**Práctica 7**

*Martín Morales Zufía*

**1.3.- Preparación del contenedor de la asignatura.**

**1. Realiza inventario de versiones de software instalado en contenedor:**

**a. Version S.O., shell, python**

**b. Apache httpd, Tomcat y otro software middleware instalado (no del GUI).**

**2. Realiza un inventario de usuarios configurados en el contenedor y servicios que ejecutan.**

**3. Identifica las ACL configuradas por defecto en las rutas /home, /opt, /etc, /usr. Realiza una comparativa de herramientas para monitorizar el estado del contenedor y eventos importantes, como conexiones, etc.**

**Considerando el pliego:**

[**https://econtra.ugr.es/licitacion/descargaAdjuntoPub?idAdjunto=129456&usg=AOvVaw1lnUWK7cByo9KsI\_WLyYN5&opi=89978449**](https://econtra.ugr.es/licitacion/descargaAdjuntoPub?idAdjunto=129456&usg=AOvVaw1lnUWK7cByo9KsI_WLyYN5&opi=89978449)

**2.-Arquitectura**

**2.1.-Realiza un diagrama de la arquitectura del sistema descrito en el pliego a diferentes niveles (incluyendo los sistemas externos en caso de alto nivel):**

**1. Arquitectura del sistema.**

* Capa de Presentación (Frontend): Interfaz de usuario, manejo de eventos.
* Capa de Aplicación (Backend): Lógica de negocio, procesamiento de datos.
* Capa de Datos (Base de Datos): Almacenamiento persistente de información.

**2. Arquitectura de componentes/paquete/app**

* Componentes de Frontend: Componentes de interfaz de usuario, como botones, formularios, etc.
* Componentes de Backend: Controladores, servicios, modelos.
* Base de Datos: Tablas, vistas, procedimientos almacenados.

**2.2.-Realiza una valoración de escalabilidad de las diferentes arquitecturas monolíticas y distribuidas aplicables en varios escenarios de crecimiento de usuarios, justificando las decisiones en tres situaciones:**

* **Crecimiento un 15%**

Una arquitectura monolítica puede manejar este crecimiento sin problemas. Se puede escalar verticalmente (mejorar el hardware del servidor) para manejar la carga adicional.

Dado que el crecimiento es modesto, no es necesario reestructurar toda la arquitectura.

* **Crecimiento un 100%**

Aunque posible, un crecimiento del 100% puede generar problemas de rendimiento en una arquitectura monolítica. Se puede escalar horizontalmente (añadir más instancias) para manejar la carga adicional, pero puede requerir más esfuerzo.

El aumento significativo en el número de usuarios podría sobrecargar los recursos de la arquitectura monolítica actual.

* **Crecimiento un 600%**

Un crecimiento del 600% es extremadamente alto y probablemente desbordaría una arquitectura monolítica. Se requeriría una revisión significativa de la arquitectura para implementar soluciones de escalabilidad adecuadas, como la migración a una arquitectura distribuida.

La carga adicional generada por un aumento tan drástico en el número de usuarios superaría la capacidad de una arquitectura monolítica para escalar eficientemente.

**3.-Entregables**

**3.1.-Obtén de un proyecto similar al del pliego (puede ser uno de los LMS) una base de código que sirva como entregable del proyecto. Indica cuál es y qué razones te han hecho seleccionarlo. Es muy recomendable clonar el repositorio del proyecto seleccionado en el contenedor.**

**3.2.-Realiza un análisis/revisión del código empleando una o varias herramientas (indica las elegidas y por qué las has considerado), generando un informe. ¿Se han detectado avisos/malas prácticas?**

**3.3.- Ejecuta la compilación del proyecto y genera el/los entregable/s.**

**3.4.- Automatiza el proceso completo, análisis–código + compilación + empaquetado, empleando una de las múltiples herramientas que hay y que pueda ejecutarse por consola (Se recomienda emplear la que use el proyecto elegido). Esta automatización tiene que ejecutarse en el contenedor con las dependencias necesarias y generar evidencias de ello**