Proyecto v1

COMPUTACIÓN TOLERANTE A FALLAS – D06 – 2024A

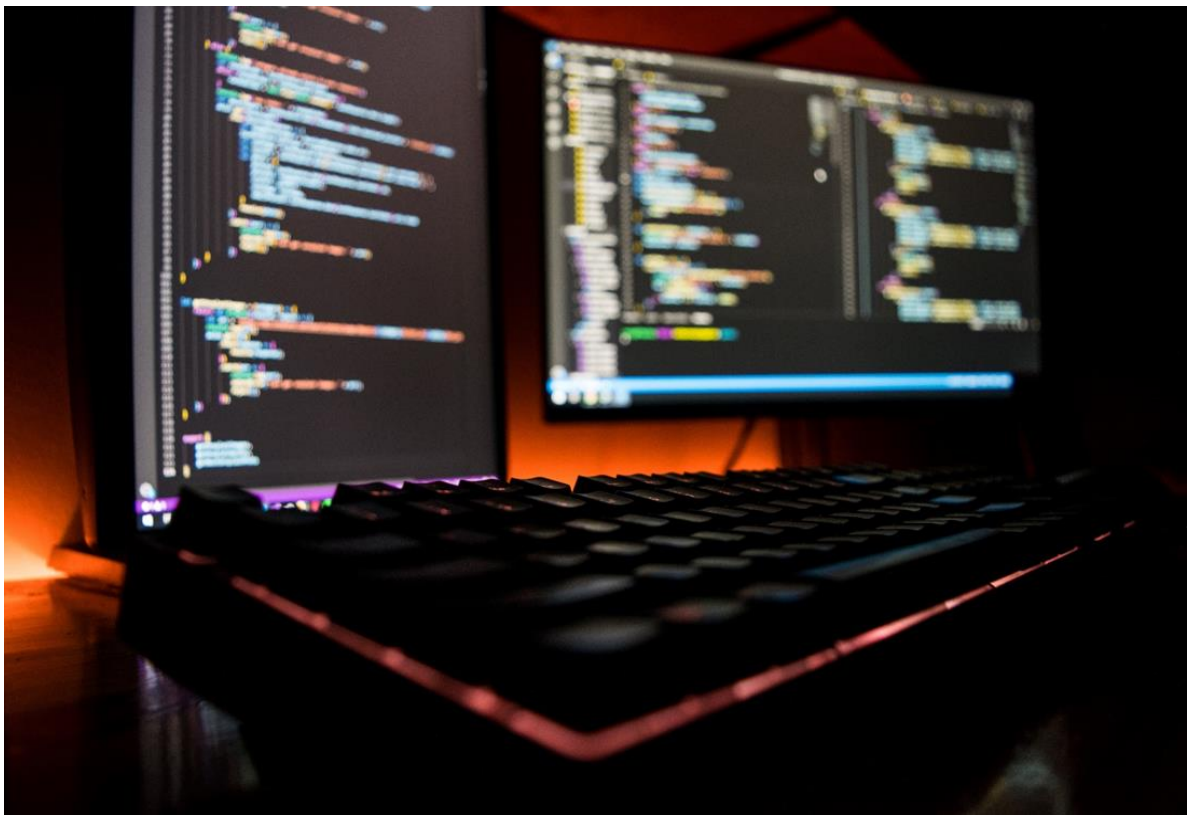
MICHEL EMANUEL LOPEZ FRANCO

Navarro Velazquez Andres – 218127792

Gonzalez Ramos Maria Fernanda - 218144778

Ramirez Rosas Randal Edin - 217470523

Fecha: 05/05/2024



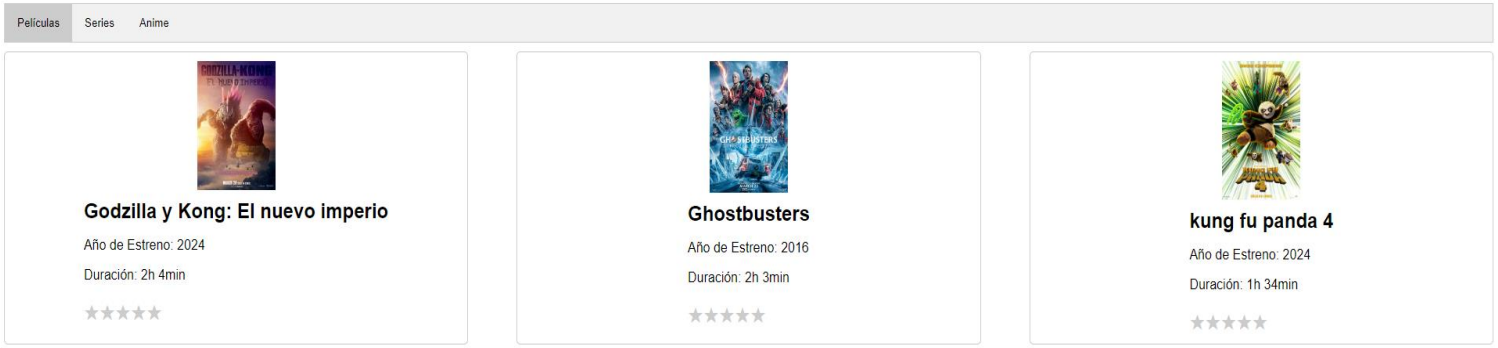
Introducción.

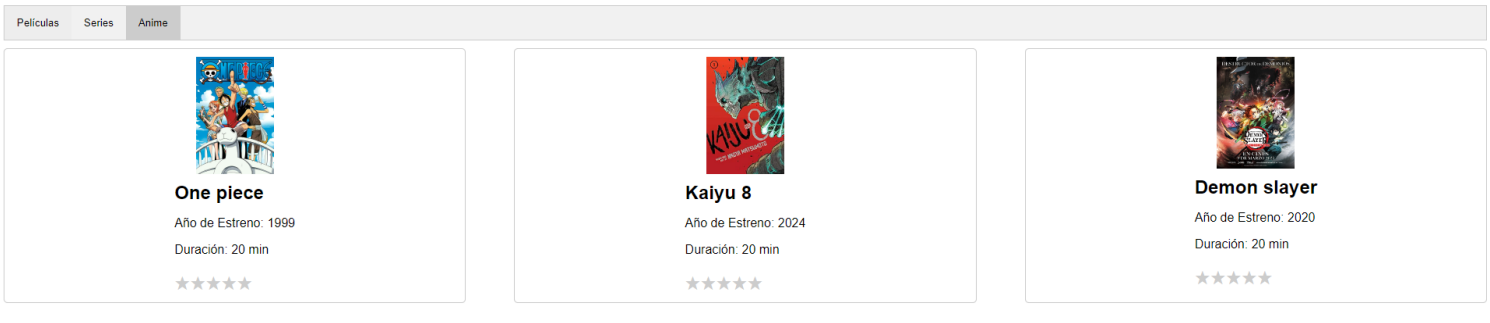
En este proyecto se realiza una aplicación de microservicios, en la que se dividen en servicios independientes y modulares, cada uno responsable de una tarea o funcionalidad específica. Se definieron las interfaces y las formas de comunicación entre los microservicios. Se empaqueta cada microservicio en un contenedor Docker, que incluye el código, las dependencias y la configuración para que el servicio funcione de manera independiente. Se utiliza un orquestador de contenedores como kubernetes para gestionar el ciclo de vida de los microservicios, incluyendo el despliegue, el escalado, el balanceo de carga y la recuperación ante fallos. Se implementan mecanismos de comunicación entre los microservicios, como APIs REST, mensajería asíncrona o llamadas remotas, para que puedan interactuar entre sí. Se configuran herramientas de monitorización y observabilidad, como Istio o Azure Monitor, para recopilar métricas, registros y trazas que permitan entender el comportamiento de la aplicación. Se automatiza el proceso de construcción, pruebas e implementación de los microservicios utilizando herramientas de integración y entrega continua (CI/CD). Se diseña la arquitectura para que los microservicios puedan escalarse horizontal y verticalmente según la demanda, y garantizar la disponibilidad de la aplicación incluso ante fallos. Se implementan mecanismos de autenticación, autorización y cifrado de comunicaciones entre los microservicios y con los clientes.

Frontend

En esta parte del proyecto decidimos utilizar html con css al igual que javascript para crear lo que sería la interfaz de nuestros servicios, como se puede ver en las imágenes hay una sección arriba a la izquierda donde se muestra un apartado de películas, series y animes, esos son los servicios. Como microservicios tenemos la puntuación que se puede expresar con estrellas, de 1 a 5 dependiendo de qué tanto éxito tuvo esa película, serie o anime.

Estos datos se alojan en ISTIO para recopilar métricas y registros, de igual manera está conectada al backend donde se trabaja con docker y kubernetes.





Código Frontend:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calificación de Películas y Series</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        .tab {
            overflow: hidden;
            border: 1px solid #ccc;
            background-color: #f1f1f1;
            margin-bottom: 10px;
```

```
        }
        .tab button {
            background-color: inherit;
            float: left;
            border: none;
            outline: none;
            cursor: pointer;
            padding: 14px 16px;
            transition: 0.3s;
        }
        .tab button:hover {
            background-color: #ddd;
        }
        .tab button.active {
            background-color: #ccc;
        }
        #movies, #series, #anime {
            display: flex;
            flex-wrap: wrap;
            justify-content: space-between;
        }
        .movie, .series, .anime {
            width: 30%;
            margin-bottom: 20px;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
            display: flex;
            flex-direction: column;
            align-items: center;
        }
```

```
        .movie img, .series img, .anime img {
            max-width: 100px;
            margin-bottom: 10px;
        }
        .movie h2, .series h2, .anime h2 {
            margin: 0;
        }
        .rating {
            display: flex;
            align-items: center;
        }
        .rating input[type="radio"] {
            display: none;
        }
        .rating label {
            cursor: pointer;
            font-size: 24px;
            color: #ccc;
            order: 5;
        }
        .rating label:hover,
        .rating label:hover ~ label,
        .rating input[type="radio"]:checked ~ label {
            color: orange;
        }
    </style>
</head>
<body>
    <div class="tab">
        <button class="tablinks" onclick="openCategory(event, 'moviesTab')"
id="defaultOpen">Películas</button>
```

```
        <button class="tablinks" onclick="openCategory(event,
'seriesTab')">Series</button>

        <button class="tablinks" onclick="openCategory(event,
'animeTab')">Anime</button>

    </div>



    <div id="moviesTab" class="tabcontent">

        <div id="movies"></div>

    </div>



    <div id="seriesTab" class="tabcontent">

        <div id="series"></div>

    </div>



    <div id="animeTab" class="tabcontent">

        <div id="anime"></div>

    </div>



    <script>

        function openCategory(evt, categoryName) {

            var i, tabcontent, tablinks;

            tabcontent = document.getElementsByClassName("tabcontent");

            for (i = 0; i < tabcontent.length; i++) {

                tabcontent[i].style.display = "none";

            }

            tablinks = document.getElementsByClassName("tablinks");

            for (i = 0; i < tablinks.length; i++) {

                tablinks[i].className = tablinks[i].className.replace("
active", "");

            }

            document.getElementById(categoryName).style.display = "block";

            evt.currentTarget.className += " active";
```

```
        }


        document.getElementById("defaultOpen").click();


        const movies = [
            {
                title: "Godzilla y Kong: El nuevo imperio",
                rating: 0,
                image:
"https://static.cinepolis.com/img/peliculas/45592/1/1/45592.jpg",
                year: 2024,
                duration: "2h 4min"
            },
            {
                title: "Ghostbusters",
                rating: 0,
                image:
"https://pics.filmaffinity.com/ghostbusters_frozen_empire-289306563-large.jpg",
                year: 2016,
                duration: "2h 3min"
            },
            {
                title: "kung fu panda 4",
                rating: 0,
                image:
"https://www.universalpictures.com.mx/tl_files/content/movies/kung_fu_panda_4/posters/01.jpg",
                year: 2024,
                duration: "1h 34min"
            },
        ];


        const series = [
```

```
            {
                title: "Lucifer",
                rating: 0,
                image: "https://www.lavanguardia.com/peliculas-
series/images/serie/poster/2016/1/w1280/wQh2ytX0f8IfC3b2mKpDGOpGTXS.jpg",
                year: 2016,
                duration: "6 temporadas"
            },
            {
                title: "The Big Bang Theory",
                rating: 0,
                image:
"https://www.formulatv.com/images/series/posters/100/185/dest_1.jpg",
                year: 2007,
                duration: "12 temporadas"
            },
            {
                title: "Sherlock Holmes",
                rating: 0,
                image: "https://pics.filmaffinity.com/Sherlock_Serie_de_TV-
635342236-large.jpg",
                year: 2010,
                duration: "4 temporadas"
            },


        ];


        const anime = [
            {
                title: "One piece",
                rating: 0,
                image: "https://pics.filmaffinity.com/one_piece-647985949-
large.jpg",
```

```
                year: 1999,
                duration: "20 min"
            },
            {
                title: "Kaiyu 8",
                rating: 0,
                image: "https://m.media-
amazon.com/images/I/51P+t1IHQBS._SY445_SX342_.jpg",
                year: 2024,
                duration: "20 min"
            },
            {
                title: "Demon slayer",
                rating: 0,
                image: "https://www.konnichiwafestival.com/wp-
content/uploads/2023/01/PC-70x100cm-DSWT2023_LR-717x1024.jpg",
                year: 2020,
                duration: "20 min"
            },
            // Add more movie objects as needed
        ];

        const moviesContainer = document.getElementById("movies");
        const seriesContainer = document.getElementById("series");
        const animeContainer = document.getElementById("anime");

        movies.forEach(movie => {
            const movieDiv = createMediaElement(movie);
            moviesContainer.appendChild(movieDiv);
        });

        series.forEach(serie => {
```

```
        const serieDiv = createMediaElement(serie);
        seriesContainer.appendChild(serieDiv);
    });

    anime.forEach(anime => {
        const animeDiv = createMediaElement(anime);
        animeContainer.appendChild(animeDiv);
    });

    function createMediaElement(media) {
        const mediaDiv = document.createElement("div");
        mediaDiv.classList.add("movie");

        const image = document.createElement("img");
        image.src = media.image;
        image.alt = media.title;
        mediaDiv.appendChild(image);

        const mediaInfoDiv = document.createElement("div");

        const title = document.createElement("h2");
        title.textContent = media.title;
        mediaInfoDiv.appendChild(title);

        const year = document.createElement("p");
        year.textContent = `Año de Estreno: ${media.year}`;
        mediaInfoDiv.appendChild(year);

        const duration = document.createElement("p");
        duration.textContent = `Duración: ${media.duration}`;
        mediaInfoDiv.appendChild(duration);
```

```
        const ratingDiv = document.createElement("div");
        ratingDiv.classList.add("rating");

        const stars = [5, 4, 3, 2, 1];
        stars.forEach(star => {
            const starInput = document.createElement("input");
            starInput.setAttribute("type", "radio");
            starInput.setAttribute("name", `${media.title}-rating`);
            starInput.setAttribute("id", `${media.title}-star${star}`);
            starInput.setAttribute("value", star);

            const starLabel = document.createElement("label");
            starLabel.setAttribute("for", `${media.title}-star${star}`);
            starLabel.textContent = "★";
            starLabel.style.order = star;

            ratingDiv.appendChild(starInput);
            ratingDiv.appendChild(starLabel);

            starInput.addEventListener("click", function() {
                media.rating = star;
                console.log(`${media.title} rated ${star}`);
            });
        });

        mediaInfoDiv.appendChild(ratingDiv);

        mediaDiv.appendChild(mediaInfoDiv);

        return mediaDiv;
```

```
        }
    </script>
</body>
</html>
```

Backend

Primero creamos los servicios, en nuestro caso serán 3 servicios los cuales se basan en las películas, series y animes, estos son creados por los archivos .py junto con los dockerfiles

```python
# servicio1.py

from flask import Flask

app = Flask(__name__)

@app.route("/mensaje")

def mensaje():

    return "Peliculas"



if __name__ == "__main__":

    app.run(debug=True, host='0.0.0.0', port=5000)
```

```python
# servicio2.py

from flask import Flask

app = Flask(__name__)

@app.route("/mensaje")

def mensaje():
```

```
    return "Series"

if __name__ == "__main__":

    app.run(debug=True, host='0.0.0.0', port=5000)
```

```
# servicio3.py

from flask import Flask

app = Flask(__name__)

@app.route("/mensaje")

def mensaje():

    return "Anime"

if __name__ == "__main__":

    app.run(debug=True, host='0.0.0.0', port=5000)
```

```
# Dockerfile para servicio1

FROM python:3.8-slim

WORKDIR /app

COPY servicio1.py /app

RUN pip install flask

EXPOSE 5000

CMD ["python", "servicio1.py"]
```

```
# Dockerfile para servicio2

FROM python:3.8-slim

WORKDIR /app

COPY servicio2.py /app
```

```
RUN pip install flask

EXPOSE 5000

CMD ["python", "servicio2.py"]
```

```
# Dockerfile para servicio3

FROM python:3.8-slim

WORKDIR /app

COPY servicio3.py /app

RUN pip install flask

EXPOSE 5000

CMD ["python", "servicio3.py"]
```

Docker Compose para definir los servicios

```
version: "3"

services:

  servicio1:

    build: ./servicio1

    ports:

      - "5001:5000"

  servicio2:

    build: ./servicio2

    ports:

      - "5002:5000"

  servicio3:

    build: ./servicio3

    ports:
```

```
      - "5003:5000"
```

Para poder desplegar los servicios en Kubernetes, se necesita crear un archivo de configuración YAML para cada servicio, que incluya un Deployment y un Service.

```
# servicio1-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: servicio1

spec:

  replicas: 3

  selector:

    matchLabels:

      app: Peliculas

  template:

    metadata:

      labels:

        app: peliculas

    spec:

      containers:

      - name: peliculas

        image: Cine/servicio1:latest  #

        ports:

        - containerPort: 5000
```

```
# servicio1-service.yaml

apiVersion: v1
```

```
kind: Service

metadata:

  name: peliculas

spec:

  selector:

    app: peliculas

  ports:

    - protocol: TCP

      port: 80

      targetPort: 5000
```

```
kubectl apply -f servicio1-deployment.yaml

kubectl apply -f servicio1-service.yaml
```

Conclusión:

En conclusión, documentamos el proceso que realizamos para el desarrollo de la aplicación de microservicios. Dividimos servicios y microservicios independientes. Utilizamos contenedores docker para empaquetar cada microservicio. También implementamos kubernetes como orquestador de contenedores.