Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Análisis y Diseño de Sistemas 2
Sección N y Sección O
Ing. Claudia Liceth Rojas Morales
Ing. Yessenia Janira Marroquín Martínez
Aux. Javier Orlando Jáuregui Juárez
Aux. David Omar Enriquez Reyes

Plataforma de almacenamiento de archivos en la nube “AyDrive”

Proyecto: Fase 1

2do. Semestre 2021

Uso de Docker, DockerFile, DockerCompose, implementación de estilo arquitectónico orientado a servicios y microservicios

OBJETIVOS:

General:

1. Introducir al estudiante en el uso del estilo arquitectónico basada en servicios y microservicios con tecnologías de contenerización como Docker.

Específicos:

- Que el estudiante aprenda a diseñar e implementar estilos arquitectónicos de software.

- Que el estudiante aprenda la estructura del archivo DockerFile para describir contendores y entornos de ejecución.

- Que el estudiante aprenda la estructura de un archivo de configuración de Docker Compose y describa satisfactoriamente un entorno de ejecución multi-contenedor.

- Que el estudiante aprenda a utilizar herramientas de gestión de proyectos e implementación de control de versiones.

Descripción:

La arquitectura basada en servicios (SOA) permite dividir la aplicación en elementos más pequeños e independientes entre sí denominados **servicios**. A diferencia del enfoque tradicional de las aplicaciones, en el que todo se compila en una sola pieza, los servicios son elementos independientes que funcionan en conjunto para llevar a cabo las mismas tareas. Los **microservicios** son un tipo de implementación SOA mucho más **granular**, es decir, dividida en elementos aún más pequeños, lo que permite una separación e independencia aún mayor que la proporcionada por los servicios tradicionales.

La idea detrás de **Docker** es crear contenedores ligeros y portables para las aplicaciones software que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues.

La fase 1 del proyecto consiste en “contenerizar”, crear servicios y microservicios en una aplicación web la cual tendrá como objetivo ser una plataforma web para el almacenamiento de archivos en la nube, más adelante se explicará el funcionamiento.

Para ello utilizarán herramientas para manejo de contenedores tales como **Docker**, **DockerFile** y **Docker-Compose** para lograrlo.

Se deberá de utilizar como repositorio de su código a **Gitlab** y deberán implementar una estrategia de branching basada en **Git-Flow**. Se revisará que las personas hayan codificado o trabajado equitativamente.

Deberán hacer uso de herramientas de **gestión de proyectos** como **Trello** o **Jira** para llevar el control de las tareas por medio de la creación de un tablero Kanban. Las listas básicas con las que deberá contar su tablero serán:

- **Tareas iniciales**
- **Tareas en proceso**
- **Tareas finalizadas**

Además, deben de utilizar herramientas que les permitan implementar los elementos básicos de la metodología **SCRUM** como lo es manejo de **Product Backlog, Sprint Backlog, Sprint Planning, y control de otras métricas que consideren necesarias para la mejora del avance del equipo.** Además, deberán presentar documentación donde se detalle el Sprint Restrospective al finalizar cada Sprint.

Requerimientos de “AyDrive”:

A usted y a su equipo se le ha contratado para la realización de la plataforma web de carga de archivos en la nube **“AyDrive”** para ello se le solicita que implemente un diseño arquitectónico de backend orientado a servicios y microservicios que será consumida por una página web también diseñada por su equipo.

Los requerimientos que se le solicitan son los siguientes:

1. El sistema debe contar con un apartado para registro de usuarios donde se solicite el nickname del usuario (este nickname debe ser único, se debe comprobar que no exista previamente en la base de datos del sistema), correo electrónico (se debe validar que sea único también), fecha de nacimiento, contraseña, confirmación de contraseña.

2. El sistema debe contar con un apartado para iniciar sesión utilizando el nickname y la contraseña. Se debe velar porque el sistema no permita acceder a ninguna página aparte del registro y login si el usuario no ha iniciado sesión.

3. Una vez dentro del sistema, el usuario podrá ver la carpeta principal (root) de su cuenta. En esta carpeta root el usuario podrá agregar nuevas carpetas y archivos.

4. El sistema debe permitir la creación de carpetas. Las carpetas únicamente se crearán en la carpeta (root) y NO habrá subcarpetas aparte. Para la creación de carpetas se debe solicitar el nombre y se debe validar que no exista otra carpeta con el mismo nombre.

5. Para las carpetas se debe poder consultar sus “propiedades” donde se debe listar el nombre, la fecha de creación de la carpeta y la cantidad de archivos que contiene.

6. Dentro de la carpeta root y dentro de las carpetas creadas se podrán subir archivos, para ello se debe implementar un FileUpload que permita seleccionar archivos de la computadora del usuario y cargarlos. En la pantalla donde se carga el archivo también se debe poder ingresar el nombre con el que se desea que se cargue el archivo y debe preservar la extensión con el que fue cargado.

7. El lugar de almacenamiento de los archivos debe ser en la nube y debe permitir su acceso por medio de un link. La forma de almacenar los archivos en la nube queda a discreción de su equipo (Bucket S3, FTP, Sistema de archivos dentro de servidor propio, etc.)

8. Los archivos deben listarse en la plataforma web dentro de la carpeta donde fueron cargadas, se debe validar que no exista un archivo con el mismo nombre y extensión.

9. El usuario debe poder consultar las “propiedades” del archivo, donde se debe listar el nombre, extensión, fecha de subida y link para su acceso.

10. Tanto las carpetas como los archivos deben poder cambiarse de nombre y eliminarse.

11. El usuario podrá cortar (mover) los archivos a otra carpeta.

12. El usuario podrá abrir los archivos los cuáles se descargarán o desplegarán en una nueva ventana/pestaña del navegador y también podrá obtener los links para que cualquiera pueda acceder a estos archivos.

13. El usuario podrá ver su perfil de cuenta y editar su información (nombre, correo, fecha de nacimiento, contraseña). El **nickname** no puede ser modificado.

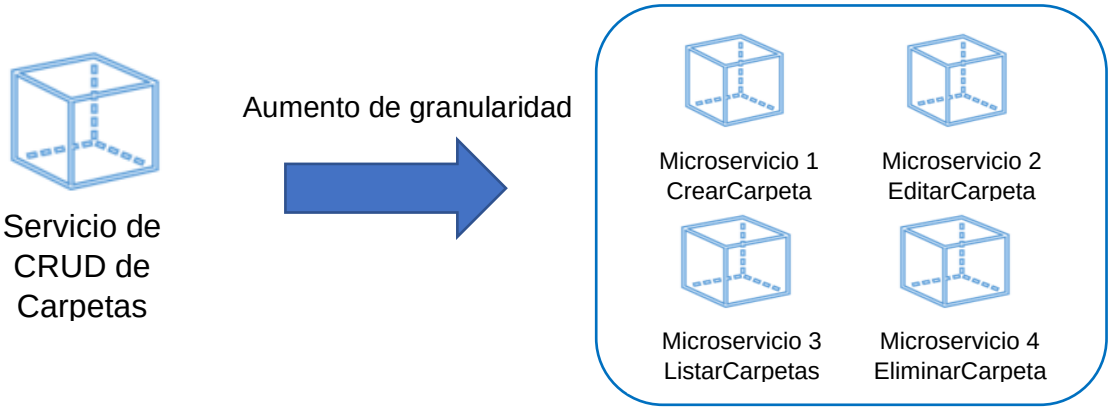
Para la implementación del diseño arquitectónico se deben tomar en cuenta los siguientes requerimientos:

1. El lenguaje y framework para implementar el FrontEnd queda a discreción del equipo pero se debe de proporcionar una interfaz atractiva e intuitiva para el usuario.

2. Los lenguajes para implementar el Backend quedan a discreción del equipo, así como la base de datos y el tipo de esta. También se podrá hacer uso de bases de datos en la nube (Amazon RDS, DynamoDB, Mongo Atlas etc.).

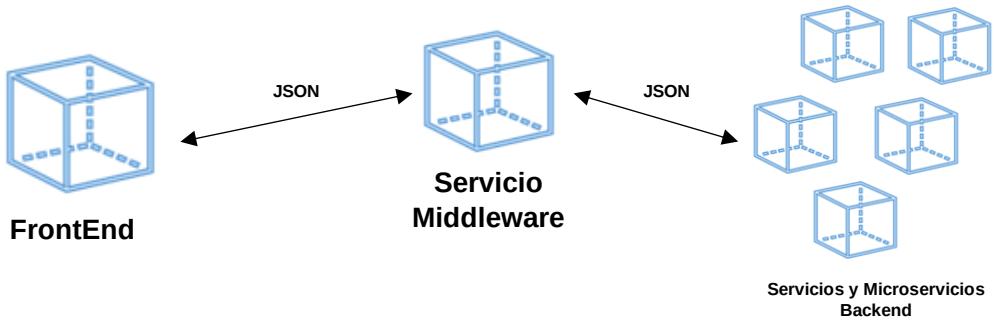
3. Se debe implementar un diseño arquitectónico basada en servicios y microservicios donde se maneje la lógica de backend de cada una de las partes del sistema (Usuarios, Carpetas, Archivos, etc) de forma separada.

4. Para la implementación de microservicios se debe elegir una parte del sistema backend que pueda implementarse como servicio (por ejemplo la funcionalidad para el CRUD de carpetas por medio de un servicio de Carpetas) y dividirlos en al menos **4 microservicios** por medio del aumento de granularidad, donde cada microservicio realice una funcionalidad:



5. Se debe implementar un servicio **middleware** que sirva de intermediario entre las solicitudes del FrontEnd y el Backend, este servicio únicamente debe dejar pasar los JSON provenientes del FrontEnd hacia el Backend y los provenientes del Backend hacia el FrontEnd.

   Toda la comunicación desde FrontEnd hacia Backend y viceversa debe pasar por este servicio:



6. **Tanto el FrontEnd y los servicios deben ser implementados haciendo uso de Docker y Docker-Compose.**

LOGS Y ERRORES:

Se deberá guardar en base de datos los JSON que van desde el FrontEnd hacia el Backend y del Backend hacia el FrontEnd y que pasan por el Middleware. En base de datos se debe crear registros con los siguientes campos por cada comunicación que exista entre FrontEnd y Backend:

Método Middleware	Entrada	Salida	EsError	Fecha y hora

/usuarios	{
	“campo1”: “valor”
	“campo2”: “valor”
	}
		{
		“salida1”: “valor”
		“salida2”: “valor”
		}
			No	01/08/2021 18:50:12

/carpetas	{
	“campo1”:“valor”
	“campo2”:“valor”
	}
		{
		“salida_error”:“valor”
		}
			Sí	01/08/2021 20:30:50

La funcionalidad para guardar los logs se puede colocar dentro del Middleware o como un servicio aparte conectado al Middleware (queda a discreción del equipo).

SEGURIDAD:

- Se le solicita que toda la información sensible de los usuarios como **correo** y **contraseñas** se almacene de forma **encriptada** en **base de datos**. Además, al momento de enviar contraseñas y datos sensibles del usuario desde el FrontEnd a los servicios y viceversa, estos deberán **viajar encriptados** dentro del **JSON** utilizado para la comunicación.

- El método de encriptación queda a discreción de los estudiantes, deben asegurarse de que, si se utiliza una librería, esta funcione tanto en el FrontEnd como en los servicios implementados. Se recomienda el uso de librerías con soporte para varios lenguajes como crypto-js - npm (npmjs.com).

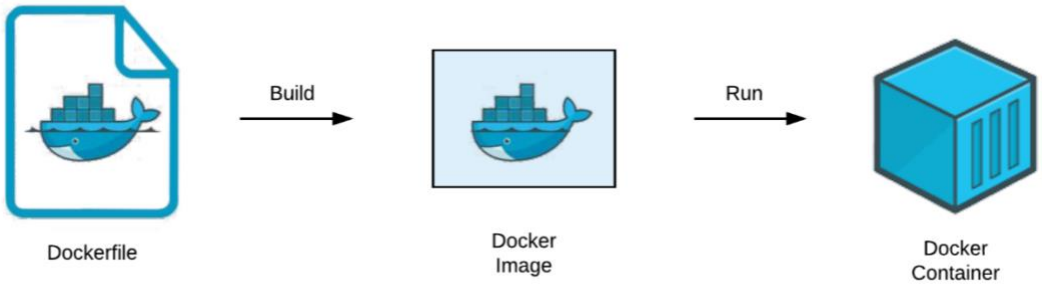
- Se debe validar que el usuario cuenta con una sesión activa antes de acceder a cualquier página que lo requiera, de lo contrario debe redireccionarlo al Login del sistema.

- Se debe validar que únicamente se puedan subir archivos de office, imágenes, videos, pdf y otros documentos de texto. Se debe tener cuidado de no

permitir extensiones peligrosas que permitan ejecutar scripts dentro del servidor como archivos .js, .php, .asp, .aspx, etc.

FLUJO DE TRABAJO DOCKER:

1.Contenerización



La contenerización es el empaquetado que se hace de una aplicación a través de un contenedor para su uso en pruebas (QA) o en producción. EL proceso para contenerización se ilustra en la imagen de arriba.

2.Ejecución del ambiente multi-contenedor

Luego de contenerizar todos los elementos de la aplicación se procederá a utilizar docker compose para ejecutar la aplicación, para ello tendrán que definir un archivo docker -compose.yml como el visto en clase.

DOCUMENTACIÓN DE SPRINTS

Deberán implementar durante todo el desarrollo de la fase al menos **4 Sprints.** Al finalizar cada Sprint deben redactar un documento con los siguientes elementos:

- Tabla comparativa **general** donde se muestre los elementos del Sprint BackLog que se lograron terminar y los que no.

- Sprint Retrospective**: Cada uno de los integrantes** debe responder a las siguientes preguntas: “¿Qué se hizo bien durante el Sprint?” “¿Qué se hizo mal durante el Sprint?” “¿Qué mejoras implementar para el próximo Sprint?”

- Explicación de la metodología utilizada para determinar y priorizar las tareas del Backlog.

DOCUMENTACIÓN

- Documento con los prototipos del FrontEnd (Mockups), se recomienda el uso del programa **Balsamiq**.

- Diagrama del diseño arquitectónico implementado

- Diagramas de casos de uso

- Diagrama de componentes

- Diagrama de despliegue

CONSIDERACIONES

- El lenguaje para utilizar, framework y demás herramientas para crear el proyecto queda a discreción del estudiante, las únicas herramientas obligatorias a utilizar son Docker y DockerCompose.
- Utilizar control de versiones GitLab.
- Copias totales y parciales tendrán una nota de 0 y será reportada a la escuela de sistemas.
- Deben realizar al menos 4 Sprints.
- Deben implementar al menos 4 microservicios.
- Se revisará la organización de su trabajo.
- La estrategia de branching a utilizar debe ser **Git-Flow.**
- Dudas sobre la práctica se responderán en los Foros de sus respectivas secciones.

Entregables

- Link a repositorio con el código fuente, que incluya los archivos DockerFile y Docker-Compose.
- Tanto en el **repositorio**, como en la herramienta para llevar control del trabajo (**Trello o Jira**, por ejemplo) se debe incluir y dar permiso a:

  @3018910950101 (3018910950101@ingenieria.usac.edu.gt **)**
  [Aux. Javier Jáuregui]
  @3003302270101 (3003302270101@ingenieria.usac.edu.gt **)**
  [Aux. David Enriquez]

  Ambos auxiliares deben poder acceder a su repositorio y tablero.

- Colocar documentación solicitada dentro del repositorio.

ENTREGAR A MAS TARDAR EL 31/08/2021 ANTES DE MEDIANOCHE EN UEDI