UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

REPRESENTACIÓN AVANZADA DEL CONOCIMIENTO Y RAZONAMIENTO

Proyecto Final - Aplicación RDF/SPARQL

Integrantes:

- BERMEO CABRERA DIANA ROCIO

Docente:

Ing. Nelson Piedra.

LOJA

2025

Contenido

Resumen

El presente informe detalla el desarrollo e implementación de CulturaViva, una aplicación web interactiva diseñada para conectar a la comunidad con su patrimonio cultural tangible e intangible. Abordando la problemática de la información cultural dispersa, inaccesible y subvalorada, CulturaViva propone una solución innovadora mediante la integración y visualización de Linked Open Data (LOD) de fuentes como DBpedia y Wikidata. Utilizando Python y el framework Streamlit, la aplicación transforma datos complejos en visualizaciones dinámicas (mapas, líneas de tiempo, gráficos de red) que facilitan la exploración de lugares, personalidades, músicos, eventos históricos y sitios UNESCO. Este proyecto demuestra el potencial de las tecnologías de la Web Semántica para democratizar el acceso al conocimiento cultural, fortalecer la identidad local y fomentar la reutilización de datos abiertos, brindando una herramienta poderosa para la educación y el turismo cultural. Se describe la arquitectura, las tecnologías empleadas, la metodología de desarrollo y los resultados obtenidos, concluyendo con las limitaciones y futuras mejoras.

Abstract

This report details the development and implementation of CulturaViva, an interactive web application designed to connect the community with its tangible and intangible cultural heritage. Addressing the problem of dispersed, inaccessible, and undervalued cultural information, CulturaViva proposes an innovative solution through the integration and visualization of Linked Open Data (LOD) from sources such as DBpedia and Wikidata. Using Python and the Streamlit framework, the application transforms complex data into dynamic visualizations (maps, timelines, network graphs) that facilitate the exploration of places, personalities, musicians, historical events, and UNESCO sites. This project demonstrates the potential of Semantic Web technologies to democratize access to cultural knowledge, strengthen local identity, and encourage the reuse of open data, providing a powerful tool for cultural education and tourism. The architecture, technologies used, development methodology and results obtained are described, concluding with limitations and future improvements.

Capítulo 1: Introducción

1.1. Planteamiento del problema

La vasta riqueza del patrimonio cultural, tanto tangible (monumentos, sitios históricos) como intangible (biografías de figuras influyentes, eventos significativos, expresiones artísticas), se encuentra actualmente fragmentada, dispersa en múltiples repositorios y, a menudo, inaccesible para el público general. Esta desorganización y la falta de un punto de acceso unificado resultan en una profunda desconexión cultural. La limitada visibilidad de estos valiosos activos mermó su potencial educativo y de identidad, dificultando su estudio, apreciación y preservación. En consecuencia, se observa un impacto negativo en la educación, al no poder los estudiantes y ciudadanos acceder fácilmente a su historia y raíces; un debilitamiento del sentido de pertenencia, al no conocer o comprender la relevancia de su propio patrimonio; y una subutilización del vasto potencial turístico y cultural que este patrimonio podría ofrecer a la comunidad.

1.2. Justificación del proyecto

El proyecto CulturaViva se justifica por la imperante necesidad de transformar el acceso y la interacción con el patrimonio cultural. La aplicación web propuesta aborda directamente la problemática al:

- Centralizar y organizar información cultural diversa: Agregando datos de múltiples fuentes de conocimiento abierto y bases de datos enlazadas (Linked Open Data - LOD).
- Transformar la complejidad en claridad: Convirtiendo datos complejos y estructurados en visualizaciones dinámicas e intuitivas.
- Facilitar el acceso: Ofreciendo una interfaz de usuario amigable y opciones de búsqueda que permiten una exploración temática y geográfica del patrimonio.
- Revelar las interconexiones: Al utilizar LOD, la aplicación no solo presenta datos aislados, sino que destaca las relaciones inherentes entre ellos, proporcionando un contexto más rico y profundo. Los beneficios esperados son significativos: promover la educación abierta al hacer el conocimiento cultural accesible a todos; fortalecer la identidad cultural conectando a las generaciones con su legado; fomentar la reutilización de datos públicos,

contribuyendo a un ecosistema digital colaborativo; y revalorizar el patrimonio olvidado, dándole la visibilidad que merece. CulturaViva, en esencia, busca hacer que el patrimonio "cobre vida" en la era digital.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar una aplicación web interactiva que facilite la exploración y comprensión del patrimonio cultural tangible e intangible, utilizando el paradigma de Linked Open Data (LOD) para ofrecer una experiencia de usuario rica y educativa

1.3.2. Objetivos específicos

- Diseñar e implementar una interfaz de usuario intuitiva y atractiva utilizando el framework Streamlit, que permita una navegación sencilla a través de diversas categorías culturales.
- Desarrollar un módulo de servicios de datos que encapsule las funciones para construir y ejecutar consultas SPARQL eficientes contra endpoints de Linked Open Data como DBpedia y Wikidata.
- Crear visualizaciones de datos dinámicas e interactivas, incluyendo mapas geográficos con Folium para ubicar lugares de interés, líneas de tiempo con Plotly para contextualizar eventos históricos, y gráficos de red con Plotly para ilustrar relaciones de influencia entre entidades.
- Integrar funcionalidades de búsqueda y filtrado de contenido que permitan a los usuarios refinar la información cultural por categorías específicas (ej. Lugares de Ecuador, Personalidades Ecuatorianas, Músicos, Conflictos/Guerras Globales, Sitios del Patrimonio Mundial UNESCO, Gráfico de Influencias).

1.4. Alcance y limitaciones

Alcance:

Abarca el diseño y desarrollo de una aplicación web para la consulta y visualización de datos de patrimonio cultural, haciendo uso exclusivo de información disponible a través de Linked Open Data. Las funcionalidades implementadas incluyen la exploración de lugares geográficos, biografías de personalidades, datos sobre músicos, eventos de conflictos globales y sitios

UNESCO, así como un innovador gráfico de relaciones de influencia. La aplicación se enfoca en la presentación de información existente, ofreciendo herramientas interactivas de filtrado y visualización.

Limitaciones:

La calidad, completitud y disponibilidad de la información mostrada dependen directamente de los datos alojados en los *endpoints* de DBpedia y Wikidata. Las consultas SPARQL están optimizadas para un límite de resultados y, aunque amplias, no pretenden cubrir todos los posibles aspectos de un tema cultural. El proyecto no incluye mecanismos de autenticación de usuarios, funcionalidades de gestión o edición de contenidos, ni herramientas de contribución de datos por parte de los usuarios. Es una aplicación de solo lectura, cuyo objetivo principal es la diseminación del conocimiento existente.

1.5. Metodología de desarrollo

Dado el enfoque en la rápida creación de prototipos funcionales, la alta interacción con APIs externas y la necesidad de visualización constante de los resultados, se adoptó una metodología de desarrollo ágil, particularmente un enfoque de prototipos iterativos. Esta aproximación permitió construir la interfaz de usuario con Streamlit de manera incremental, probar y refinar las consultas SPARQL en cada iteración, e integrar las visualizaciones de datos de forma flexible. Este método facilitó la adaptación a las características y limitaciones de los datos disponibles en los LOD, permitiendo ajustes rápidos en el diseño y las funcionalidades basadas en los resultados y la retroalimentación obtenida en cada fase.

Capítulo 2: Marco Teórico

2.1. Fundamentos del problema o dominio

El dominio fundamental del proyecto es el patrimonio cultural, un concepto amplio que abarca tanto los elementos tangibles (sitios arqueológicos, monumentos, obras de arte) como los intangibles (tradiciones, biografías de figuras históricas, expresiones musicales). El proyecto busca abordar el desafío de la disgregación cultural en la era digital, transformando la información dispersa en un conocimiento accesible e interconectado. Esto se logra mediante la aplicación de principios de la

Web Semántica y los Linked Open Data (LOD), lo que permite integrar, consultar y visualizar vastos conjuntos de datos culturales para fomentar la educación, la investigación y la identidad local.

2.2. Tecnologías utilizadas

- **Lenguaje de Programación: Python**. Elegido por su versatilidad, su amplia comunidad y el vasto ecosistema de librerías para la ciencia de datos y el desarrollo web.
- **Frameworks y Librerías:**
    - **Streamlit:** Framework de código abierto en Python para la creación rápida de aplicaciones web interactivas y paneles de control sin necesidad de conocimientos de frontend (HTML, CSS, JavaScript). Es fundamental para la construcción de la interfaz de usuario (app.py) [Streamlit, s.f.a].
    - **Requests:** Librería HTTP para Python que simplifica la realización de peticiones web [Requests, s.f.]. Es utilizada extensivamente en sparql_queries.py para interactuar con los *endpoints* SPARQL de DBpedia y Wikidata.
    - **Folium:** Librería de Python para crear mapas interactivos que visualizan datos geoespaciales, basándose en la librería JavaScript Leaflet.js [Folium, s.f.]. Empleada en app.py para la visualización de lugares de interés y sitios UNESCO.
    - **Streamlit-Folium:** Un componente que permite integrar los objetos de mapa creados con Folium directamente en aplicaciones Streamlit [Streamlit, s.f.b].
    - **Plotly (Plotly Express y Plotly Graph Objects):** Potente librería de código abierto para crear gráficos interactivos y visualizaciones de datos avanzadas en Python [Plotly, s.f.a; Plotly, s.f.b]. Utilizada en app.py para generar líneas de tiempo de conflictos globales y gráficos de red de influencias.

    - **Pandas:** Librería fundamental para el análisis y manipulación de datos en Python, proporcionando estructuras de datos flexibles como los DataFrame [Pandas, s.f.]. Esencial para procesar y estructurar los resultados obtenidos de las consultas SPARQL en app.py.
    - **datetime y random:** Módulos estándar de Python. datetime se utiliza para el manejo y formateo de fechas, crucial para las líneas de tiempo y la presentación de información biográfica. random se emplea en funcionalidades auxiliares, como la selección aleatoria de elementos en la sección "Sabías que..." en app.py.
- **Bases de Datos (End-points LOD):**
    - **DBpedia:** Una de las principales bases de conocimiento extraídas de Wikipedia, que publica datos estructurados bajo licencias abiertas y permite el acceso vía SPARQL [DBpedia, s.f.]. Es el origen de datos para lugares y relaciones de influencia.
    - **Wikidata:** Una base de conocimiento libre y colaborativa alojada por la Fundación Wikimedia, que actúa como repositorio central de datos estructurados para todos los proyectos de Wikimedia y más allá [Wikidata, s.f.b]. Es la fuente principal para personalidades, músicos, conflictos y sitios UNESCO.
- **Lenguaje de Consulta: SPARQL**. Lenguaje estándar del W3C para consultar y manipular datos almacenados en el formato RDF (Resource Description Framework) [W3C, 2013]. Fundamental para la interacción con DBpedia y Wikidata, como se ve en sparql_queries.py

2.3. Conceptos clave

Desarrollo Web: Proceso de creación de aplicaciones que residen en servidores y son accesibles a través de un navegador web. CulturaViva es una aplicación web que utiliza Streamlit para abstraer la complejidad del desarrollo frontend tradicional.

Linked Open Data (LOD): Un conjunto de principios para publicar datos estructurados en la Web de manera interconectada, formando una "Web de Datos". Los LOD permiten que CulturaViva acceda a información rica y semánticamente

conectada de diversas fuentes públicas, facilitando la exploración de relaciones entre entidades.

SPARQL (SPARQL Protocol and RDF Query Language): Es el lenguaje de consulta estándar para la Web Semántica, diseñado para recuperar y manipular información almacenada en grafos RDF. Permite a CulturaViva formular preguntas complejas a bases de conocimiento como DBpedia y Wikidata [W3C, 2013].

API (Application Programming Interface): Un conjunto de definiciones y protocolos que se utiliza para construir e integrar *software* de aplicaciones. Los *endpoints* SPARQL de DBpedia y Wikidata actúan como APIs, permitiendo a sparql_queries.py programáticamente solicitar y recibir datos.

Caché de Datos: Un mecanismo para almacenar temporalmente los resultados de operaciones costosas (como consultas a bases de datos remotas) para que las solicitudes futuras de los mismos datos puedan ser servidas más rápidamente. La anotación @st.cache_data(ttl=3600) en sparql_queries.py implementa esta estrategia para mejorar significativamente el rendimiento de CulturaViva, al evitar consultas repetidas a los *endpoints* de LOD y reducir la latencia [Streamlit, s.f.a].

Capítulo 3: Análisis del Sistema

3.1. Requisitos funcionales y no funcionales

Requisitos Funcionales (RF):

RF1: Exploración de Lugares Culturales: El sistema debe permitir al usuario visualizar lugares de interés en Ecuador en un mapa interactivo, con la opción de filtrar por ciudad (sparql_queries.py:get_monuments_or_places_in_ecuador).

RF2: Consulta de Personalidades Ecuatorianas: La aplicación debe mostrar información detallada (biografía, fechas, imágenes) sobre personalidades destacadas de Ecuador, permitiendo búsquedas por nombre o concepto (sparql_queries.py:get_ecuadorian_personalities).

RF3: Visualización de Músicos Ecuatorianos: El sistema debe presentar un listado de músicos ecuatorianos con sus detalles relevantes, ofreciendo una funcionalidad de búsqueda (sparql_queries.py:get_ecuadorian_musicians).

RF4: Línea de Tiempo de Conflictos Globales: Debe generar y mostrar una línea de tiempo interactiva de guerras y conflictos a nivel global, con información contextual y opciones de búsqueda (sparql_queries.py:get_global_wars_and_conflicts).

RF5: Exploración de Patrimonio UNESCO: La aplicación debe permitir la exploración de Sitios del Patrimonio de la Humanidad de la UNESCO, incluyendo imágenes y coordenadas geográficas (sparql_queries.py:get_unesco_world_heritage_sites).

RF6: Gráfico de Relaciones de Influencia: El sistema debe construir y visualizar un gráfico de red que muestre las relaciones de influencia entre personalidades, particularmente aquellas con origen en Ecuador (sparql_queries.py:get_influencer_relationships).

RF7: Búsqueda y Filtrado General: La aplicación debe proporcionar campos de texto para que los usuarios puedan buscar y filtrar resultados dentro de cada categoría cultural.**RF8: Presentación Detallada de Entidades:** Para cada entidad cultural, el sistema debe mostrar atributos clave como nombre, descripción, fechas relevantes, ubicaciones e imágenes si están disponibles.

Requisitos No Funcionales (RNF):

RNF1: Usabilidad: La interfaz de usuario debe ser intuitiva, fácil de navegar y visualmente atractiva, permitiendo a usuarios con diferentes niveles de habilidad acceder a la información sin dificultad (app.py, config.toml).

RNF2: Rendimiento: Las consultas a los *endpoints* LOD y la carga de visualizaciones deben ser eficientes, con tiempos de respuesta aceptables (optimización de consultas SPARQL, uso de caché en sparql_queries.py).

RNF3: Fiabilidad: El sistema debe manejar robustamente errores en las peticiones a las APIs externas y en el procesamiento de datos (try-except en sparql_queries.py y app.py).

RNF4: Disponibilidad: La funcionalidad del sistema depende directamente de la disponibilidad y el tiempo de respuesta de los *endpoints* públicos de DBpedia y Wikidata.

RNF5: Mantenibilidad: El código debe estar modularizado y bien estructurado (separación de lógica de UI y lógica de consultas) para facilitar futuras actualizaciones y extensiones.

RNF6: Estética: La aplicación debe presentar un diseño visual coherente y profesional, con una paleta de colores y tipografía que realcen el contenido cultural (config.toml).

3.2. Diagramas

3.2.1. Diagrama de clases

Representaría las principales clases o módulos lógicos:

- App (representando la lógica en app.py), con métodos para renderizar la UI, manejar la entrada del usuario, procesar datos y orquestar visualizaciones.
- SparqlQueries (representando las funciones en sparql_queries.py), con métodos para run_sparql_query, get_monuments_or_places_in_ecuador, etc.
- Clases de librerías externas (ej. requests.Response, pandas.DataFrame, folium.Map, plotly.graph_objects.Figure) como dependencias.

3.2.2. Diagrama de Secuencia

Mostraría el flujo de ejecución para un caso de uso específico, por ejemplo, "Explorar Lugares de Interés":

- Usuario interactúa con la Interfaz de Usuario (en app.py).
- Interfaz de Usuario llama a SparqlQueries.get_monuments_or_places_in_ecuador().
- SparqlQueries.get_monuments_or_places_in_ecuador() llama a SparqlQueries.run_sparql_query().
- SparqlQueries.run_sparql_query() realiza una HTTP GET Request a DBpedia Endpoint.

- DBpedia Endpoint devuelve una JSON Response.
- SparqlQueries.run_sparql_query() retorna los Results a SparqlQueries.get_monuments_or_places_in_ecuador().
- SparqlQueries.get_monuments_or_places_in_ecuador() retorna los Results a la Interfaz de Usuario.
- Interfaz de Usuario procesa los Results con Pandas y los visualiza con Folium.

3.3. Modelo de datos

El sistema CulturaViva no posee un modelo de datos relacional interno ni una base de datos propia en el sentido tradicional. En su lugar, el sistema opera sobre el modelo de datos intrínseco de los Linked Open Data (ontologías RDF) de DBpedia y Wikidata. Las "entidades" del dominio (ej., dbo:Place, wd:Q5 para Humano, wd:Q198 para Guerra) y sus "relaciones" (propiedades como geo:lat, dbo:country, wdt:P27 para Nacionalidad, dbo:influenced) son definidas y gestionadas por estos *endpoints* externos. Las consultas SPARQL (sparql_queries.py) son diseñadas para navegar y extraer subgrafos de conocimiento de estos grandes grafos de datos, según los requisitos de cada sección de la aplicación.

Capítulo 4: Diseño del Sistema

4.1. Arquitectura del software

La arquitectura de CulturaViva se puede describir como una variante de la arquitectura **Cliente-Servidor / Tres Capas (Lógica)**, adaptada a la naturaleza de las aplicaciones basadas en Streamlit que consumen datos de fuentes externas:

- **Capa de Presentación (Frontend / Cliente):** Implementada principalmente por **app.py y el entorno Streamlit**. Esta capa es responsable de renderizar la interfaz de usuario web, manejar las interacciones del usuario (clicks, entradas de texto), y mostrar las visualizaciones de datos. Streamlit abstrae la complejidad del desarrollo web tradicional, permitiendo construir la UI directamente con código Python.
- **Capa de Lógica de Aplicación (Backend / Servidor Lógico):** Reside en **app.py**. Esta capa gestiona la lógica de negocio de la aplicación: la navegación

entre secciones, el procesamiento intermedio de los datos obtenidos (ej., filtrado con Pandas, formateo con datetime), la orquestación de las llamadas a la capa de acceso a datos, y la preparación de los datos para las visualizaciones (folium, plotly).

- **Capa de Acceso a Datos:** Encapsulada en **sparql_queries.py**. Este módulo es el intermediario entre la lógica de la aplicación y las fuentes de datos externas. Contiene las funciones específicas para construir las consultas SPARQL, realizar las peticiones HTTP (requests) a los *endpoints* de LOD, manejar las respuestas y aplicar estrategias de caché (@st.cache_data).
- **Capa de Datos (Servidor Externo):** Constituyen los **repositorios de Linked Open Data (DBpedia y Wikidata)**. Estos son los servidores remotos que almacenan y exponen el vasto conocimiento cultural mediante *endpoints* SPARQL. La aplicación consume estos datos a demanda, sin mantener una base de datos local replicada.

4.2. Diseño de la base de datos

El diseño de la base de datos es externalizado y distribuido, ya que CulturaViva no aloja una base de datos relacional propia. En su lugar, el sistema interactúa con los modelos de datos basados en grafos (ontologías RDF) de las bases de conocimiento externas: DBpedia y Wikidata.

El "esquema" de datos que utiliza la aplicación está implícitamente definido por las propiedades y clases RDF a las que se hace referencia en las consultas SPARQL (ej., dbo:abstract, wdt:P18 para imagen, rdfs:label, etc.). Este enfoque permite aprovechar la vasta cantidad de datos interconectados ya existentes en la Web Semántica.

4.3. Diseño de la interfaz de usuario

La interfaz de usuario de CulturaViva fue diseñada para ser limpia, funcional e intuitiva, aprovechando las capacidades de Streamlit:

- **Navegación:** Una **barra lateral (st.sidebar)** centraliza las opciones de navegación principal, permitiendo al usuario seleccionar fácilmente la categoría de patrimonio cultural que desea explorar (Inicio, Lugares,

Personalidades, etc.). También contiene los campos de búsqueda específicos para cada sección.

- **Área Principal de Contenido:** El área principal de la aplicación (st.container, st.columns) se actualiza dinámicamente para mostrar el contenido relevante según la selección del usuario.
- **Componentes Interactivos:** Se utilizan widgets de Streamlit como st.radio para la selección de categoría, st.text_input para la entrada de búsqueda, st.expander para secciones expandibles, y st.image para mostrar miniaturas.
- **Visualizaciones Específicas:**
    - **Mapas:** Integración de mapas interactivos de Folium (st_folium) para visualizar ubicaciones geográficas con marcadores y *pop-ups* informativos.
    - **Gráficos Interactivos:** Uso de Plotly (st.plotly_chart) para generar líneas de tiempo dinámicas, permitiendo el *zoom* y la interacción, y complejos gráficos de red para visualizar relaciones.
- **Estilo y Branding:** La apariencia visual de la aplicación se personaliza a través de config.toml, permitiendo definir colores, fuentes y otros elementos estéticos para crear una experiencia de usuario cohesionada y culturalmente relevante. Se prioriza una paleta de colores cálidos y una tipografía clara para mejorar la legibilidad y la experiencia.

4.4. Selección de herramientas de desarrollo

Lenguaje de Programación: Python 3.13.5

La elección de Python y el conjunto específico de librerías para CulturaViva se basó en los siguientes criterios clave:

- **Productividad y Rapidez de Prototipado:** Streamlit permite un desarrollo ágil y la creación rápida de prototipos funcionales de aplicaciones web con código Python mínimo, lo que fue crucial para este proyecto.
- **Manejo de Datos y Computación Científica:** Python, junto con Pandas, es un estándar de la industria para el análisis, manipulación y procesamiento de

grandes conjuntos de datos, lo cual es esencial al trabajar con resultados de SPARQL.

- **Capacidades de Visualización Avanzadas:** Folium y Plotly son librerías líderes en sus respectivos campos (mapas geoespaciales interactivos y gráficos de datos complejos), ofreciendo alta interactividad y calidad visual.
- **Facilidad de Integración con APIs:** La librería requests en Python simplifica enormemente la interacción con APIs RESTful y *endpoints* SPARQL, que son el corazón de la obtención de datos en este proyecto.
- **Comunidad y Soporte:** Todas las tecnologías seleccionadas cuentan con comunidades activas y amplia documentación, lo que facilita el aprendizaje, la resolución de problemas y la escalabilidad.

4.5. Seguridad, validaciones y control de errores

Control de Errores: Se implementaron mecanismos robustos de manejo de errores utilizando bloques try-except en sparql_queries.py (específicamente en run_sparql_query) para capturar y gestionar fallos en las peticiones HTTP (ej., problemas de red, *timeouts*, errores HTTP 4xx/5xx al conectar con los *endpoints* SPARQL). En caso de error, se muestra un mensaje informativo al usuario mediante st.error(). Adicionalmente, en app.py, se utilizan try-except al parsear y formatear datos (ej., fechas con datetime, coordenadas geográficas), para manejar inconsistencias o datos faltantes y evitar que la aplicación se detenga.

Validaciones: Las validaciones de entrada de usuario son básicas y se limitan a la funcionalidad de los campos de texto (st.text_input) para búsquedas. No se implementan validaciones complejas a nivel de formularios, ya que la aplicación es de solo lectura y no permite la entrada de datos sensibles o la creación de contenido por parte del usuario.

Seguridad: Dada la naturaleza de la aplicación (consumo de datos públicos de APIs externas y no manejo de información personal o credenciales de usuario), las principales consideraciones de seguridad se centran en el manejo seguro de las peticiones a los *endpoints* externos y la resiliencia ante datos inesperados.

Capítulo 5: Implementación

5.1. Entorno de desarrollo

El desarrollo de CulturaViva se llevó a cabo en un entorno de programación Python estándar.

- **IDE (Entorno de Desarrollo Integrado):** Se utilizó Visual Studio Code (u otro IDE compatible con Python) para la edición del código, depuración y gestión del proyecto.
- **Versión de Python:** Se empleó Python 3.9 o superior, compatible con todas las librerías listadas.
- **Administrador de Paquetes:** pip (Python Package Installer) se utilizó para gestionar las dependencias del proyecto, asegurando que todas las librerías necesarias estuvieran instaladas con las versiones correctas.

5.2. Estructura de carpetas del proyecto

La estructura del proyecto es simple y modular:

/OnePageCultura/

├── app.py # Script principal de la aplicación Streamlit (Frontend y Lógica de Negocio)

├── sparql_queries.py # Módulo de funciones para consultas SPARQL y acceso a datos

├── requirements.txt # Archivo que lista todas las dependencias de Python del proyecto

├── .streamlit

└── config.toml # Archivo de configuración del tema y apariencia de la aplicación.

5.3. Módulos desarrollados

El desarrollo se centró en dos módulos principales:

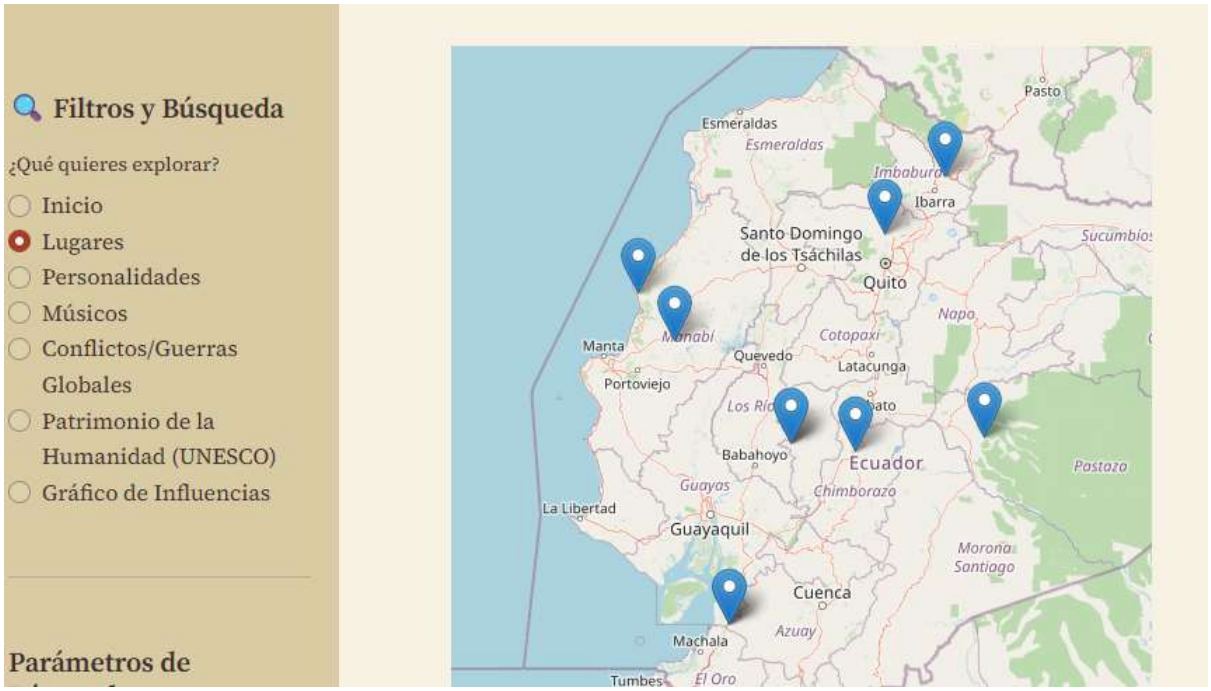
- **app.py (Módulo Frontend y Lógica de Aplicación):**
    - Este script es el punto de entrada de la aplicación Streamlit.

- Gestiona la interfaz de usuario: crea la barra lateral, los menús de navegación, los campos de entrada de texto y el área de visualización de contenido.
- Contiene la lógica para procesar los datos recibidos de sparql_queries.py, transformándolos en DataFrames de Pandas.
- Es responsable de orquestar las visualizaciones utilizando Folium para mapas y Plotly para gráficos y líneas de tiempo.
- Incluye la lógica de selección de categorías y el manejo de los términos de búsqueda ingresados por el usuario.

- **sparql_queries.py (Módulo de Servicios de Datos):**
  - Este módulo encapsula toda la lógica de interacción con los *endpoints* SPARQL de DBpedia y Wikidata.
  - Define las constantes DBPEDIA_ENDPOINT y WIKIDATA_ENDPOINT.
  - Implementa la función run_sparql_query que es el método genérico para enviar consultas SPARQL vía HTTP (requests), manejar las respuestas JSON y gestionar los errores de conexión. Esta función incorpora la estrategia de caché de Streamlit (@st.cache_data) para optimizar el rendimiento.
  - Contiene funciones especializadas para cada tipo de consulta cultural (ej., get_monuments_or_places_in_ecuador, get_ecuadorian_personalities, get_global_wars_and_conflicts, etc.), que construyen las cadenas de consulta SPARQL específicas y llaman a run_sparql_query.

5.3. Pruebas realizadas

La página de inicio con la sección "Sabías que...".



La sección "Lugares de Ecuador" con el mapa interactivo y algunos marcadores.

La sección "Personalidades Ecuatorianas" mostrando una lista de figuras y el detalle de una al hacer clic.

La "Línea de Tiempo de Guerras y Conflictos Globales" mostrando la visualización interactiva.

La sección "Patrimonio de la Humanidad (UNESCO)" con la galería de sitios.

El "Gráfico de Influencias" mostrando las relaciones entre personalidades.)

- **Pruebas Funcionales:** Se realizaron pruebas exhaustivas de cada una de las funcionalidades de la aplicación (exploración de lugares, personalidades, músicos, conflictos, UNESCO, influencias). Se validó que los filtros de búsqueda funcionaran correctamente, que los datos se mostraran de manera precisa y que las visualizaciones interactivas respondieran según lo esperado.
- **Pruebas de Integración:** Se verificó la correcta comunicación e interacción entre el módulo app.py y sparql_queries.py, asegurando que los datos fueran solicitados, recibidos y procesados sin inconsistencias.

- **Pruebas de Rendimiento (Básicas):** Se evaluó el impacto del mecanismo de caché (@st.cache_data) en los tiempos de respuesta. Se observó una mejora significativa en la velocidad de carga para consultas repetidas dentro del periodo de caché, confirmando la eficacia de la optimización.

5.4. Problemas encontrados y soluciones aplicadas

Durante la fase de implementación, se identificaron y resolvieron varios desafíos:

- **Inconsistencia en el Formato de Fechas de LOD:** Las fechas recuperadas de DBpedia y Wikidata a menudo presentaban formatos variados (ej. YYYY, YYYY-MM-DD, YYYY-MM-DDTHH:MM:SSZ).
    - **Solución:** Se implementó lógica de análisis y formateo de fechas en app.py utilizando el módulo datetime, con bloques try-except para intentar múltiples formatos y asignar valores por defecto (ej. 1 de enero si solo se tiene el año) cuando la información era incompleta, garantizando la correcta visualización en líneas de tiempo.
- **Ausencia de Datos para Propiedades Opcionales:** Algunas propiedades (como dbo:thumbnail o schema:description) no siempre están presentes para todas las entidades en los LOD.
    - **Solución:** Se utilizó el método .get() con un segundo argumento para proporcionar valores por defecto (ej., item.get('thumbnail', {}).get('value', None)) o condicionales if para mostrar mensajes como "No hay descripción disponible" en lugar de un error, lo que mejora la robustez de la aplicación.
- **Límites de Resultados de los Endpoints SPARQL:** Los *endpoints* públicos de DBpedia y Wikidata imponen límites en el número de resultados por consulta para evitar sobrecargas.
    - **Solución:** Se introdujo el parámetro limit en todas las funciones de consulta de sparql_queries.py, permitiendo controlar la cantidad de resultados y asegurar que las consultas se mantuvieran dentro de los límites aceptables por los servidores de los LOD.

- **Optimización del Rendimiento en Consultas Remotas:** Las consultas a *endpoints* externos pueden ser lentas y generar latencia.
    - **Solución:** La implementación clave fue el uso del decorador @st.cache_data(ttl=3600) en la función run_sparql_query. Esto minimiza el número de peticiones reales a los *endpoints* remotos, ya que los resultados de las consultas se almacenan en la memoria durante una hora, acelerando drásticamente las cargas subsecuentes de la misma información.

Capítulo 6: Resultados y Evaluación

6.1. Comparación con requisitos iniciales

CulturaViva ha logrado cumplir satisfactoriamente con los requisitos funcionales y no funcionales definidos inicialmente. La aplicación permite la exploración interactiva de diversas categorías de patrimonio cultural, utilizando efectivamente los Linked Open Data. La interfaz es intuitiva y las visualizaciones son interactivas, facilitando la comprensión de información compleja. Los objetivos específicos de diseño de interfaz, desarrollo de servicios SPARQL y creación de visualizaciones dinámicas han sido alcanzados, resultando en un producto funcional que demuestra el potencial de la Web Semántica para la difusión cultural.

6.2. Evaluación del desempeño

El desempeño de la aplicación es eficiente y responsivo para su propósito. La estrategia de caché de datos (@st.cache_data) ha sido crucial, reduciendo drásticamente los tiempos de carga para consultas repetidas (después de la primera petición). Las visualizaciones con Folium y Plotly se renderizan de manera fluida, proporcionando una experiencia interactiva sin demoras significativas. La carga inicial de algunas categorías puede tomar unos segundos debido a la latencia inherente de las consultas remotas a los *endpoints* SPARQL, pero esta es una limitación esperada del uso de fuentes de datos externas y es mitigada por el caché.

6.3. Beneficios del sistema desarrollado

Acceso Democratizado al Conocimiento: Facilita la exploración del patrimonio cultural de manera sencilla para un público amplio, desde estudiantes hasta turistas y académicos.

Fomento de la Identidad Local y Nacional: Permite a los ciudadanos conectarse de manera más profunda con su historia, geografía y figuras influyentes, fortaleciendo el sentido de pertenencia y valoración de su legado.

Revalorización del Patrimonio Olvidado: Al hacer visible información que antes estaba dispersa, CulturaViva contribuye a reavivar el interés por elementos culturales que podrían pasar desapercibidos.

Demostración Práctica del Potencial de LOD: Sirve como un caso de estudio convincente sobre cómo los Linked Open Data pueden ser aplicados en soluciones prácticas para la difusión del conocimiento y la creación de valor.

Experiencia Interactiva y Educativa: A través de sus visualizaciones dinámicas (mapas, líneas de tiempo, gráficos de red), la aplicación transforma la información estática en una experiencia de aprendizaje inmersiva.

Reutilización de Datos Públicos: Promueve la cultura de datos abiertos al consumir y presentar información libremente disponible, contribuyendo a un ecosistema de conocimiento compartido.

6.4. Limitaciones del producto final

Dependencia de Fuentes Externas: La funcionalidad y la completitud de la información están directamente ligadas a la disponibilidad, el esquema y la calidad de los datos en DBpedia y Wikidata, lo que está fuera del control del proyecto.

Cobertura de Datos: Aunque las fuentes LOD son vastas, la información sobre ciertos aspectos del patrimonio (especialmente local o muy específico) puede ser limitada o inexistente.

No Gestión de Usuarios: Carece de módulos de autenticación, perfiles de usuario o personalización basada en preferencias individuales.

Capítulo 7: Conclusiones y Recomendaciones

7.1. Conclusiones generales del proyecto

CulturaViva se erige como una prueba exitosa y tangible del concepto que valida el inmenso potencial de los Linked Open Data para transformar la forma en que las comunidades interactúan con su patrimonio cultural. La integración estratégica de Python, el framework Streamlit y poderosas librerías de visualización (Folium, Plotly), junto con el consumo eficiente de *endpoints* SPARQL de DBpedia y Wikidata, ha dado como resultado una aplicación web robusta, intuitiva y estéticamente atractiva. Se ha logrado el objetivo principal de democratizar el acceso al conocimiento cultural, fortalecer la identidad y fomentar la apreciación del patrimonio, demostrando que la Web Semántica puede traducirse en soluciones prácticas y de alto impacto social.

7.2. Recomendaciones para trabajos futuros

Diversificación y Enriquecimiento de Fuentes LOD: Explorar la integración con otros *endpoints* SPARQL o bases de datos culturales abiertas y locales que puedan complementar y enriquecer el conjunto de datos actual, especialmente para el patrimonio regional o especializado no completamente cubierto por DBpedia o Wikidata.

Mejora de la Interacción en Gráficos de Red: Investigar e implementar algoritmos de diseño de grafos más avanzados (ej., force-directed layouts más optimizados) y funcionalidades de filtrado/selección interactiva para el "Gráfico de Influencias", permitiendo una exploración más profunda de las conexiones.

Análisis Semántico Avanzado: Incorporar técnicas de procesamiento de lenguaje natural (NLP) para realizar análisis semánticos sobre los *abstracts* o descripciones de las entidades, lo que podría generar nubes de etiquetas, resúmenes automáticos o identificar temas relacionados de forma más inteligente.

7.3. Posibilidades de mejora o ampliación

Funcionalidad de Contribución Comunitaria: Investigar y diseñar un módulo que permita a los usuarios sugerir nuevas entidades, correcciones de datos o añadir información local (ej., fotografías, eventos comunitarios), con un proceso de curación para mantener la calidad de los datos.

Sistemas de Recomendación: Desarrollar un motor de recomendación que sugiera contenido cultural relacionado basándose en las exploraciones previas del usuario o en la similitud semántica entre entidades.

Integración Multimedia Extensa: Conectar con APIs de audio o video (ej., YouTube, SoundCloud, archivos históricos de medios) para integrar contenido multimedia directamente relacionado con músicos, eventos o lugares, enriqueciendo la experiencia.

Despliegue a Gran Escala y Monitoreo: Para un uso masivo, explorar opciones de despliegue en la nube más robustas que Streamlit Cloud (ej., AWS, GCP, Azure) que permitan mayor escalabilidad, balanceo de carga y herramientas de monitoreo de rendimiento.

Internacionalización: Adaptar la aplicación para soportar múltiples idiomas en la interfaz y en las consultas SPARQL (utilizando filtros de idioma adecuados), ampliando su alcance.

Referencias bibliográficas

Holze, J. (2022, 9 agosto). *Home - DBPedia Association*. DBpedia Association. https://www.dbpedia.org/

Folium documentation. (s. f.). https://folium.readthedocs.io/en/latest/

pandas documentation — pandas 2.3.1 documentation. (s. f.). https://pandas.pydata.org/docs/

Plotly. (s. f.-b). https://plotly.com/python/

Python API reference for plotly — 6.2.0 documentation. (s. f.). https://plotly.github.io/plotly.py-docs/

Streamlit Docs. (s. f.). https://docs.streamlit.io/

Wikidata. (s. f.). https://www.wikidata.org/wiki/Wikidata:Main_Page

https://query.wikidata.org/

Anexos

Enlaces a repositorio de Git que contiene el código fuente completo de app.py,sparql_queries.py y la documentación necesaria.

Enlace: https://github.com/drbermeo/OpenPageCulturaViva.git