Trevenque. Proyecto de formación dual.

Una empresa ha contratado nuestros servicios para implementar un sistema informático que gestione todos sus procesos internos. Esta empresa gestiona y diseña sistemas de información para la creación de herramientas meteorológicas, dentro de los servicios contratados están los asociados a administración de sus sistemas informáticos y los asociaciones al diseño de software .

- Diseño del software cliente y servidor.

1. Necesitan crear una herramienta software que permita ver de manera rápida y eficiente el tiempo asociado a cada uno de los puntos de control especificados y crear un sistema óptimo para mostrar y visualizar toda la información (usando un mapa interactivo). Para la construcción de esa herramienta será necesario crear un cliente con una capa Javascript para la visualización de datos (Tablas, Gráficas y Videos).
2. Crear un sistema que apoye a ese cliente usando una RestAPI global que a su vez se alimente de la información extraída del tiempo.net (https://www.el-tiempo.net/api) y de Aemet (https://www.aemet.es/es/datos_abiertos/AEMET_OpenData). El sistema estará construido sobre PHP-Laravel apoyado por una base de datos MariaDB.

   El software del lado del servidor de preparar todas las rutas para:

   - Consulta de tiempo para una localidad.
   - Consulta de tiempo para una provincia.
   - Consulta de tiempo general para una autonomía.
   - Datos estadísticos globales.
   - Asociar por parte del cliente una localidad a un punto del mapa usando leaflet o openlayers (https://leafletjs.com/) o (https://openlayers.org/), esos puntos de control serán almacenados en la base de datos.

- Diseño del sistema por parte del equipo de administración de sistemas.

1. Preparación de los servidores necesarios para soportar MariaDB y Laravel. El sistema deberá estar diseñado de manera independiente, es decir en un principio debe de haber un servidor independiente para cada uno de ellos.
2. Añadir un sistema de balanceo de carga al servidor MariaDB usando Galera o MaxScale, por supuesto hacerlo de manera totalmente independiente de Laravel.
3. Preparar un sistema de balanceo de carga usando NGINX para los servidores Laravel.
4. Dockerizar toda la configuración una vez que los sistemas estén funcionando.
5. Instalar un configurar un servidor GIT para el control del proyecto, preparar un día para informar a los desarrolladores sobre el uso de esta herramienta.
6. Preparar todos los scripts necesarios (python, bash, node) para el control de las configuraciones.

- Avanzado.

Nos proponen por parte de la empresa que cierta información estaría muy bien que estuviera disponible a través de video, usando un presentador virtual para describir la información captada por alguna de las APIs. Por tanto sería necesario crear videos, almacenarlos en un recurso compartido e indexarlos para ser accesibles desde el servidor Laravel. Nuestro equipo de investigación ha propuesto este proyecto: https://github.com/TMElyralab/MuseTalk, aunque también se aceptarán otras propuestas.

Rúbrica para el Proyecto: Sistema Informático para la Gestión Meteorológica

El proyecto tiene dos partes principales: **Desarrollo de Software** y **Administración de Servidores**. La siguiente rúbrica está dividida en dos secciones para evaluar ambos aspectos de manera detallada.

---

trevenque group

Parte 1: Desarrollo de Software

Criterio	Excelente (5)	Bueno (4)	Satisfactorio (3)	Insuficiente (1-2)
Desarrollo de la interfaz cliente (JavaScript)	Interfaz altamente interactiva, fácil de usar, con gráficos, tablas y videos perfectamente integrados. Implementación fluida con Leaflet/OpenLayers.	Interfaz funcional con tablas, gráficos y videos, pero podría mejorar en cuanto a la interacción y usabilidad.	La interfaz funciona, pero la interacción es limitada y carece de algunos elementos importantes, como gráficos o videos.	La interfaz no cumple con las expectativas o tiene problemas graves de usabilidad o funcionalidad.
Integración con APIs externas (El tiempo.net y Aemet)	Integración eficiente y sin errores con ambas APIs, manejo adecuado de errores y excepciones, datos actualizados correctamente.	Integración funcional con las APIs externas, pero podría optimizarse o tener más controles para errores.	Integración básica que podría fallar en algunas circunstancias o no actualizarse con regularidad.	No hay integración o la integración presenta graves fallos o datos incorrectos.
Desarrollo de la API Restful (PHP-Laravel)	API bien estructurada, completamente funcional con rutas bien definidas y documentadas, respuestas rápidas y eficientes.	API funcional con rutas básicas definidas, aunque faltan detalles de optimización o documentación.	La API está implementada pero con falta de rutas importantes o con fallos en la eficiencia y manejo de peticiones.	La API no funciona correctamente o tiene muchas rutas faltantes o mal implementadas.
Base de datos (MariaDB)	Base de datos bien estructurada con relaciones claras, optimización de consultas, uso adecuado de índices.	Estructura de base de datos funcional, aunque algunas relaciones o consultas podrían ser optimizadas.	Base de datos con problemas en la estructura, consultas ineficientes o relaciones mal definidas.	La base de datos está mal diseñada o presenta fallos graves en el rendimiento y la integridad de los datos.

Funcionalidad del mapa interactivo (Leaflet/OpenLayers)	Mapa completamente funcional con puntos de control bien asociados a las localidades, navegación fluida.	El mapa funciona bien, aunque puede ser mejorado en términos de interactividad o precisión de los puntos de control.	El mapa interactúa, pero con limitaciones en la colocación de puntos o navegación.	El mapa no interactúa correctamente o no muestra los puntos de control de manera adecuada.
Generación de videos (Avanzado)	Videos generados de manera eficiente con un presentador virtual integrado, accesibles desde el servidor Laravel.	Los videos funcionan bien, pero el proceso de generación o acceso a los mismos podría mejorarse.	Los videos están presentes, pero la generación o visualización tiene fallos importantes.	No hay generación o visualización de videos o la calidad es inaceptable.

Parte 2: Administración de Servidores

Criterio	Excelente (5)	Bueno (4)	Satisfactorio (3)	Insuficiente (1-2)
Preparación de servidores (MariaDB y Laravel)	Servidores configurados de manera óptima e independiente, con instalación adecuada de MariaDB y Laravel.	Servidores configurados correctamente, pero con algunos detalles o configuraciones menores que pueden optimizarse.	Servidores funcionales, pero con varios problemas de configuración o falta de optimización en el rendimiento.	La configuración de los servidores está incompleta o presenta fallos graves que afectan la operatividad del sistema.
Balanceo de carga para MariaDB (Galera/MaxScale)	Implementación exitosa de balanceo de carga con Galera o MaxScale, con alta disponibilidad y escalabilidad.	Implementación de balanceo de carga con algunas optimizaciones necesarias, pero funcional en su mayoría.	El balanceo de carga está implementado, pero con algunos fallos que pueden afectar la disponibilidad o la escalabilidad.	El balanceo de carga está mal implementado o no se ha implementado.

Balanceo de carga para Laravel (NGINX)	Implementación robusta de NGINX para balanceo de carga, con alta disponibilidad y optimización del tráfico.	Implementación funcional de NGINX para balanceo de carga, aunque puede necesitar más optimización.	El balanceo de carga con NGINX funciona parcialmente, pero presenta limitaciones en cuanto a rendimiento o configuración.	No se ha implementado el balanceo de carga o está mal configurado.
Dockerización de la configuración	Configuración completamente dockerizada, todos los servicios funcionan dentro de contenedores con redes, volúmenes y variables adecuadas.	Dockerización implementada correctamente, pero con pequeñas optimizaciones necesarias en redes o volúmenes.	Dockerización básica, con algunas configuraciones faltantes o no completamente optimizadas.	La configuración no está dockerizada o tiene fallos graves de implementación.
Instalación y configuración del servidor GIT	GIT configurado correctamente, repositorio funcional, con accesos configurados y documentación para los desarrolladores.	GIT instalado y funcional, pero con algunos detalles de configuración o falta de documentación.	El servidor GIT funciona de manera básica, pero con limitaciones en los accesos o la gestión de ramas.	El servidor GIT no está instalado correctamente o no funciona.
Preparación de scripts para control de configuraciones	Scripts completamente funcionales (Python, Bash, Node), automatizando tareas clave, sin errores.	Los scripts automatizan tareas correctamente, pero pueden mejorarse en cuanto a eficiencia o manejo de errores.	Scripts básicos que funcionan, pero con algunas limitaciones en la automatización o los errores no manejados adecuadamente.	Los scripts no están funcionando correctamente o no automatizan tareas importantes.

Generación de videos (Avanzado)	Videos generados de manera eficiente con un presentador virtual integrado, accesibles desde el servidor Laravel.	Los videos funcionan bien, pero el proceso de generación o acceso a los mismos podría mejorarse.	Los videos están presentes, pero la generación o visualización tiene fallos importantes.	No hay generación o visualización de videos o la calidad es inaceptable.

Comentarios Adicionales:

- **Desarrollo de Software**: La calidad del software debe ser evaluada no solo por su funcionalidad, sino también por la experiencia del usuario, la eficiencia del código y la escalabilidad del sistema.
- **Administración de Servidores**: La administración de los servidores debe garantizar un entorno de alta disponibilidad, escalabilidad y facilidad de mantenimiento, lo que incluye el uso adecuado de tecnologías como Docker y el balanceo de carga.

Esta rúbrica proporciona una evaluación detallada de ambas partes del proyecto, asegurando que tanto la parte de desarrollo como la de administración de servidores se aborden con calidad y eficacia.

Grupo Trevenque