasos:

- **Preparar el entorno de desarrollo.**
  Instalar Python 3.12, configurar el entorno virtual (`venv`), instalar dependencias base (`Flask`, `Werkzeug`, `sqlite3`) y validar la conectividad SSH con la máquina virtual de producción.

- **Crear la estructura inicial del repositorio y la base de datos.**
  Inicializar Git, organizar los directorios del proyecto (`app.py`, `templates/`, `static/`, `uploads/`), definir el esquema SQLite (`events`, `admins`) y ejecutar el script de inicialización con un dataset de prueba.

- **Empezar la implementación de los casos de uso prioritarios.**
  Desarrollar primero el motor de importación JSON y la API REST interna (`/api/events`, `/api/search`), seguido del frontend básico (listado, filtros y vista de detalle), para finalmente integrar el calendario mensual, la gestión de favoritos y el panel de administración.

El análisis funcional proporciona una visión clara, realista y técnica del proyecto. A partir de aquí, comienza la fase de desarrollo: convertir especificaciones en código, pruebas en estabilidad y datos públicos en una herramienta accesible al servicio de la cultura.