Proyecto Final: Glassdoor Univalle

Integrantes

- Pedro Bernal Londoño - 2259548
- Laura Tatiana Coicue – 2276652
- Laura Sofia Peñaloza - 2259485
- Santiago Reyes Rodriguez - 2259738
- Jota Lopez Ramirez - 2259394
- Esmeralda Rivas Guzmán - 2259580

Introducción

Glassdoor Univalle es una plataforma diseñada para que los estudiantes de la Universidad del Valle puedan evaluar a sus profesores en función de sus experiencias en los cursos que han tomado. Este proyecto permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre las evaluaciones de los docentes y utiliza técnicas de web scraping para obtener información de los profesores habilitados para ser calificados desde un sistema externo.

Objetivos del Proyecto

1. Diseñar y desarrollar una plataforma web que permita a los estudiantes calificar a sus profesores.
2. Implementar una arquitectura eficiente que integre un backend, un frontend y una base de datos.
3. Utilizar contenedores Docker para la solución local y escalar la arquitectura en la nube con Kubernetes.
4. Proporcionar un sistema replicable y escalable, con una arquitectura clara y modular.

Arquitectura

1. Frontend: Construido con Astro, sirve como la interfaz de usuario para que los estudiantes interactúen con la plataforma.
2. Backend: Implementado con Node.js, utilizando Express y TypeScript, gestiona las operaciones del sistema, incluida la autenticación y el manejo de datos.
3. Base de Datos: Utiliza PostgreSQL para almacenar información de estudiantes, evaluaciones y detalles de los docentes.

Solución Local

La solución local está contenida utilizando Docker Compose, lo que permite gestionar de manera eficiente los diferentes componentes del sistema. Cada servicio (frontend, backend y

base de datos) se despliega en un contenedor aislado, garantizando una fácil configuración y portabilidad.

Imágenes Personalizadas

Se utilizaron imágenes personalizadas para el backend y el frontend, por configuraciones específicas y para garantizar que las dependencias necesarias estuvieran presentes en los contenedores.

Dockerfile del Frontend:

```
FROM node:22.12.0-alpine AS runtime

WORKDIR /app
COPY . .

EXPOSE 3000

RUN npm install
RUN npm run build

CMD ["node", "dist/server/entry.mjs"]
```

Dockerfile del Backend:

```
FROM node:22.12.0-alpine

WORKDIR /app
COPY . /app

EXPOSE 5000

RUN npm install
RUN npm run build

CMD ["npm", "start"]
```

Configuración con Docker Compose

El archivo docker-compose.yml contiene la configuración de los servicios y sus dependencias, así como la configuración de la red interna para permitir la comunicación entre los contenedores.

```
services:
  postgres:
    image: postgres:15
    container_name: postgres_db
    restart: always
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    volumes:
      - postgres-data:/var/lib/postgresql/data
      -
./db/init/init-db.sql:/docker-entrypoint-initdb.d/init-db.sql
    networks:
      - backend_network
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER}"]
      interval: 10s
      timeout: 5s
      retries: 5

  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: backend_app
    restart: always
    dns:
      - 8.8.8.8
      - 4.4.4.4
    depends_on:
      postgres:
        condition: service_healthy
    environment:
      PORT: 5000
      JWT_SECRET: ${JWT_SECRET}
      DATABASE_URL: ${DATABASE_URL}
    networks:
      - backend_network
```

```
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    container_name: frontend_app
    restart: always
    ports:
      - "3000:3000"
    environment:
      API_BASE_URL: ${API_BASE_URL}
      PORT: 3000
      HOST: 0.0.0.0
    depends_on:
      - backend
    networks:
      - backend_network
volumes:
  postgres-data:

networks:
  backend_network:
    driver: bridge
```

Para ejecutar localmente, se utiliza:

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

Los servicios estarán disponibles en las siguientes URLs:

- Frontend: http://localhost:3000

Solución en la nube

Minikube

La solución en la nube se implementó utilizando Kubernetes en un clúster de Minikube. Se utilizó una configuración similar a la local, pero con ajustes específicos para garantizar la escalabilidad y la disponibilidad de los servicios.

Despliegue en Kubernetes

Pasos para desplegar la solución en Kubernetes:

Se recomienda utilizar herramientas como Kompose para convertir el archivo docker-compose.yml en configuraciones de Kubernetes, lo que facilita la creación de manifiestos iniciales. Por ejemplo:

```
kompose convert
```

Después de generar los manifiestos, es posible ajustarlos manualmente para cumplir con los requisitos específicos del proyecto, por simplificar la configuración se subieron las imágenes personalizadas a Docker Hub y se utilizaron en los manifiestos de Kubernetes.

Configuración de los Manifiestos

Deployment para el Backend (con 3 réplicas):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image: code3743/backend:latest
          ports:
            - containerPort: 5000
          env:
            - name: PORT
              value: "5000"
            - name: JWT_SECRET
              value: ProyectoFinalInfraestructuraG50
            - name: DATABASE_URL
```

```
              value:
postgres://admin:admin_password@postgres:5432/glassdoor_db
---
apiVersion: v1
kind: Service
metadata:
  name: backend
spec:
  selector:
    app: backend
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
  type: ClusterIP
```

Deployment para el Frontend:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: code3743/frontend:latest
          ports:
            - containerPort: 3000
          env:
            - name: API_BASE_URL
              value: http://backend:5000
            - name: PORT
              value: "3000"
            - name: HOST
```

```
              value: 0.0.0.0
---
apiVersion: v1
kind: Service
metadata:
  name: frontend
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 3000
  type: LoadBalancer
```

Deployment para la Base de Datos:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
      - name: postgres
        image: postgres:15
        env:
        - name: POSTGRES_USER
          value: admin
        - name: POSTGRES_PASSWORD
          value: admin_password
        - name: POSTGRES_DB
          value: glassdoor_db
        volumeMounts:
        - name: init-db-volume
```

```
          mountPath: /docker-entrypoint-initdb.d/init-db.sql
          subPath: init-db.sql
      volumes:
      - name: init-db-volume
        configMap:
          name: init-db-sql
---
apiVersion: v1
kind: Service
metadata:
  name: postgres
spec:
  selector:
    app: postgres
  ports:
    - protocol: TCP
      port: 5432
      targetPort: 5432
  type: ClusterIP
```

Despliegue en Minikube

Para desplegar en Minikube, se utilizan los siguientes comandos:

```
minikube start
kubectl apply -f backend-deployment.yaml
kubectl apply -f frontend-deployment.yaml
kubectl apply -f postgres-deployment.yaml
kubectl apply -f postgres-pvc.yaml
```

Para acceder al frontend, se puede utilizar el siguiente comando:

```
minikube service frontend
```

Para automatizar el proceso de despliegue, se puede utilizar el script deploy.sh:

```
sh deploy.sh
```

Google Cloud

Para desplegar la solución en la nube de Google Cloud Platform, se utilizó Google Kubernetes Engine (GKE) para crear un clúster de Kubernetes y desplegar los servicios de frontend, backend y base de datos en la nube, los deployments y servicios de Kubernetes se reutilizaron de la solución con Minikube.

Los servicios estarán disponibles en las siguientes URLs:

- Frontend: http://104.197.220.104/

Seguridad

Una de las principales consideraciones de seguridad en este proyecto fue garantizar que solo el servicio de frontend esté expuesto al exterior. Tanto el archivo docker-compose.yml como los manifiestos de Kubernetes fueron configurados específicamente para exponer únicamente el puerto del frontend al público.

- Backend: El servicio de backend utiliza una red interna en Docker y un tipo de servicio ClusterIP en Kubernetes, lo que garantiza que no sea accesible desde fuera del entorno local o del clúster.
- Base de Datos: La base de datos PostgreSQL se configura con un servicio ClusterIP en Kubernetes, lo que significa que solo los servicios internos pueden acceder a ella. Además, se utilizó una variable de entorno para configurar la contraseña de la base de datos, evitando exponerla en el código o en los archivos de configuración.

Prueba de Estrés

Para probar la escalabilidad y el rendimiento de la solución, se realizaron pruebas de estrés con un script que solicita al frontend un número específico de solicitudes para simular una carga alta en el sistema, para ejecutar el script se puede utilizar el siguiente comando:

```
node stress-test.js IP_FRONTEND PORT_FRONTEND
```

Resultados de las Pruebas de Estrés

Los resultados de las pruebas de estrés mostraron que la solución en la nube es capaz de manejar una carga alta de solicitudes sin problemas, gracias a la escalabilidad y la distribución de los servicios en Kubernetes, se logró mantener un tiempo de respuesta bajo y una alta disponibilidad del sistema en todos los enfoques probados, sin embargo, para muestras pequeñas, los entornos locales demostraron ventajas significativas debido a su

menor latencia inherente, la ausencia de dependencias de red externa y tiempos de respuesta más consistentes en escenarios de baja carga.

Análisis Comparativo

Se recopilaron datos para diferentes cantidades de solicitudes, comenzando en 100 y aumentando hasta 10,000.

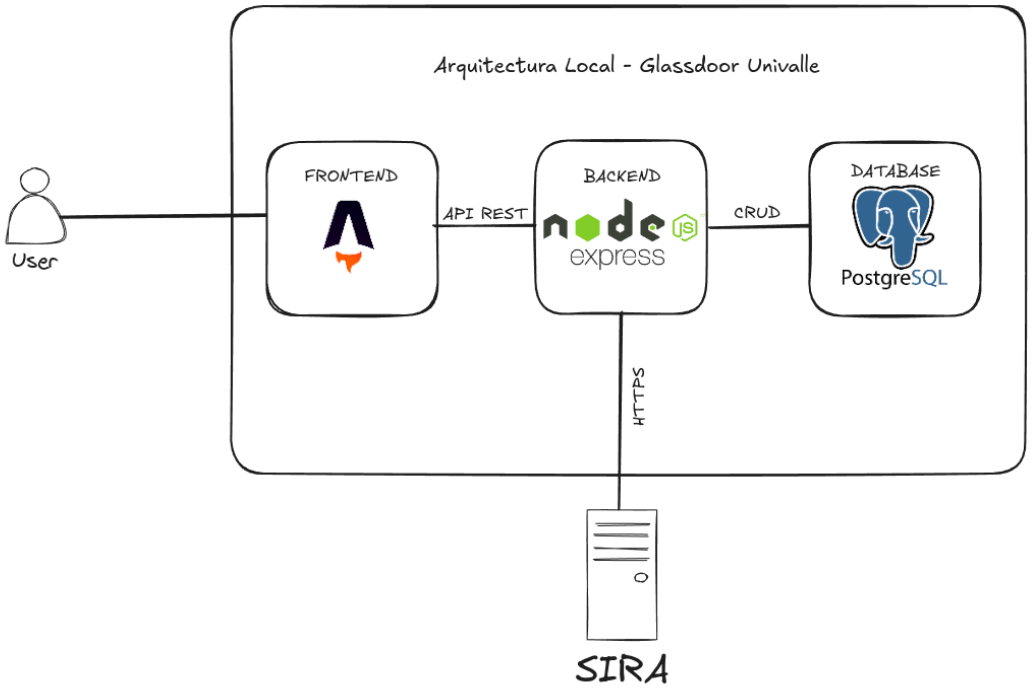
Número de Solicitudes	Local (Docker Compose)	Minikube	Google Cloud
100	15 ms	18 ms	25 ms
200	25 ms	28 ms	35 ms
500	60 ms	50 ms	40 ms
1,000	150 ms	120 ms	45 ms
10,000	1,200 ms	900 ms	60 ms

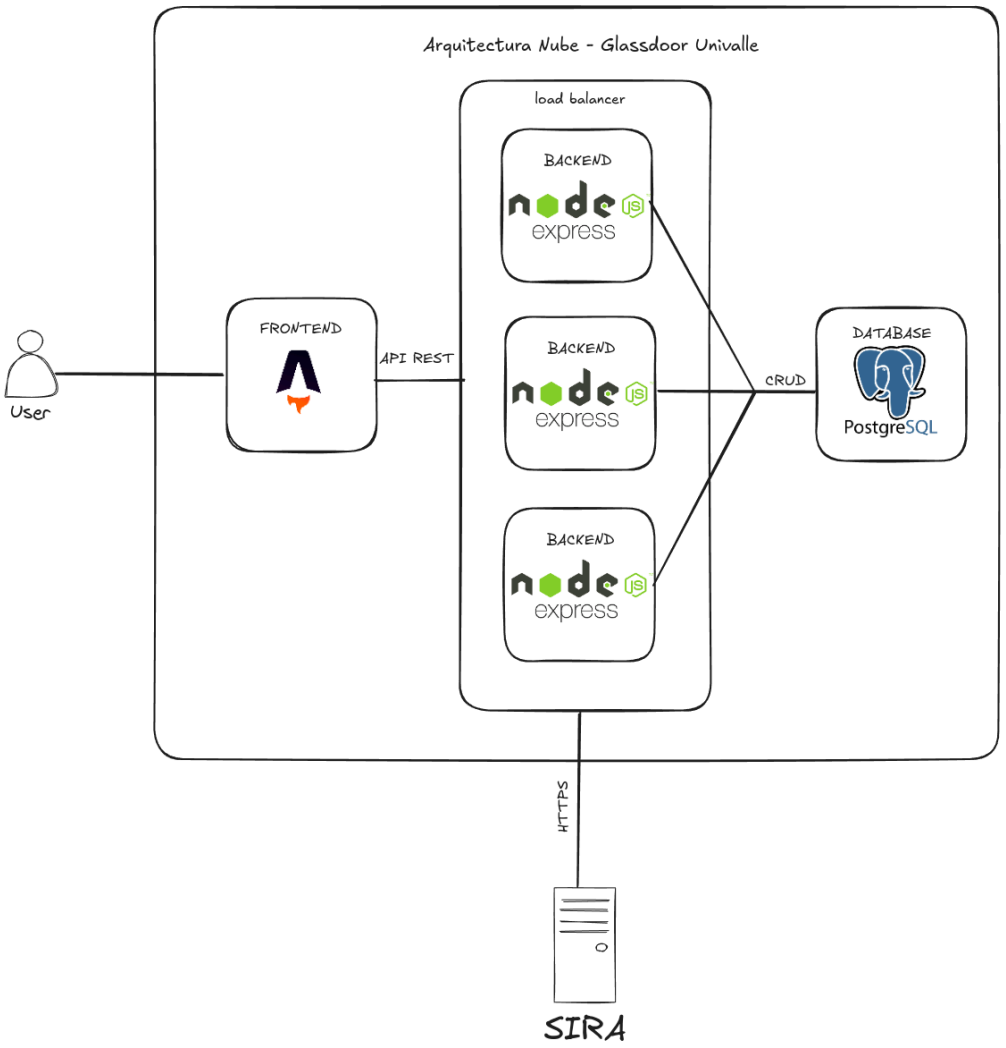
Nota: Los valores indicados corresponden al tiempo promedio de respuesta para cada enfoque y carga de solicitudes, esos valores pueden variar dependiendo de la máquina y la red en la que se realicen las pruebas.

1. Cargas pequeñas (≤1,000 solicitudes):
    - Docker Compose tuvo tiempos de respuesta rápidos y consistentes debido a la ausencia de latencia externa.
    - Google Cloud presentó una ligera desventaja en este escenario debido a la latencia inherente a las comunicaciones en la nube.
2. Cargas medias (10,000–100,000 solicitudes):
    - Google Cloud fue el enfoque más eficiente, manteniendo tiempos de respuesta bajos gracias a su capacidad de escalado.
    - Minikube mostró un desempeño aceptable pero limitado por la infraestructura física.
    - Docker Compose no fue capaz de manejar estas cargas debido a la saturación de los recursos locales.
3. Cargas altas (>100,000 solicitudes):
    - Google Cloud sobresalió al mantener tiempos de respuesta consistentes incluso bajo cargas extremas.
    - Los otros enfoques fallaron al alcanzar el límite de los recursos disponibles.

Diagrama de Arquitectura

El diagrama de arquitectura muestra la relación entre los diferentes componentes del sistema y cómo interactúan entre sí.





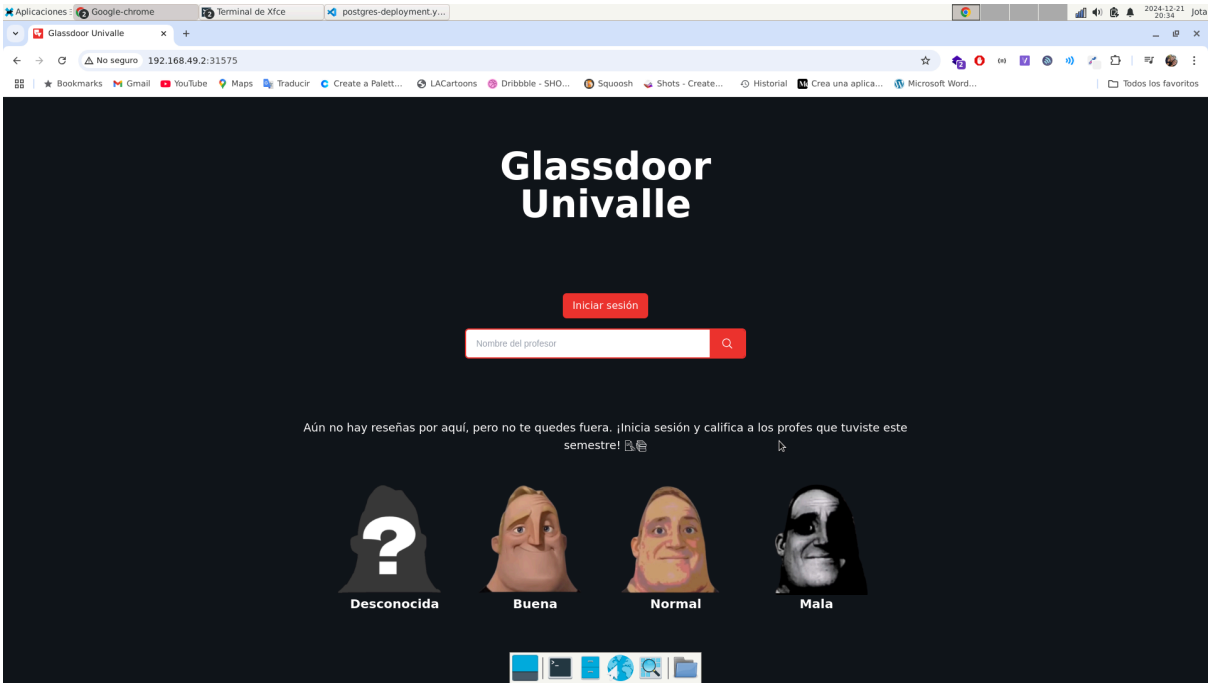
Conclusiones

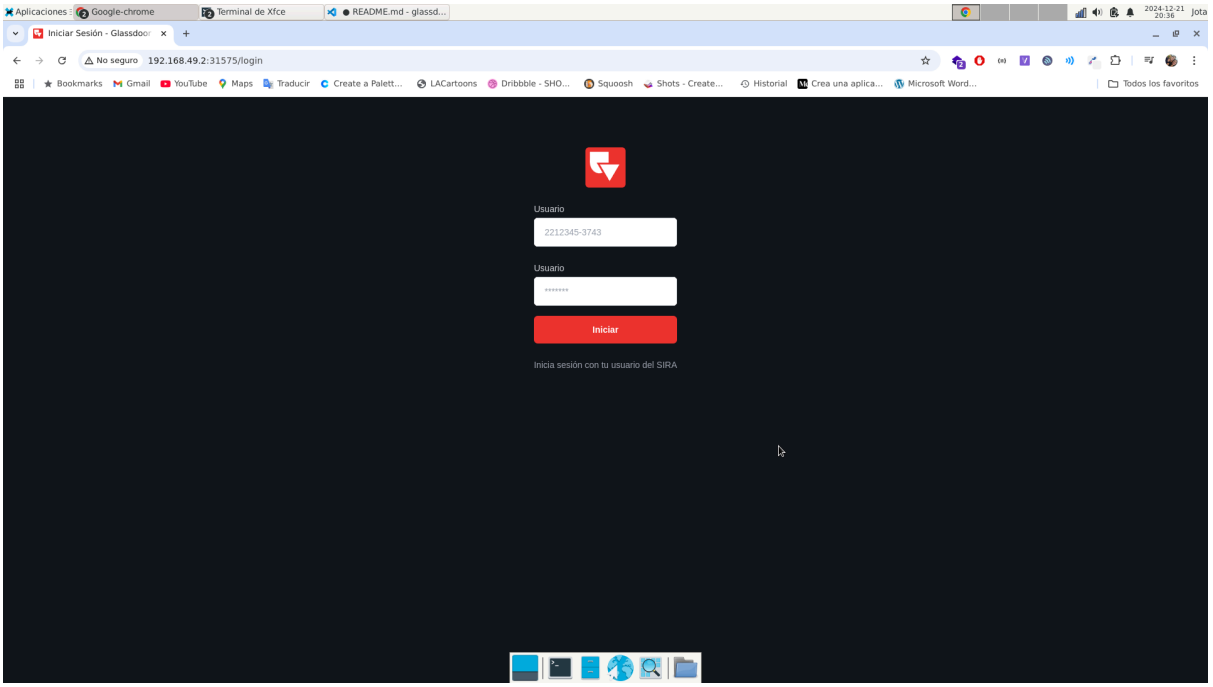
El proyecto Glassdoor Univalle ha demostrado la viabilidad de utilizar diversas tecnologías para el desarrollo de una plataforma web escalable, tanto en entornos locales como en la nube, la implementación con Docker Compose en el entorno local permitió gestionar de manera sencilla los contenedores de los diferentes componentes (frontend, backend y base de datos), lo que facilitó la configuración y el despliegue rápido, sin embargo, Docker Compose presenta limitaciones en términos de escalabilidad y balanceo de carga, ya que no ofrece soluciones automáticas para estos aspectos, lo que requiere intervenciones manuales y el uso de herramientas adicionales, como balanceadores de carga externos.

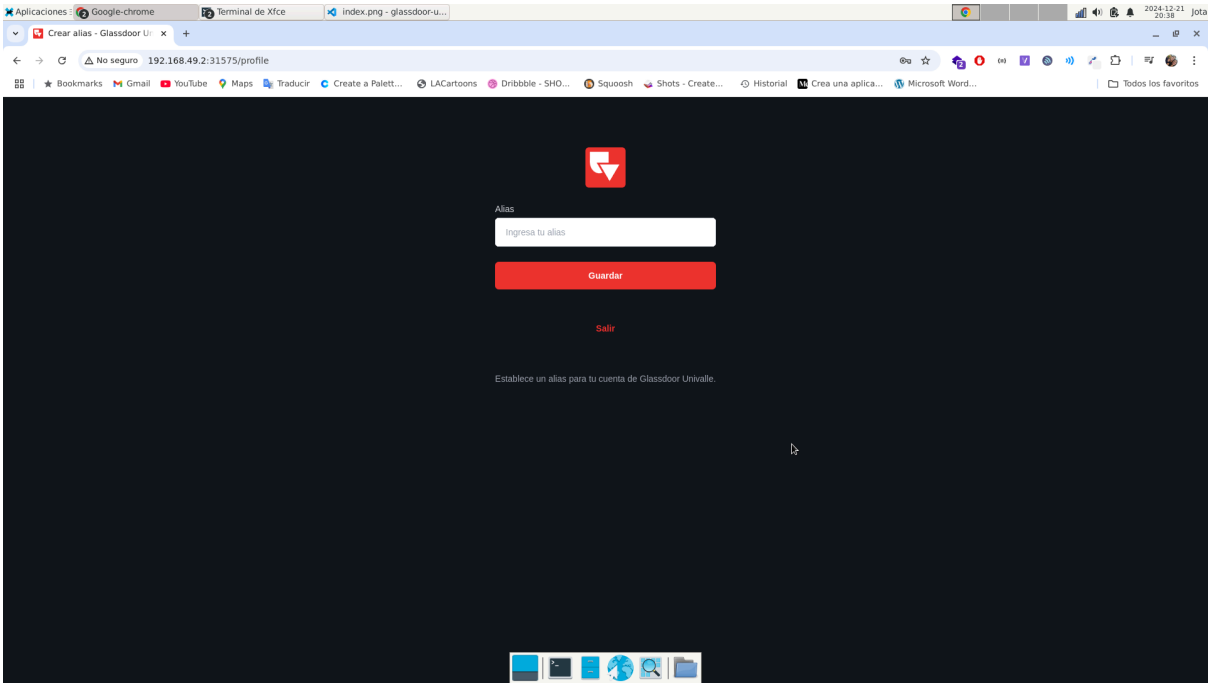
Por otro lado, el uso de Kubernetes proporcionó una solución más adecuada para gestionar la plataforma de manera dinámica, permitiendo una distribución eficiente de los servicios y facilitando la escalabilidad automática, Kubernetes también garantizó una mayor disponibilidad, ya que los servicios se distribuyen y gestionan de forma más flexible, sin la necesidad de configuraciones estáticas.

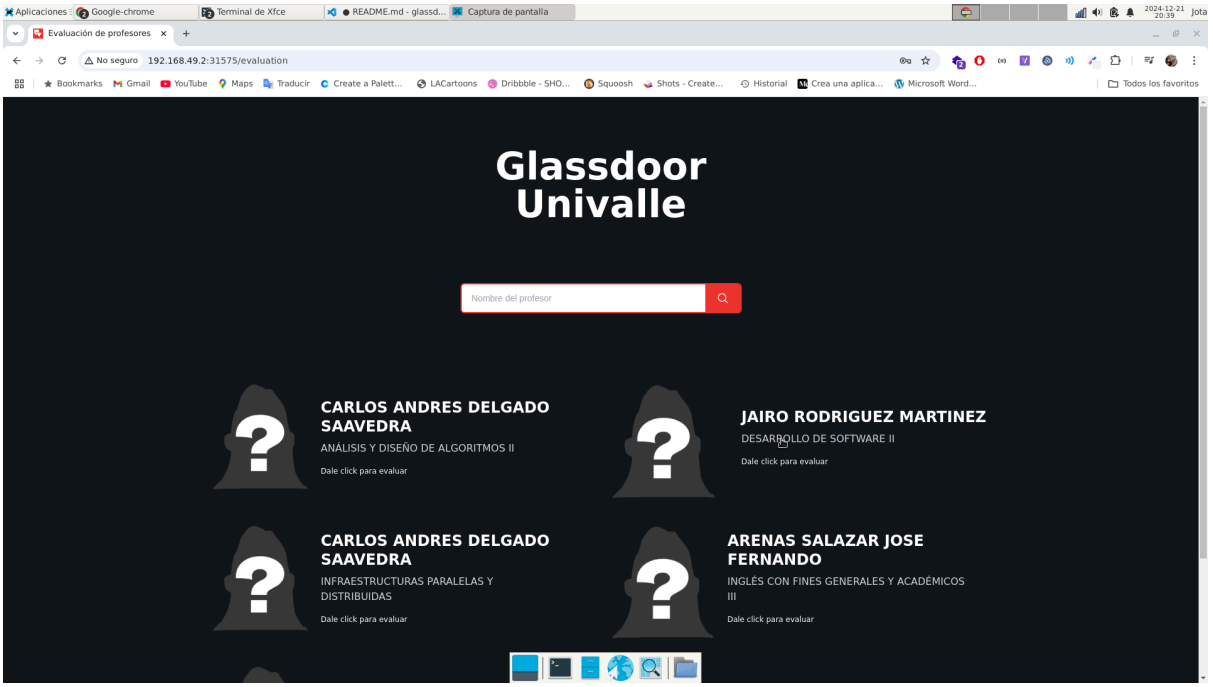
Ahora bien, la elección entre Docker Compose y Kubernetes depende de diversos factores, como los objetivos específicos del proyecto, los requisitos de escalabilidad, la complejidad de la infraestructura, y la capacidad del equipo para gestionar y mantener la solución, ambas herramientas tienen su lugar dependiendo del contexto en el que se utilicen, no es una decisión definitiva o excluyente, y en muchos casos, pueden complementarse en diferentes etapas del ciclo de vida del proyecto.

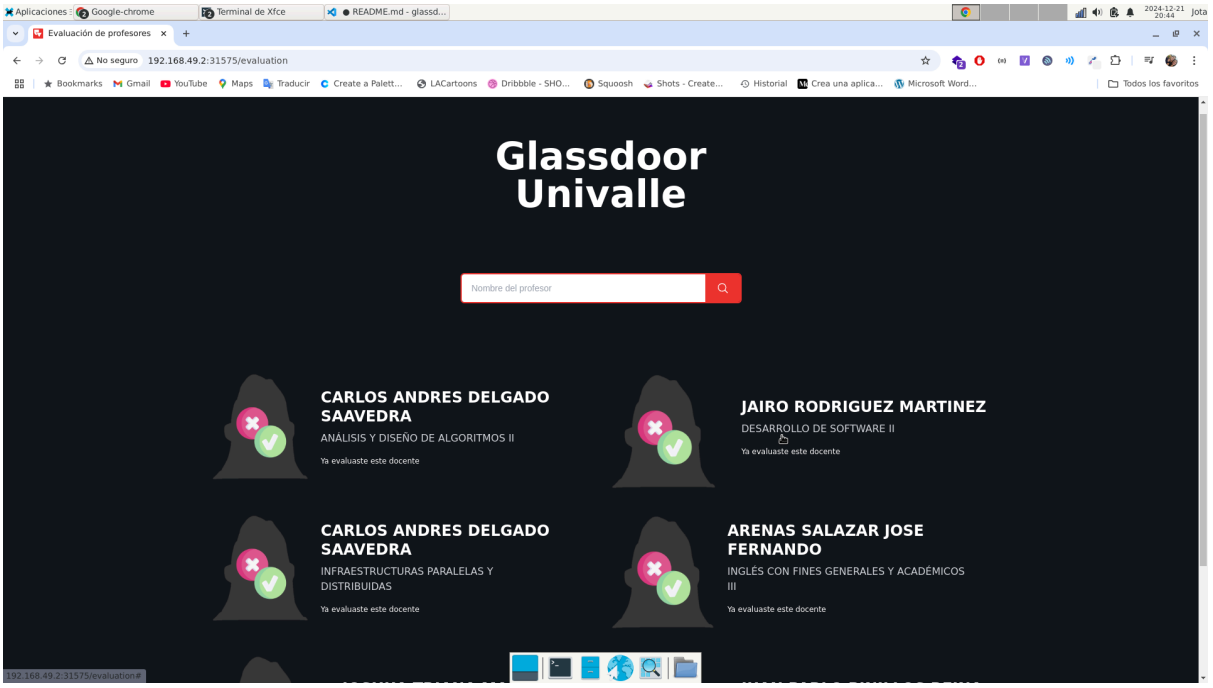
Capturas de Pantalla

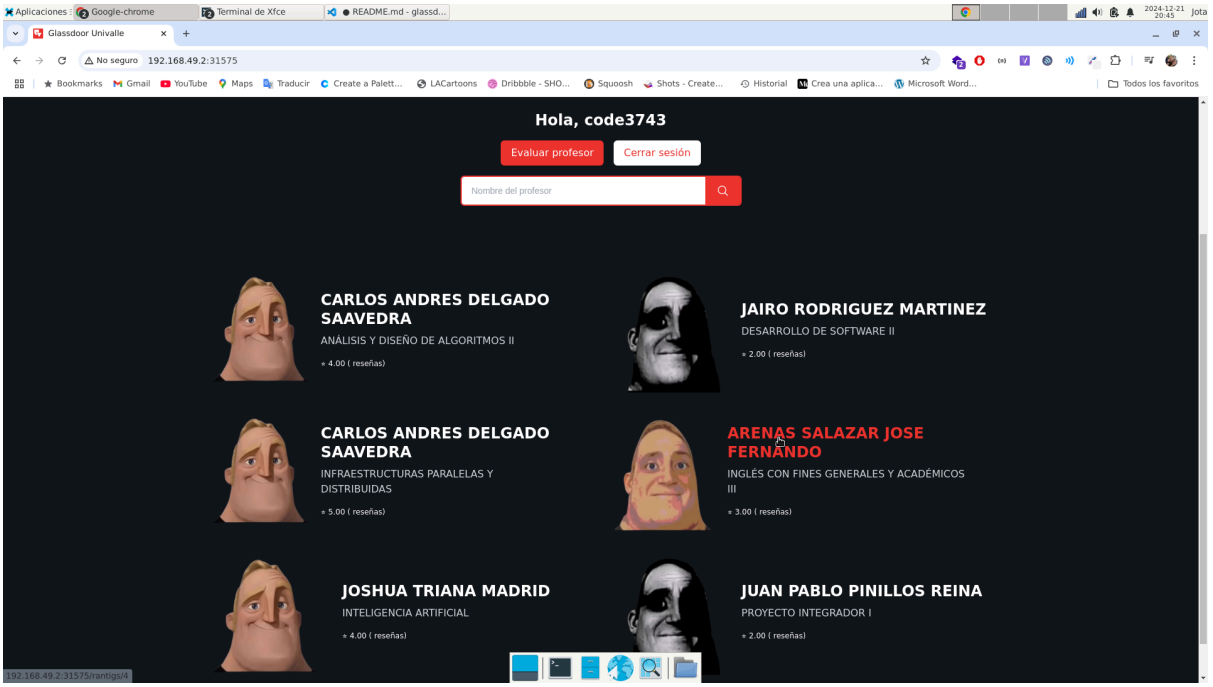


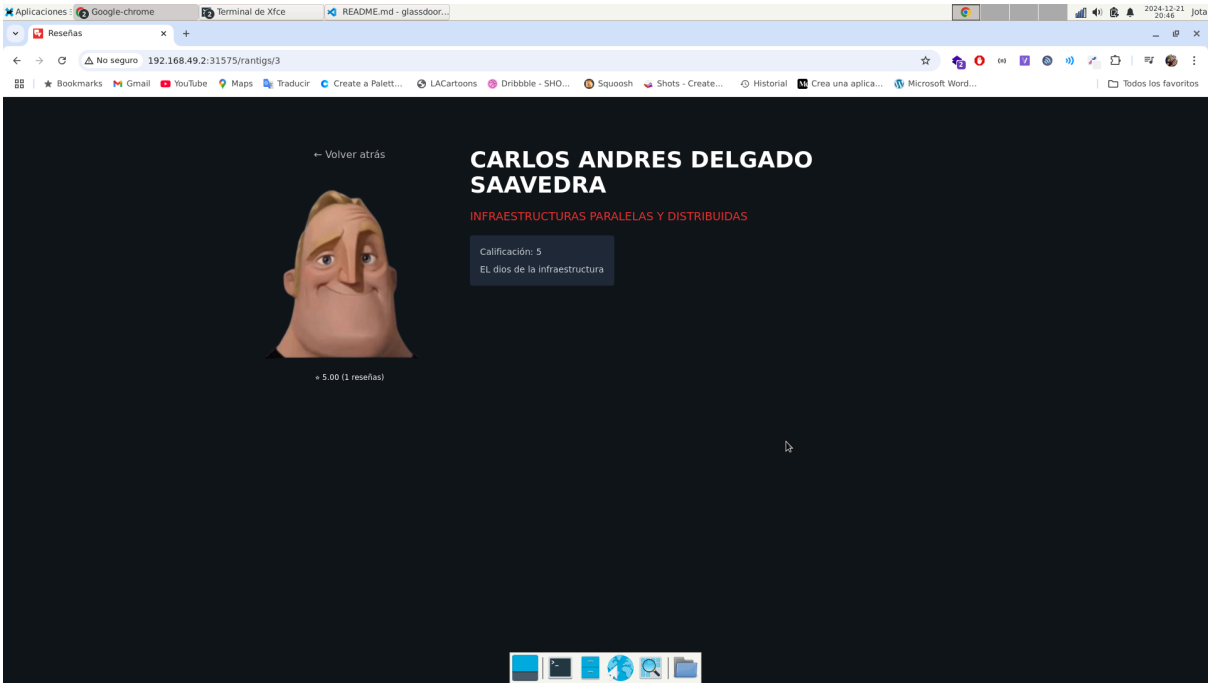

Aplicaciones
Google-chrome
Terminal de Xfce
README.md - glassd...
2024-12-21 20:36
Jota
Iniciar Sesión - Glassdoor
No seguro
192.168.49.2:31575/login
Bookmarks
Gmail
YouTube
Maps
Traducir
Create a Palett...
LACartoons
Dribbble - SHO...
Squoosh
Shots - Create...
Historial
Crea una aplica...
Microsoft Word...
Todos los favoritos
Usuario
2212345-3743
Usuario
****
Iniciar
Inicia sesión con tu usuario del SIRA


Aplicaciones
Google-chrome
Terminal de Xfce
index.png - glassdoor-u...
2024-12-21 20:38
Jota
Crear alias - Glassdoor Un
No seguro
192.168.49.2:31575/profile
Bookmarks
Gmail
YouTube
Maps
Traducir
Create a Palett...
LACartoons
Dribbble - SHO...
Squoosh
Shots - Create...
Historial
Crea una aplica...
Microsoft Word...
Todos los favoritos
Alias
Ingresa tu alias
Guardar
Salir
Establece un alias para tu cuenta de Glassdoor Univalle.


Aplicaciones
Google-chrome
Terminal de Xfce
README.md - glassd...
Captura de pantalla
2024-12-21 20:39
Jota
Evaluación de profesores
No seguro
192.168.49.2:31575/evaluation
Bookmarks
Gmail
YouTube
Maps
Traducir
Create a Palett...
LACartoons
Dribbble - SHO...
Squoosh
Shots - Create...
Historial
Crea una aplica...
Microsoft Word...
Todos los favoritos
Glassdoor Univalle
Nombre del profesor
CARLOS ANDRES DELGADO SAAVEDRA
ANÁLISIS Y DISEÑO DE ALGORITMOS II
Dale click para evaluar
JAIRO RODRIGUEZ MARTINEZ
DESARROLLO DE SOFTWARE II
Dale click para evaluar
CARLOS ANDRES DELGADO SAAVEDRA
INFRAESTRUCTURAS PARALELAS Y DISTRIBUIDAS
Dale click para evaluar
ARENAS SALAZAR JOSE FERNANDO
INGLÉS CON FINES GENERALES Y ACADÉMICOS III
Dale click para evaluar


Aplicaciones
Google-chrome
Terminal de Xfce
README.md - glassd...
2024-12-21 20:44
Jota
Evaluación de profesores
No seguro
192.168.49.2:31575/evaluation
Bookmarks
Gmail
YouTube
Maps
Traducir
Create a Palett...
LACartoons
Dribbble - SHO...
Squoosh
Shots - Create...
Historial
Crea una aplica...
Microsoft Word...
Todos los favoritos
Glassdoor Univalle
Nombre del profesor
CARLOS ANDRES DELGADO SAAVEDRA
ANÁLISIS Y DISEÑO DE ALGORITMOS II
Ya evaluaste este docente
JAIRO RODRIGUEZ MARTINEZ
DESARROLLO DE SOFTWARE II
Ya evaluaste este docente
CARLOS ANDRES DELGADO SAAVEDRA
INFRAESTRUCTURAS PARALELAS Y DISTRIBUIDAS
Ya evaluaste este docente
ARENAS SALAZAR JOSE FERNANDO
INGLÉS CON FINES GENERALES Y ACADÉMICOS III
Ya evaluaste este docente
192.168.49.2:31575/evaluation#


Hola, code3743
Evaluar profesor
Cerrar sesión
Nombre del profesor
CARLOS ANDRES DELGADO SAAVEDRA
ANÁLISIS Y DISEÑO DE ALGORITMOS II
⋆ 4.00 (reseñas)
JAIRO RODRIGUEZ MARTINEZ
DESARROLLO DE SOFTWARE II
⋆ 2.00 (reseñas)
CARLOS ANDRES DELGADO SAAVEDRA
INFRAESTRUCTURAS PARALELAS Y DISTRIBUIDAS
⋆ 5.00 (reseñas)
ARENAS SALAZAR JOSE FERNANDO
INGLÉS CON FINES GENERALES Y ACADÉMICOS III
⋆ 3.00 (reseñas)
JOSHUA TRIANA MADRID
INTELIGENCIA ARTIFICIAL
⋆ 4.00 (reseñas)
JUAN PABLO PINILLOS REINA
PROYECTO INTEGRADOR I
⋆ 2.00 (reseñas)


← Volver atrás
CARLOS ANDRES DELGADO SAAVEDRA
INFRAESTRUCTURAS PARALELAS Y DISTRIBUIDAS
Calificación: 5
EL dios de la infraestructura
⋆ 5.00 (1 reseñas)