

Universidad Rafael Landívar

Facultad de Ingeniería

Virtualización

Ing. Samayoa

PROYECTO VIRTUALIZACIÓN

Carlos Andrés Vargas Orué 1125417

José Daniel De León Chang 1170419

Ximena Elizardi 1101720

Guatemala 23 de mayo de 2024

Base de datos

-Creamos una instancia con firebase, para poder almacenar nuestros registros y así mismo crear logs para su utilización.

Creación imágenes docker

-Creación frontend con nginx

Se creó un archivo .html para la interacción con el API, tiene esta forma

Login ADMINISTRATIVO

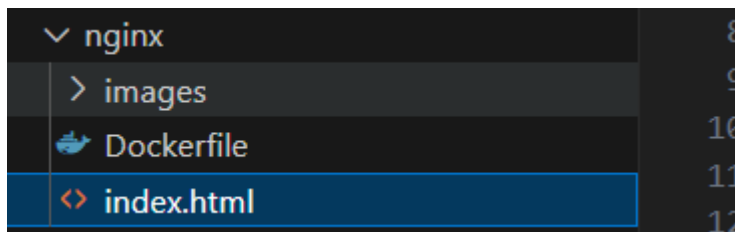
Login Tico

Login Chang

Login Ximena

Consumo de Endpoint

Su carpeta se ve en esta forma conteniendo la página (el js para su funcionamiento) y una carpeta de imágenes para dejarlo más llamativo.



Para la creación de su imagen creamos su respectivo Dockerfile

```

nginx > Dockerfile
1  # Usar la imagen oficial de NGINX como base
2  FROM nginx:latest
3
4  # Copiar el archivo HTML principal al directorio de contenido estático de NGINX
5  COPY index.html /usr/share/nginx/html
6
7  # Copiar recursos estáticos adicionales, si es necesario
8  COPY images /usr/share/nginx/html/images
9
10 # Copiar archivo de configuración personalizado de NGINX, si es necesario
11 #COPY nginx.conf /etc/nginx/conf.d/
12
13 # Exponer el puerto 80 para que NGINX sea accesible desde fuera del contenedor
14 EXPOSE 80
15

```

```

<!-- index.html -->

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Consumo de Endpoint</title>
  <!-- Agregar Bootstrap CDN para estilos -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhJY6hW+ALEwIH" crossorigin="anonymous">
</head>

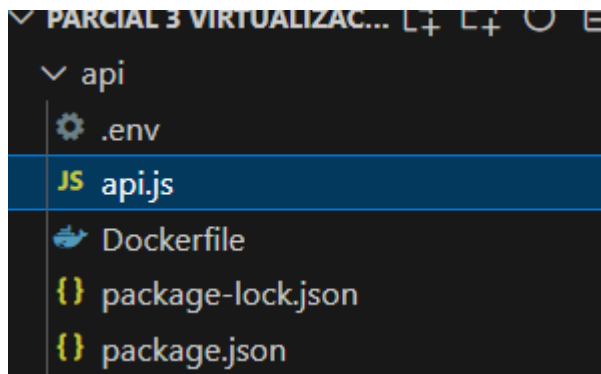
<body>
  <div class="container mt-5">
    <h1 class="mb-4">Login ADMINISTRATIVO</h1>
    <div class="row">
      <div class="col">
        <button id="fetchDataTico" class="btn btn-primary">Login Tico</button>
        <div id="imageTico" class="mt-3"></div>
      </div>
      <div class="col">
        <button id="fetchDataChang" class="btn btn-primary">Login Chang</button>
        <div id="imageChang" class="mt-3"></div>
      </div>
      <div class="col">
        <button id="fetchDataXimena" class="btn btn-primary">Login Ximena</button>
        <div id="imageXimena" class="mt-3"></div>
      </div>
    </div>
    <div class="row mt-3">
      <div class="col">
        <p id="resultTico"></p>
      </div>
      <div class="col">
        <p id="resultChang"></p>
      </div>
    </div>
  </div>
</body>
</html>

```

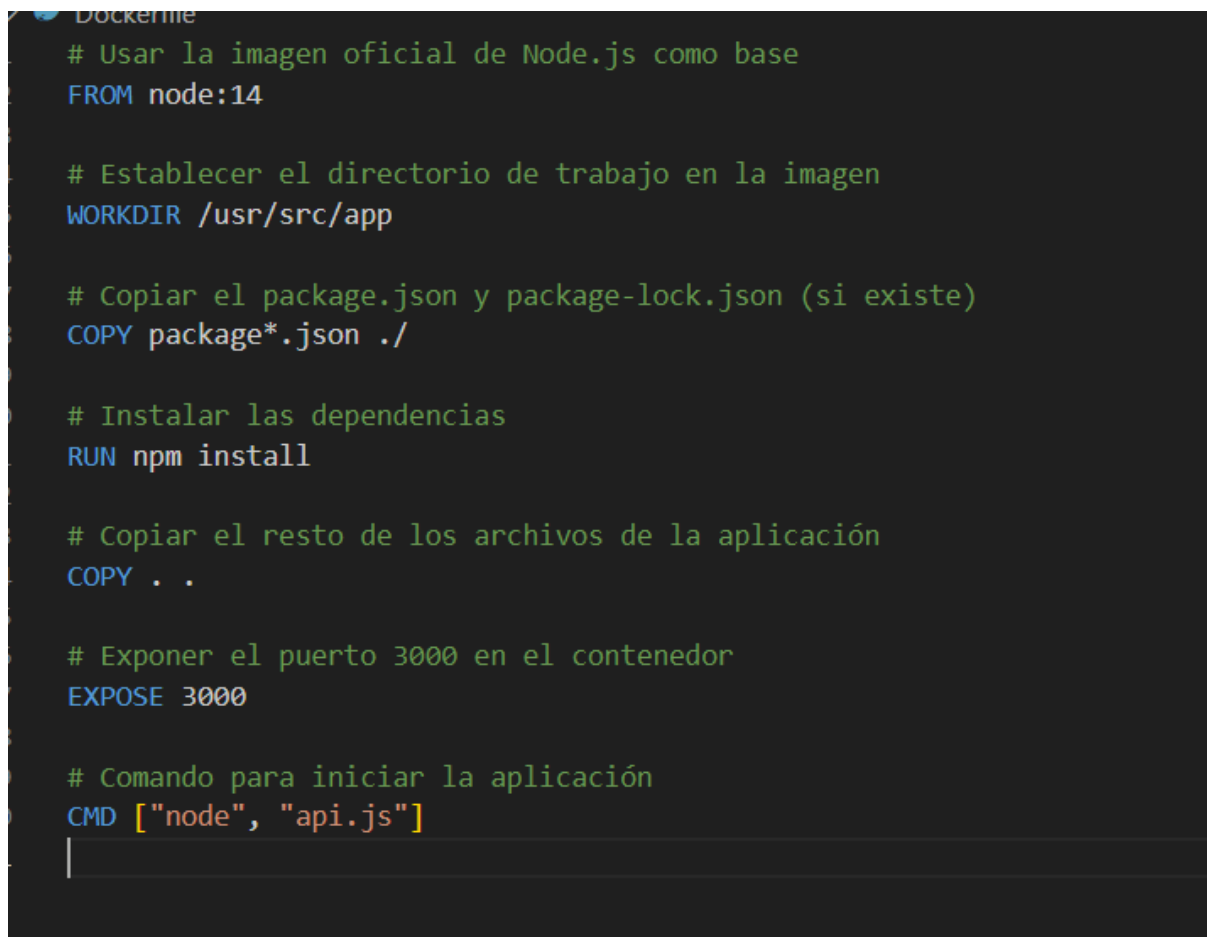
Se hizo uso de comando de docker “docker build -t nginx . ”

-Creación backend con node

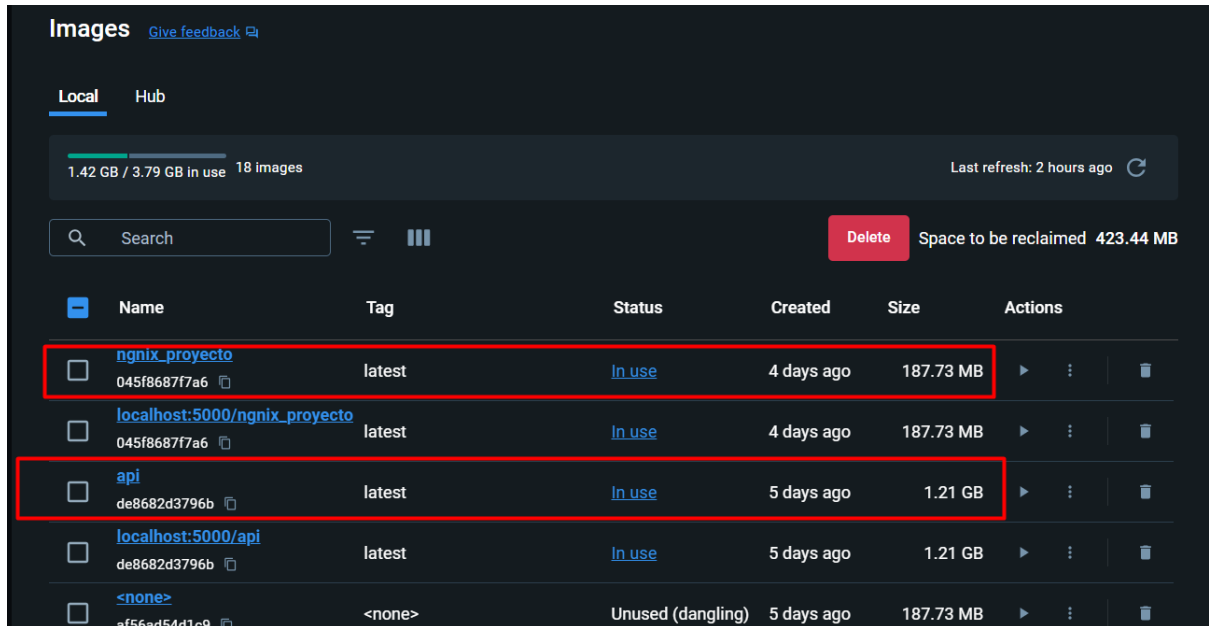
Asi se ve su caperta donde creamos un archivo .js y los .json para utilizar paquetes.



Asi mismo su creación de imagen con el archivo Dockerfile



Visualización imágenes y explicación



The screenshot shows the Docker Images management interface. At the top, it indicates '1.42 GB / 3.79 GB in use' and '18 images'. A search bar and a 'Delete' button are visible. The table below lists the images with columns for Name, Tag, Status, Created, Size, and Actions. Three images are highlighted with red boxes:

	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	nginx_proyecto 045f8687f7a6	latest	In use	4 days ago	187.73 MB	▶ ⋮ 🗑
<input type="checkbox"/>	localhost:5000/nginx_proyecto 045f8687f7a6	latest	In use	4 days ago	187.73 MB	▶ ⋮ 🗑
<input type="checkbox"/>	api de8682d3796b	latest	In use	5 days ago	1.21 GB	▶ ⋮ 🗑
<input type="checkbox"/>	localhost:5000/api de8682d3796b	latest	In use	5 days ago	1.21 GB	▶ ⋮ 🗑
<input type="checkbox"/>	<none> af56ad54d1c9	<none>	Unused (dangling)	5 days ago	187.73 MB	▶ ⋮ 🗑

-nginx_proyecto: este levanta el frontend, haciendo la petición a nuestro api.

-api: este levanta la api con node.js la cual hace una conexión hacia firebase para guardar registros y también poder hacer uso de logs.

Creación terraform

Para el uso de terraform se creo un archivo llamado main.tf (entiendo terraform ya instalado asi mismo creada su variable de entorno)

Lo que contiene este es las configuraciones necesarios para poder levantar nuestros sistemas con terraform y asi mismo incluir el apartado de kubernetes.

```
terraform{  
  required_providers{  
    kubernetes = {  
      source = "hashicorp/kubernetes"  
      version = "~> 2.0"  
    }  
  }  
}
```

```
provider "kubernetes" {  
  config_path  = "~/.kube/config"  
  config_context = "docker-desktop"  
}
```

Deployment para la API

```
resource "kubernetes_deployment" "api" {  
  metadata {  
    name = "api-deployment"  
  }  
  
  spec {
```

```
replicas = 1

selector{
  match_labels = {
    app = "api"
  }
}

template {
  metadata {
    labels = {
      app = "api"
    }
  }
}

spec {
  container {
    image = "localhost:5000/api:latest"
    name = "api"
    port {
      container_port = 3000
    }
  }
}
}
```

Service para la API

```
resource "kubernetes_service" "api" {
```

```
  metadata {
```

```
    name = "api-service"
```

```
  }
```

```
  spec {
```

```
    selector = {
```

```
      app = "api"
```

```
    }
```

```
    port {
```

```
      port      = 3000
```

```
      target_port = 3000
```

```
    }
```

```
    type = "LoadBalancer"
```

```
  }
```

```
}
```

```
# Deployment para Nginx
```

```
resource "kubernetes_deployment" "nginx" {
```

```
  metadata {
```

```
    name = "nginx-deployment"
```

```
  }
```

```
  spec {
```

```
    replicas = 2
```

```
    selector {
```



```
match_labels = {  
  app = "nginx"  
}  
}
```

```
template {  
  metadata {  
    labels = {  
      app = "nginx"  
    }  
  }  
}
```

```
spec {  
  container {  
    image = "localhost:5000/nginx_proyecto:latest"  
    name = "nginx"  
    port {  
      container_port = 80  
    }  
  }  
}  
}
```

```
# Service para Nginx  
resource "kubernetes_service" "nginx" {  
  metadata {
```

```
    name = "nginx-service"
  }

  spec {
    selector = {
      app = "nginx"
    }

    port {
      port      = 80
      target_port = 80
    }

    type = "LoadBalancer"
  }
}
```

Explicacion

Indicarle que vamos a trabajar con kubernetes

```
terraform {  
  required_providers {  
    kubernetes = {  
      source  = "hashicorp/kubernetes"  
      version = "~> 2.0"  
    }  
  }  
}
```

Las configuraciones con las que se usaran kubernetes (en este caso las default de docker desktop)

```
provider "kubernetes" {  
  config_path    = "~/.kube/config"  
  config_context = "docker-desktop"  
}
```

En este deployment se realizan las configuraciones necesarias y asi mismo obtener la imagen creada anteriormente con el Dockerfile del api para poder ser usada.

```

# Deployment para la API
resource "kubernetes_deployment" "api" {
  metadata {
    name = "api-deployment"
  }

  spec {
    replicas = 1
    selector {
      match_labels = {
        app = "api"
      }
    }

    template {
      metadata {
        labels = {
          app = "api"
        }
      }

      spec {
        container {
          image = "localhost:5000/api:latest"
          name  = "api"
          port {
            container_port = 3000
          }
        }
      }
    }
  }
}

```

El service api para que este siempre este arriba

```
# Service para la API
resource "kubernetes_service" "api" {
  metadata {
    name = "api-service"
  }

  spec {
    selector = {
      app = "api"
    }

    port {
      port          = 3000
      target_port = 3000
    }

    type = "LoadBalancer"
  }
}
```

El deployment de nginx

```
# Deployment para Nginx
resource "kubernetes_deployment" "nginx" {
  metadata {
    name = "nginx-deployment"
  }

  spec {
    replicas = 2
    selector {
      match_labels = {
        app = "nginx"
      }
    }

    template {
      metadata {
        labels = {
          app = "nginx"
        }
      }

      spec {
        container {
          image = "localhost:5000/nginx_proyecto:latest"
          name  = "nginx"
          port {
            container_port = 80
          }
        }
      }
    }
  }
}
```

```
3 # Service para Nginx
4 resource "kubernetes_service" "nginx" {
5   metadata {
6     name = "nginx-service"
7   }
8
9   spec {
10    selector = {
11      app = "nginx"
12    }
13
14    port {
15      port          = 80
16      target_port = 80
17    }
18
19    type = "LoadBalancer"
20  }
21 }
```

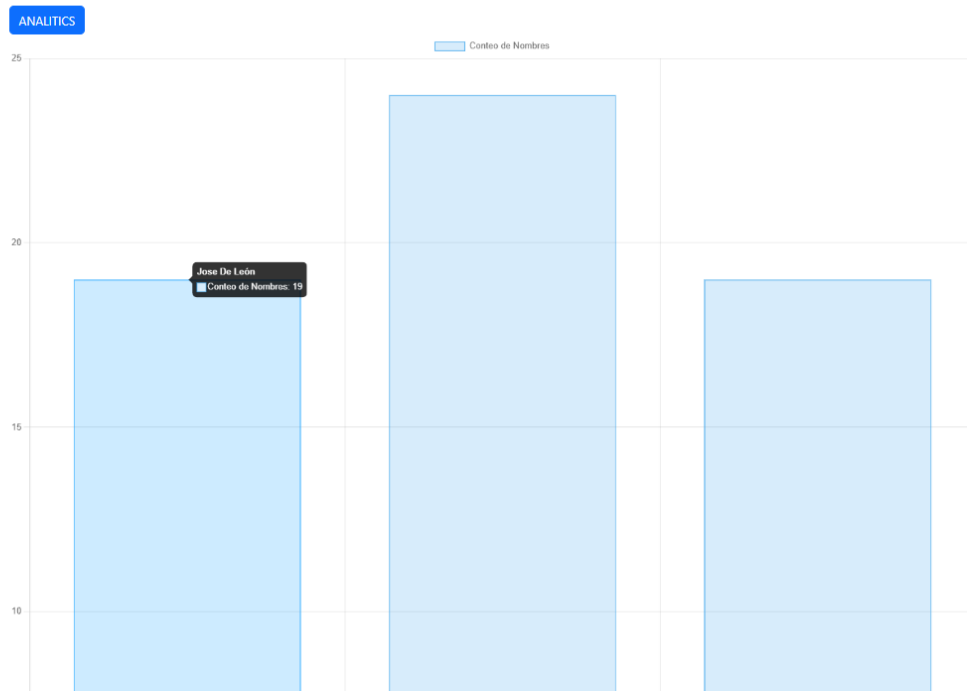
Mismo funcionamiento que api pero con imagen de nginx

Interpretacion de LOGS

En este proyecto los manejamos de 2 maneras

1 donde hacemos uso de los logs de Firebase en el cual nos devuelve cuantas veces ingresamos al sistema o cada usuario al sistema

Consumo de Endpoint



2 docker stats

Docker nos brinda la información de consumo de cada uno de los contenedores que se estan ejecutando.



Anexo:

Link de proyecto

<https://github.com/ZeroJChang/ProyectoVirtualizacion>

Link video youtube:

<https://youtu.be/xJTZAYDlEsc>