Ejercicio de Arquitectura Software
Curso 2022-2023

Versión 1.1

Tabla de contenidos

1. Objetivos

El principal objetivo de la práctica es la aplicación de las diferentes actividades estudiadas en la asignatura para ensayarlas en una situación práctica.

A tal fin los grupos de estudiantes desarrollarán una arquitectura de software para un sistema y la implementación de una parte del sistema usando una arquitectura Pipeline y realizaran a continuación el diseño de la arquitectura del sistema completo.

El caso de estudio se lleva a cabo trabajando en equipo. El lenguaje de programación, la base de datos y el sistema operativo serán elegidos libremente por los participantes. El código debería poderse ejecutar en un ordenador portátil y el grupo deberá ser capaz de demostrar el funcionamiento si el profesor se lo requiere. Para el desarrollo de la arquitectura se usaran las herramientas de modelado que se estimen más adecuadas. En la siguiente URL se pueden encontrar algunas del as herramientas disponibles (https://www.workingsoftware.dev/documentation-as-code-tools/)Debido al corto plazo de la corrección, el código debe tener los siguientes atributos de calidad: **Legible, reusable, mantenible y fácilmente instalable.**

Para el apartado de documentación de la arquitectura del sistema completo se requerirá el uso de plantillas de documentación (arc42.org, SEI template https://learning.oreilly.com/library/view/documenting-software-architectures/9780132488617/, c4model en c4model.com) o estándares de documentación (p. Ej: IEEE/ISO/IEC 42010 https://ieeexplore.ieee.org/document/9938446). La documentación de la arquitectura como un conjunto salvaje de diagramas se penalizará negativamente en la calificación.

El ejercicio se divide en tres tareas separadas, donde la primera tarea debe completarse usando comandos de linea de sistema operativos (Linux, Unix, Windows), la segunda tarea se debe diseñar la arquitectura del caso de estudio siguiendo los pasos del método ADD. Y la tercera es el correspondiente análisis retrospectivo del esfuerzo realizado

2. Descripción del Caso de Estudio

Descripción de la aplicación a diseñar

La meta del ejercicio será implementar una aplicación basada en base de datos, aplicando arquitecturas. La aplicación ha sido encargada por un organismo oficial de la Comunidad de Madrid. La aplicación está pensada para que sirva de apoyo al turismo en el contexto de las ciudades inteligentes.

Los datos de la aplicación se toman de la fuente de datos abiertos OpenStreetMap, y se deberá filtrar los nodos correspondientes para los teatros, cines, pubs, discos y demás salas de fiesta que se consideren. Además se requiere disponer de información de las las paradas del servicio de metro de la Comunidad de Madrid.

El usuario accediendo a la aplicación debería de ser capaz de encontrar a cualquiera de los elementos filtrados, bien indicando unas coordenadas que definirán un área de búsqueda (un cuadrado delimitador) y/o por una categoría especifica (e.j. “teatro”). El punto central del cuadrado corresponderá a las coordenadas dadas por el usuario y el tamaño del cuadrado debería ser ajustable por el usuario (por ejemplo, 100x100m).

Además el sistema debe permitir al usuario la posibilidad de seleccionar y guardar los resultados de las búsquedas como búsquedas favoritas.

Las fuentes de datos a manejar son:

- Mapa OpenStreetMap de España (disponible en http://download.geofabrik.de/europe.html)
- Fichero de Puntos de Interés de Madrid: POI_Madrid.csv (disponible en Moodle de la asignatura)
- Limites de Madrid: ComunidadMadrid.poly, si fuera necesario se puede trabajar solo dentro de la ciudad de Madrid usar el fichero CiudadMadrid.poly/ambos ficheros disponibles en Moodle de la asignatura).

Las tareas a realizar son:

1. Realización y ejecución de un producto con arquitectura Pipeline (estilo de comandos de Unix)
2. Realización del diseño arquitectónico
3. Presentación del esfuerzo realizado

A. Tarea 1 Pipeline -Linea de comandos de Unix (30 puntos)

Escribir primero una tubería de comandos Unix (o similar) para importar un dataset a una base de datos, usando el concepto de tubería y herramientas de la linea de comandos. Al final de la ejecución la base de datos debe estar completamente poblada. Los datos deben ser procesados, convertidos a CSV, filtrados y sin almacenarse en archivos, pasar a una base de datos SQL.

Existen herramientas disponibles para tratar con los datos del OpenStreetMap. Entre las herramientas disponibles se encuentran osmfilter, osmconvert, osmosis.

- Descargar OpenStreetMap-datos para España: (spain-latest.osm.pbf) Escribir una pipeline de comandos de Unix (no se debe crear un script) que resuelve las tareas siguientes:
    - Extraer TODOS los nodos de tipo cines, teatros, salas de fiesta, así como las estaciones de metro que haya en el mapa
    - Filtrar aquellos que quedan justo dentro de Madrid. Escoger, o bien el termino municipal de Madrid (se reduce más el tamaño de los datos), o bien la comunidad autónoma (se amplia el volumen de datos). Para ello, puede utilizar este archivo .poly: (CiudadMadrid disponible en Moodle)

- Convertir el mapa a texto con un formato datos separado por comas (CSV, se puede usar otro separador si fuera necesario). Se puede usar osmconvert para convertir. Se debe guardar al menos el identificador, la longitud, la latitud, nombre y el horario de apertura. (Ese es el conjunto mínimo pero sois libres si queréis añadir mas metadatos).
- Usando **solamente** herramientas estándar del sistema operativo (awk, sed, grep, ...) filtrar las entradas que no tienen un nombre (es decir, el nombre está vacío), si los hay.
- Cargar el contenido a una base de datos donde se pueda consultar con SQL estándar (por ejemplo: sqlite3, Postgres)

L**a creación de la base de datos y de las tablas también se requiere que se genere en la "pipeline"**. Además, el pipeline no debería fallar en el caso de que las bases de datos o las tabla(s) ya existieran. Tu puedes usar un script para crear la infraestructura de base de datos o importar los datos, pero el script necesita ser ejecutado directamente desde el pipeline (no se admite comando &&).

Utilizando el mismo conjunto de datos que para la primera tubería, escribir una segunda tubería de comandos de Unix que resuelve las tareas siguientes:

- Utilice un filtro con el fin de mantener los teatros que aborden el genero musical.
- Se puede ensayar usando OpenStreetMap.org para visualizar los resultados.

- Al igual que en la tubería anterior, mirar a ver si en este caso puedes de guardar el teléfono y la dirección web de los nodos, siempre que la tengan.
    - Filtrar los nodos que quedan dentro de Madrid. Para ello, puede utilizar este archivo .poly: (Madrid.poly)
    - Convertir también al formato CSV. Tienes que guardar al menos el identificador, longitud, latitud y el nombre de las entradas, pero eres libre de utilizar metadatos adicionales que desees guardar, por ejemplo horas de apertura.
    - Utilizando sólo programas de terminal Unix estándar (como awk, sed, grep, ...) con el fin de filtrar los nodos que no tienen un nombre (es decir, el nombre está vacío), si los hay.
- Cargar el contenido a una base de datos que comprende consultas SQL estándar, por ejemplo sqlite3, Postgres,….

La creación de la base de datos y tabla (s) es necesario que ocurra en una de las dos tuberías (lo normal es en la primera). Por otra parte, las tuberías no deben fallar si la base de datos o las tablas ya existen. Se puede usar un script para crear la infraestructura necesaria para la base de datos o cargar los datos, pero el script tiene que ser ejecutado desde su tubería.

Se puede especificar el orden en que las dos tuberías deben ser ejecutadas (es decir, la que se ejecuta en primer lugar crea la base de datos). La decisión sobre el comportamiento en relación con la sobreescritura o de agregar las entradas ya existentes depende de lo que se decida en el grupo. Se puede bien sobrescribir o bien anexar.

Nota: Se pueden utilizar archivos permanentes, pero debe haber al menos dos operadores '|' de encadenamiento o bien operadores de redireccionamiento ('>', '<'). dentro de cada tubería. No pueden aparecer comandos estilo '&&' en la tubería.

Ejemplo de tubería: cat /proc/cpuinfo | grep -A 5 -B 5MHz | grep 'core id' | awk NR == 1

B. Tarea 2 Realización diseño arquitectónico y su documentación (50 puntos)

Realizar el diseño arquitectónico de una aplicación siguiendo el método ADD. Es importante documentar la arquitectura usando plantillas de documentación de arquitectura como arc42 (arc42.org/download), o c4model (c4model.com), SEI template (https://learning.oreilly.com/library/view/documenting-software-architectures/9780132488617/,) o estándares de documentación (IEEE/ISO/IEC 42010 https://ieeexplore.ieee.org/document/9938446). Las decisiones sobre la arquitectura tomadas deben registrarse siguiendo plantillas (disponibles en: https://github.com/joelparkerhenderson/architecture-decision-record)

El diseño generado debería representar la arquitectura de una aplicación web donde se pudieran presentar, filtrar selectivamente y editar los datos almacenados de los locales de diversión y estaciones del metro disponibles en Madrid generados por la pipeline de la tarea 1. Se debe asumir que los usuarios finales no son expertos en tecnología. El sistema debería soportar 500 usuarios concurrentes. El sistema debe ser capaz de recibir un gran número de consultas cada segundo y se nos ha solicitado que debe ser capaz de procesarlas dando tiempos de respuesta, en al menos el 80% de los casos, inferiores a dos segundos. El sistema debe estar disponible 24 horas al día, los siete días de la semana.

Un conjunto mínimo de las funcionalidades de la aplicación serian:

- Listar y filtrar los establecimientos por categoría, nombre, alrededor de un punto de interés (POI) seleccionado por el usuario (por ejemplo, 100 m), (10 puntos)
- Salvar una búsqueda como una búsqueda favorita con un nombre descriptivo. Por ejemplo, si buscáramos los teatros que ofrecen musicales cerca de la plaza del Callao, un posible nombre descriptivo seria "Teatros del genero Musical cerca de la plaza del Callao" (10 puntos)
- Editar/borrar una búsqueda favorita (10 puntos)
- Añadir/editar/borrar cines, teatros, salas de fiesta, discos(10 puntos).

Este es el conjunto mínimo de funcionalidades que nos ha marcado el cliente. La evolución de la aplicación permite suponer que en un futuro se incluirán funciones adicionales.

Una vez realizado el diseño, el siguiente paso es documentar la arquitectura de manera profesional. En términos generales, la documentación de la arquitectura debe describir las "decisiones de diseño mas importantes" del proyecto, y "por qué se tomaron".

Básicamente debe responder a los siguientes aspectos:

1. Introducción y metas
    1. Proporcionar el propósito y el alcance del sistema.
    2. Además, se presentarán las principales metas y restricciones de la arquitectura. Se trata tanto de incluir la funcionalidad como también incluir aspectos cómo los atributos de

calidad mas importantes que pudiera tener el sistema. Así como las posibles restricciones técnicas y de gestión que pudiera afectar al desarrollo del sistema.

2. Estrategia de la solución. Una explicación de las decisiones fundamental para la solución que dan forma a la arquitectura del sistema. Esta es la parte más importante del informe. Entre las decisiones se encuentran decisiones acerca de la descomposición a alto nivel de un sistema, por ejemplo, el uso de una arquitectura de referencia o patron de diseño. Decisiones sobre como alcanzar las metas de calidad mas prioritarias. Se recomienda seguir alguna de las plantillas presente en (https://adr.github.io/). Por ejemplo se puede usar la plantilla de Michale Nygard para registrar la decisión sobre las unidades físicas de proceso se vayan a incorporar en la vista de despliegue.
3. Vista de bloques, representando la perspectiva lógica del sistema. el nivel de detalle requerido llegará al equivalente al nivel 3 de Componentes del modelo c4model.com. Se deben instancia todos los elementos de la arquitectura. Todos los elementos de la arquitectura deben tener asignada una responsabilidad y haberse definido las interfaces. Hay que tener cuidado para identificar correctamente los diferentes tipos de elementos en los diagramas usando diferentes notaciones (forma, color, etc.) y no hay que olvidarse de incluir una leyenda en el diagrama que explique el significado de las notaciones.
4. Vista física de despliegue. La vista física describe la infraestructura técnica usada para ejecutar el sistema. Los elementos software descritos en la vista de bloques deben de estar rapeados a los elementos de infraestructura.

Cada elemento y cada interfaz presente en la arquitectura deberá asegurarse que está definido. Se valorará negativamente la falta de definición tanto de los componentes como de las interfaces.

Recordar que el objetivo de la documentación es mantener el registro de las diferentes vistas de la arquitectura junto con todas las decisiones de diseño adoptadas en su elaboración.

Documentar una arquitectura como un conjunto salvaje de diagramas demuestra poca profesionalidad, y se valorará muy negativamente. Es importante, acogerse a un estándar y ajustar el estándar de documentación para proporcionar un aspecto profesional a la documentación. Recordar que se disponen de plantillas de documentación tales como arc42.org, SEI template https://learning.oreilly.com/library/view/documenting-software-architectures/9780132488617/, c4model en c4model.com o estándares de documentación (p. Ej: IEEE/ISO/IEC 42010 https://ieeexplore.ieee.org/document/9938446). Evidentemente habrá que ajustar el contenido al producto realizados. Evaluar si fuera posible reducir el número de vistas en función de las características de la aplicación.

C. Tarea 3 Presentación del esfuerzo dedicado (20 puntos)

Esta tarea tendrá el siguiente resultado:

A. Un fichero de presentación donde explique el equipo de manera sintética el esfuerzo que ha dedicado a la tareas y los resultados de cada tarea. El equipo debería realizar hincapié en lo conseguido y también en las dificultades surgidas.

Para facilitar la corrección será necesario tanto el fichero de presentación se generen o bien en formato bien .PPTX (Microsoft PowerPoint) o bien .Key (Apple Keynote)

3. Entrega

La entrega se realizará a través del sistema Moodle.

Debe enviarse un fichero comprimido (.zip o similar) a la plataforma Moodle. La estructura del fichero comprimido se ajustará a una carpeta por tarea de manera clara (por ejemplo, Tarea 1, Tarea 2,… o Tarea01, Tarea02, o Task1, Task2,…).

El equipo podrá elegir el conjunto de herramientas que vaya a utilizar para describir las vistas. Sin embargo, independientemente de la herramienta utilizada, el formato de los documentos que se presente deberá ser txt, word, PDF o .pptx o key indicados anteriormente.

La práctica se deberá entregar en la fecha prevista en la plataforma Moodle. Se recuerda que previamente a la entrega, los equipos puede consultar por email o via Teams aquellos aspectos que le planteen dudas al profesor de la asignatura en tomas.sanfeliu@upm.es.