

TO-DO LIST APP

UNIVERSIDAD DE GUADALAJARA

Objetivo General:

- Diseñar e implementar una aplicación tolerante a fallas utilizando tecnologías modernas.

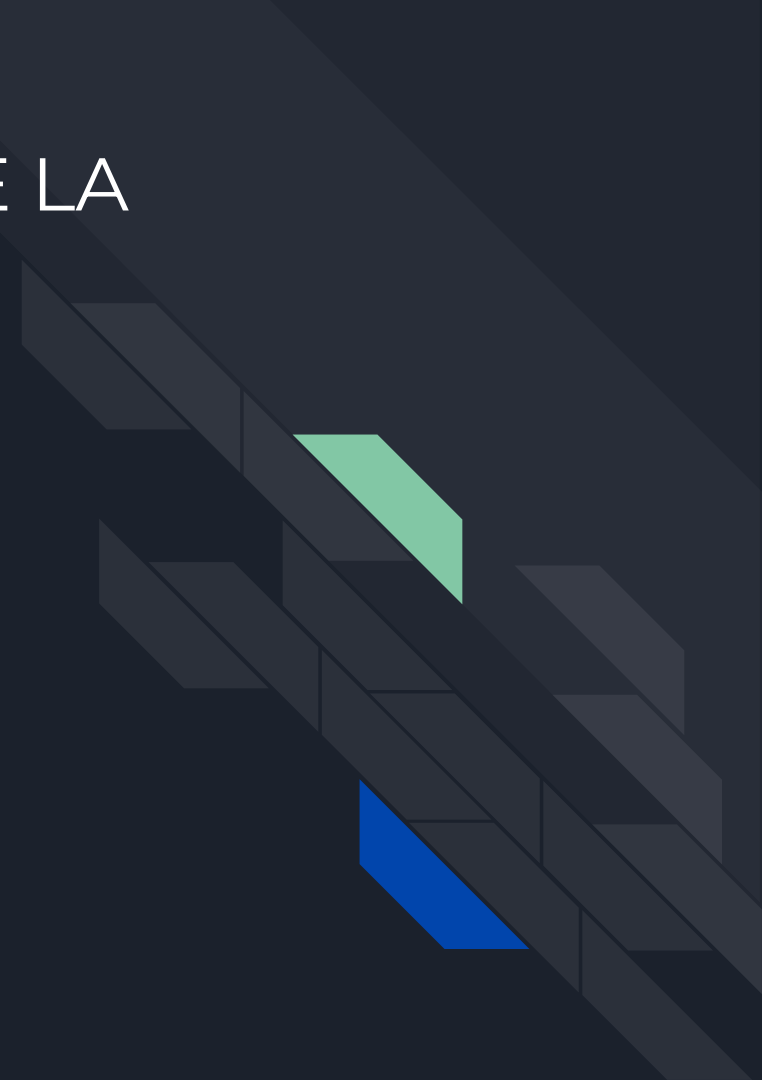
Objetivos Específicos:

- Desarrollar microservicios para la gestión de usuarios y tareas.
- Containerizar los microservicios utilizando Docker.
- Desplegar los microservicios en Kubernetes.
- Implementar prácticas de Chaos Engineering para probar la resiliencia.

ARQUITECTURA DE LA APLICACIÓN

Componentes Principales:

- Microservicio de Usuarios
- Microservicio de Tareas
- Base de Datos MongoDB
- Cliente (Navegador o Postman)



MICROSERVICIOS DE USUARIO

Descripción:

- Desarrollado en Node.js con Express.
- Maneja el registro y autenticación de usuarios.

Características:

- Uso de JWT (JSON Web Tokens) para autenticación.
- Conexión a MongoDB para almacenamiento de datos.

Endpoints Principales:

- POST /register
- POST /login

MICROSERVICIOS DE TAREAS

Descripción:

- Desarrollado en Python con Flask.
- Permite a los usuarios autenticados gestionar sus tareas.

Características:


- Protección de rutas mediante JWT.
- Conexión a MongoDB para almacenamiento de tareas.

Endpoints Principales:

- GET /tasks
- POST /tasks

INICIO DEL PROYECTO

The background features a series of dark gray, three-dimensional rectangular blocks arranged in a stepped, descending pattern from the top right towards the bottom left. A light green parallelogram is positioned on one of the upper blocks, and a blue parallelogram is on a lower block, both oriented diagonally to match the perspective of the blocks.



Comenzamos por ejecutar el siguiente comando en la terminal, desde la raíz del proyecto. De esta manera se orquestan los contenedores existentes para ejecutarse juntos

```
docker-compose up --build |
```

```
=> [user-service internal] load build definition from dockerfile          5.7s
=> => transferring dockerfile: 415B                                     0.2s
=> [todo-service internal] load build definition from dockerfile        5.2s
=> => transferring dockerfile: 505B                                     1.0s
=> [todo-service internal] load metadata for docker.io/library/python:3.9-alpine 4.9s
=> [user-service internal] load metadata for docker.io/library/node:19-alpine 4.6s
=> [todo-service auth] library/python:pull token for registry-1.docker.io 0.0s
=> [user-service auth] library/node:pull token for registry-1.docker.io 0.0s
=> [user-service internal] load .dockerignore                          0.2s
=> => transferring context: 2B                                           0.0s
=> [todo-service internal] load .dockerignore                          0.3s
=> => transferring context: 2B                                           0.0s
=> [user-service 1/5] FROM docker.io/library/node:19-alpine@sha256:8ec543d4795e2e85af924a24f8acb039792ae9fe8a42a 0.5s
=> => resolve docker.io/library/node:19-alpine@sha256:8ec543d4795e2e85af924a24f8acb039792ae9fe8a42ad5b4bf4c277ab 0.4s
=> [user-service internal] load build context                          0.3s
=> => transferring context: 158B                                         0.0s
=> [todo-service 1/5] FROM docker.io/library/python:3.9-alpine@sha256:2ae855d07a137e4e39f9da8995f2fcd938c5bcdce 0.5s
=> => resolve docker.io/library/python:3.9-alpine@sha256:2ae855d07a137e4e39f9da8995f2fcd938c5bcdce466f9a8ac437b 0.4s
=> [todo-service internal] load build context                          0.4s
=> => transferring context: 357B                                         0.0s
=> CACHED [user-service 2/5] WORKDIR /app                             0.0s
=> CACHED [user-service 3/5] COPY package*.json ./                    0.0s
=> CACHED [user-service 4/5] RUN npm install                           0.0s
=> CACHED [user-service 5/5] COPY . .                                  0.0s
=> [user-service] exporting to image                                    3.9s
=> => exporting layers                                                  0.0s
```

Se inicializa el entorno de trabajo y la imagen se construye

```


✓Container zipkin                Created          0.0s
✓Container mongodb               Created          0.0s
✓Container tolerante_a_fallas-main-todo-service-1 Recreat...      14.1s
✓Container tolerante_a_fallas-main-user-service-1 Recreat...      13.5s
Attaching to mongodb, todo-service-1, user-service-1, zipkin
todo-service-1 | * Serving Flask app 'app'
todo-service-1 | * Debug mode: on
mongodb         | {"t":{"$date":"2024-11-11T04:37:40.473+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main", "m
sg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongodb         | {"t":{"$date":"2024-11-11T04:37:40.532+00:00"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"main", "m
sg":"Multi threading initialized"}
todo-service-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
todo-service-1 | * Running on all addresses (0.0.0.0)
todo-service-1 | * Running on http://127.0.0.1:5000
todo-service-1 | * Running on http://172.18.0.4:5000
todo-service-1 | Press CTRL+C to quit
todo-service-1 | * Restarting with stat
mongodb         | {"t":{"$date":"2024-11-11T04:37:40.718+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main", "m
sg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set at least one of the related parameters","attr":
{"relatedParameters":["tcpFastOpenServer","tcpFastOpenClient","tcpFastOpenQueueSize"]}}
mongodb         | {"t":{"$date":"2024-11-11T04:37:40.746+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main", "m
sg":"Initialized wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":25},"
incomingInternalClient":{"minWireVersion":0,"maxWireVersion":25},"outgoing":{"minWireVersion":6,"maxWireVersion":25},"is
InternalClient":true}}}}
mongodb         | {"t":{"$date":"2024-11-11T04:37:40.748+00:00"},"s":"I",  "c":"TENANT_M",  "id":7091600, "ctx":"main", "m
sg":"Starting TenantMigrationAccessBlockerRegistry"}
mongodb         | {"t":{"$date":"2024-11-11T04:37:40.748+00:00"},"s":"I",  "c":"CONTROL",  "id":4615611, "ctx":"initandl
isten", "msg":"MongoDB starting", "attr":{"pid":1, "port":27017, "dbPath":"/data/db", "architecture":"64-bit", "host":"ce75150
176e0"}}

```

Se inician los contenedores necesarios para que se ejecute el proyecto.

FUNCIONAMIENTO DEL PROYECTO





```

mongodb | {"t":{"$date":"2024-11-11T04:39:42.825+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpoint", "msg":"WiredTiger message", "attr":{"message":{"ts_sec":1731299982, "ts_usec":825573, "thread":"1:0x7f541264a6c0", "session_name":"WT_SESSION.checkpoint", "category":"WT_VERB_CHECKPOINT_PROGRESS", "category_id":7, "verbose_level":"DEBUG_1", "verbose_level_id":1, "msg":"saving checkpoint snapshot min: 7, snapshot max: 7 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 31"}}}}
mongodb | {"t":{"$date":"2024-11-11T04:40:43.617+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpoint", "msg":"WiredTiger message", "attr":{"message":{"ts_sec":1731300043, "ts_usec":617120, "thread":"1:0x7f541264a6c0", "session_name":"WT_SESSION.checkpoint", "category":"WT_VERB_CHECKPOINT_PROGRESS", "category_id":7, "verbose_level":"DEBUG_1", "verbose_level_id":1, "msg":"saving checkpoint snapshot min: 8, snapshot max: 8 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 31"}}}}
mongodb | {"t":{"$date":"2024-11-11T04:41:43.642+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpoint", "msg":"WiredTiger message", "attr":{"message":{"ts_sec":1731300103, "ts_usec":642068, "thread":"1:0x7f541264a6c0", "session_name":"WT_SESSION.checkpoint", "category":"WT_VERB_CHECKPOINT_PROGRESS", "category_id":7, "verbose_level":"DEBUG_1", "verbose_level_id":1, "msg":"saving checkpoint snapshot min: 9, snapshot max: 9 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 31"}}}}
mongodb | {"t":{"$date":"2024-11-11T04:42:43.673+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpoint", "msg":"WiredTiger message", "attr":{"message":{"ts_sec":1731300163, "ts_usec":672997, "thread":"1:0x7f541264a6c0", "session_name":"WT_SESSION.checkpoint", "category":"WT_VERB_CHECKPOINT_PROGRESS", "category_id":7, "verbose_level":"DEBUG_1", "verbose_level_id":1, "msg":"saving checkpoint snapshot min: 11, snapshot max: 11 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 31"}}}}
mongodb | {"t":{"$date":"2024-11-11T04:43:43.699+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpoint", "msg":"WiredTiger message", "attr":{"message":{"ts_sec":1731300223, "ts_usec":699071, "thread":"1:0x7f541264a6c0", "session_name":"WT_SESSION.checkpoint", "category":"WT_VERB_CHECKPOINT_PROGRESS", "category_id":7, "verbose_level":"DEBUG_1", "verbose_level_id":1, "msg":"saving checkpoint snapshot min: 13, snapshot max: 13 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 31"}}}}

```


Se utiliza mongodb para el funcionamiento de la base de datos, de esta manera se pueden registrar usuarios y las tareas respectivas de cada usuario.


Se utiliza un entorno local para almacenar la base de datos.

✓ microservicios_tareas

> static


> templates


 app.py


 dockerfile


≡ requirements.txt


✓ microservicios_usuario

 dockerfile

 package-lock.json

 package.json

 server.js

 user.js

Se realizaron dos carpetas para cada servicio, uno contiene los microservicios de las tareas y otro los microservicios de usuario.

Como se puede ver, para las tareas se usó python, mientras que para los usuarios se optó por usar Javascript



FUNCIONAMIENTO

Iniciar Sesión

Usuario:

Contraseña:

Iniciar Sesión

Registrarse

Registrarse

Usuario:

Contraseña:

Registrarse

Iniciar sesión

Al ejecutarlo, podemos ver el inicio de la aplicación.

- Nos da el requisito de ingresar el usuario y la contraseña.
- Si queremos registrarnos, podemos abrir una pestaña aparte para poder hacerlo

Iniciar Sesión

Usuario no encontrado

Usuario:

Contraseña:

Iniciar Sesión

Registrarse

Si el usuario no ha sido registrado previamente, nos manda un mensaje de error, lo cual nos asegura que únicamente usuarios registrados puedan acceder.

Lista de Tareas

Comprar huevos

Ir al supermercado

Eliminar

Nueva tarea

Una vez dentro, podemos ver la lista de tareas pendientes que hemos ingresado.

Nos da la opción de eliminar esa tarea o agregar otra



CHAOS ENGINEERING

EN PROCESO...