

# PROYECTO 1-DESARROLLO Y DESPLIEGUE DE UNA APLICACIÓN EN LA NUBE

Presentado:

Niédila Braga 202515263

Luisa Quiroga 202222982

Diego Bueno 202314082

Para:

Seguridad Cloud

## Arquitectura

Imagen adjunta: Arquitectura blog BitSeguro.png.

## Configuración de la infraestructura

1. **Base de datos:** Se usó una base de datos Cloud SQL

Filter Enter property name or value										
<input type="checkbox"/>	Status	Instance ID ⓘ ↑	Issues	Cloud SQL edition	Type	Public IP address	Private IP address	Instance connection name	High availability	Loc Actions
<input type="checkbox"/>	✓	blog-db	<div>Allows unencrypted direct connections</div> <div>Auditing not enabled</div> <div>No password policy</div>	Enterprise	PostgreSQL 13	34.46.113.180 ⓘ	10.120.208.3	fleetproject-403015us-... ▾	ENABLE	us- ⋮

2. **Cluster GKE**

autopilot-cluster-1

DETAILS

STORAGE

OBSERVABILITY

LOGS

APP ERRORS (18)

Cluster basics

Name	autopilot-cluster-1	
Tier	Standard	
Mode	Autopilot	
Location type	Regional	
Region	us-central1	
Default node zones	us-central1-b us-central1-c us-central1-f us-central1-a	
Release channel	Regular channel	
Version	1.31.5-gke.1169000	UPGRADE AVAILABLE
Current COS version	cos-117-18613-75-114	
End of standard support	Dec 21, 2025	
End of extended support	Oct 21, 2026	
Rollout sequence	To use rollout sequencing, register your cluster to a fleet	

Upgrades

Auto-upgrade status	Active	
Minor version auto-upgrade target	Unavailable	
Patch version auto-upgrade target	Unavailable	
Upgrade history	SHOW UPGRADE HISTORY	

Automation

Maintenance window	Any time	
Maintenance exclusions	None	
Notifications	Disabled	
Vertical Pod Autoscaling	Enabled	
Node auto-provisioning (Autopilot mode)	Enabled	
Auto-provisioning network tags		
Autoscaling profile	Optimize utilization	

### 3. Red VPC

Google Cloud

fleetproject

VPC Network / VPC networks / Network: default

VPC networks

IP addresses

Internal ranges

Bring your own IP

Firewall

Routes

VPC network peering

Shared VPC

Serverless VPC access

Packet mirroring

VPC Flow Logs

VPC network details

DELETE VPC NET

default

OVERVIEWSUBNETSSTATIC INTERNAL IP AD

EDIT

Description

Default network for the project

Maximum transmission unit

1460

VPC network ULA internal IPv6 range

Disabled

Subnet creation mode

Auto subnets

Dynamic routing mode

Regional

Best path selection mode

Legacy

Tags

EQUIVALENT REST

## 4. Load Balancer

Google Cloud

fleetproject

load balancer

Search

Network Services

Load balancing

Cloud DNS

Cloud CDN

Cloud NAT

Cloud Service Mesh (Traff...

Service Directory

Load balancing

+ CREATE LOAD BALANCER

REFRESH

DELETE

LOAD BALANCERSBACKENDSFRONTENDSSERVICE LB POLICIES

Filter Enter property name or value

Name	Load balancer type	Access type	Protocols	Region	Backends
a85dced360e324d578cfd6057c08de9a	Network (Passthrough target-pool)	External	TCP	us-central1	1 target pool (4 instances)
a8d259be708f74ae0980e8ca9f523c73	Network (Passthrough target-pool)	External	TCP	us-central1	1 target pool (4 instances)

To view or delete load balancing resources like forwarding rules and target proxies, go to

## 5. Artifact Registry

Artifact Registry

Repositories

Settings

← Images for blog

DELETE

EDIT REPOSITORY

SETUP INSTRUCTIONS

us-central1-docker.pkg.dev > fleetproject-403015 > blog

Repository Details



Format Docker

Type Standard

SHOW MORE

Filter

Enter property name or value

<input type="checkbox"/>	Name ↑	Connection	Created	Updated
<input type="checkbox"/>	 <a href="#">backend</a>	—	8 days ago	4 hours ago
<input type="checkbox"/>	 <a href="#">frontend</a>	—	8 days ago	1 hour ago

## Configuración del API (Backend)

El API Backend se encuentra publicada en <http://34.134.153.141/docs>

Blog de Seguridad de Nicolás

backend seguridad blog - Swi...

blog db - SQL - fleepproject

Images for blog - Anfract Reg

34.134.153.141/docs/

backend seguridad blog 1.0 OAS 3.1

OpenAPI JSON

API con autenticación en FastAPI

Authorize

Posts

GET

/posts/

Get Posts

POST

/posts/create

Create Post

GET

/posts/{post\_id}

Get Posts Detail

DELETE

/posts/posts/{post\_id}

Delete Post

PUT

/posts/postsupdate/{post\_id}

Update Post

GET

/posts/posts-by-tag/{tag\_id}

Get Posts By Tag

Ratings

GET

/ratings/posts/average

Get Posts With Average Ratings

GET

/ratings/{post\_id}/average

Get Average Rating

POST

/ratings/

Rate Post

Tags

POST

/tags/createtag

Create Tag

GET

/tags/gettags

Get Tags

Tags

POST

/tags/createtag

Create Tag

GET

/tags/gettags

Get Tags

Usuarios

POST

/usuarios/registran

Register User

POST

/usuarios/login

Login User

default

GET

/

Read Root

GET

/luisa

Read Root

Schemas

HTTPValidationError

Expand all

object

PostCreate

Expand all

object

PostUpdate

Expand all

object

RatingCreate

Expand all

object

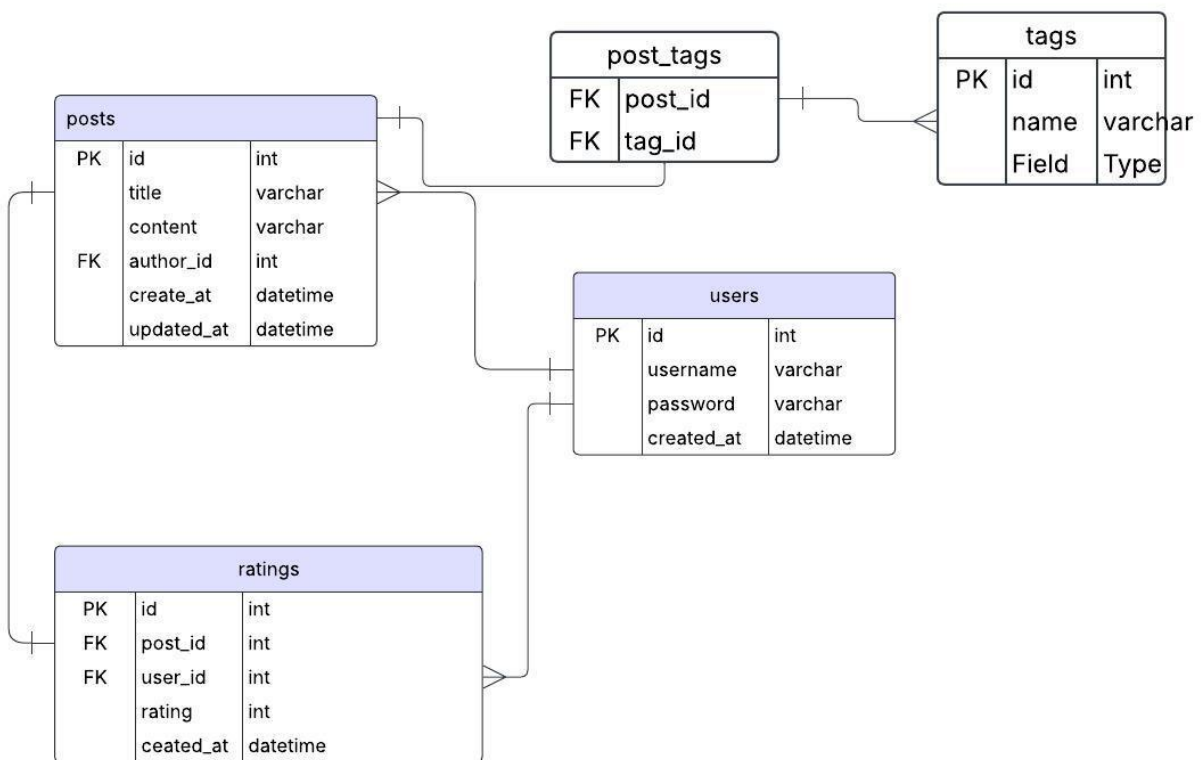
11:13 PM

9/2/2025

11:14 PM

3/2/2025

## Esquema de la base de datos



## Decisiones de diseño y seguridad

- Para el backend, como fue sugerido, se selecciona trabajar con FastAPI, un framework de python que facilita el desarrollo de apis.
  - Cross-Site Request Forgery: Fast API permite validar que solo las ips o dominios autorizados accedan al API, evitando este ataque
  - SQL Injection: Se usa SQLAlchemy, una libreria para modelar e interactuar con la base de datos, que tiene sus propias sentencias y modelos para interactuar con la base de datos, garantizando que no va a haber SQL Injection
  - Cross-Site Scripting: Se utiliza esquemas para garantizar la estructura y el formato de los datos y con la libreria Pydantic y HTML.Scape, se evita que se inserte código o caracteres maliciosos
- Para el frontend se escogió trabajar con React, un framework de frontend robusto y fácil de usar

- Para la conexión a la base de datos desde GKE, se instaló un contenedor sidecard en el pod de backend, para lograr la conexión

```
containers:
- name: fastapi
  image: us-central1-docker.pkg.dev/fleetproject-403015/blog/backend:latest
  ports:
  - containerPort: 8080
  resources:
    requests:
      cpu: "250m" # El pod solicita al menos 250 milicores de CPU
    limits:
      cpu: "500m" # Máximo 500 milicores de CPU por pod
- name: cloudsql-proxy
  image: gcr.io/cloud-sql-connectors/cloud-sql-proxy:latest
  args:
  - "--auto-iam-authn"
  - "--port=5432"
  - "fleetproject-403015:us-central1:blog-db"
  securityContext:
    runAsNonRoot: true
```

## Despliegue

1. Construir imágenes: Se usan los Dockerfile para construir las imágenes en el frontend y el backend. En nuestro caso los comandos usados son:

```
gcloud builds submit --tag us-central1-docker.pkg.dev/fleetproject-403015/blog/backend:latest
```

```
gcloud builds submit --tag us-central1-docker.pkg.dev/fleetproject-403015/blog/frontend:latest
```

2. Se despliega en GKE, para eso se utiliza el archivo de deployment.yaml en el backend y el frontend respectivamente

```
Kubectl apply -f deployment.yaml
```

Podemos ver la aplicación desplegada

```
(venv) luisaq@luisawork:~/Documents/blog/CloudBlog/frontend$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
blog-backend        2/2     2             2           6h6m
frontend            2/2     2             2           3h16m

(venv) luisaq@luisawork:~/Documents/blog/CloudBlog/frontend$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
blog-backend-7d85c6b79c-8kr2d       2/2     Running   0           4h42m
blog-backend-7d85c6b79c-vr49d       2/2     Running   0           4h43m
frontend-57cc746776-9qmv7           1/1     Running   0           81m
frontend-57cc746776-wkq9g           1/1     Running   0           83m

(venv) luisaq@luisawork:~/Documents/blog/CloudBlog/frontend$ kubectl get services
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
backend-service     LoadBalancer  34.118.226.189   34.134.153.141   80:30938/TCP     5h49m
front-service       LoadBalancer  34.118.239.87    35.192.177.211   8080:31011/TCP   8d
kubernetes           ClusterIP      34.118.224.1     <none>           443/TCP          8d
```

3. Se debe crear una cuenta de servicio de GCP con permisos de administración sobre la base de datos.

```
cloud-sql-proxy@fleetproject-403015.iam.gserviceaccount.com Cloud SQL Proxy Cloud SQL Admin
```

- Se le debe dar permisos de Cloud SQL admin
- Se debe asociar a una cuenta de servicio de GKE

```
gcloud iam service-accounts add-iam-policy-binding \ gke-pod-service-
account@TU_PROYECTO.iam.gserviceaccount.com \ --
role=roles/iam.workloadIdentityUser \ --
member="serviceAccount:TU_PROYECTO.svc.id.goog[default/gke-pod-sa]"
```

- En el deployment se debe especificar

```
spec:
  serviceAccountName: ksa-cloud-sql
  containers:
```

## Accesos

Frontend



<http://35.192.177.211:8080/>

Backend

<http://34.134.153.141/docs#/>

Github

<https://github.com/luferquisa/CloudBlog>

Arquitectura

[https://github.com/luferquisa/CloudBlog/blob/main/Arquitectura%20blog%20BitSeguro%20\(1\).png](https://github.com/luferquisa/CloudBlog/blob/main/Arquitectura%20blog%20BitSeguro%20(1).png)