Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Santa Fe

Ingeniería en Tecnologías Computacionales

Proyecto de Fin de Carrera

Assessment

Documentación del Sistema

Alumnas | Matrícula:

Samantha Covarrubias Figueroa | A01026174

Cristina Alessandra González | A01025667

Valeria Martínez Martínez | A01782413

Andrea San Martín Vital | A00833676

Fecha de entrega

9 de junio de 2025

1. Identificación del Documento

1.1 Fecha y versión

Última actualización	Última versión
09/06/2025	2.1.6

1.2 Autoras del proyecto

- Samantha Covarrubias Figueroa - Desarrolladora
- Cristina Alessandra González - Desarrolladora
- Valeria Martínez Martínez - Desarrolladora
- Andrea San Martín Vital - Desarrolladora

2. Descripción General del Sistema

2.1 Resumen ejecutivo

Página web en la cual se pueden obtener recomendaciones de cultivos dependiendo de las condiciones climatológicas utilizando una inteligencia artificial. Aplicación instanciada en la nube privada con Load Balancer para el control de carga del sistema en el backend.

2.2 Arquitectura general

- Frontend: Se utilizó next js para la elaboración de la página web, en la cual se elaboraron APIs para el enrutamiento y conexión de la base de datos e inteligencia artificial con la experiencia del usuario en la página web de uso final.
- Backend: Se utilizó Flask API para poder conectar el front y poder utilizar el modelo AI. Se utilizó nginx para elaborar el load balancer el cual está conectado tanto con el frontend en next js y el backend en Flask API. El balanceador de carga en nginx es al que se le dio acceso a internet ya que este es el que controla el direccionamiento de la página web y sirve como protección a la API y la base de datos. Posteriormente se utilizaron dockers para poder correr los servicios de la API y el Load Balancer.
- Inteligencia Artificial: entrenada de manera supervisada con el uso de una red neuronal, funciones ReLu y Softmax de activación, función ADAM de optimización y cross-entropy para la obtención del error en el backpropagation. Se utilizó el formato pkl para poder guardar el modelo entrenado posteriormente con la API.
- Base de datos: se utilizó una base de datos relacional para poder guardar la información de los usuarios lo que nos permite hacer la autenticación y dar acceso a la

página web a los usuarios, y también se guardó la información de los cultivos con los que contaba el modelo. Se hizo por medio de mySQL para poder acceder y hacer búsquedas en las tablas de la base de datos.

2.3 Tecnologías utilizadas

- Frontend: Next .js, Docker, Tailwind
- Backend: Flask, python, Docker, CORS, pickle, jwt
- Inteligencia Artificial: PyTorch, Python, Scikit-learn
- Base de datos: mySQL

2.4 Relaciones entre componentes

El frontend tiene comunicación con la API del backend por medio de comandos use effect, use State y use Ref de react para poder hacer las llamadas y el fetch de la IA, y de next server para poder hacer los post a las bases de datos. La API utiliza los archivos pkl y flask para poder tener el modelo de IA entrenado y para mandar el resultado, utiliza jwt para poder encriptar las contraseñas de los usuarios y hacer la autenticación, y mysql connector para poder hacer la conexión con la base de datos. Se utilizó nginx para poder hacer el load balancer con el algoritmo de round robin utilizando un upstream de las direcciones ip en donde se encontraban las instancias del backend y una configuración de server en donde se direccionan las rutas hacia las instancias de backend y hacia las páginas del frontend.

3. Vista de Interfaces (Interface View)

3.1 APIs REST

- Backend Endpoints
    - Autenticación
        - POST: ‘/api/auth/login’
        - Descripción: Autenticación de usuarios en el sistema
        - Content-Type: ‘application/json’
    - Predicción de cultivos
        - POST: ‘/predict-simple’
        - Descripción: Predice el cultivo recomendado
        - Content-Type: ‘application/json’
        - POST: ‘/predict-crop’
        - Descripción: Predicción con múltiples recomendaciones
        - Content-Type: ‘application/json’
    - Gestión de datos

        - GET: ‘/crops/’
        - Descripción: Obtiene la lista de todos los cultivos disponibles
        - GET: ‘/health’
        - Descripción: Verificar la salud del sistema
    - Configuración del servidor
        - HOST: ‘0.0.0.0’ (todas las interfaces)
        - Puerto: ‘5001’
- Frontend Endpoints
    - Comunicación con Backend
        - Base URL: ‘http://localhost:5001’
        - Origin: Frontend
        - Methods: `GET, POST, OPTIONS`
        - Headers: `Content-Type, Authorization`
    - Login
        - Componente: ‘LoginForm’
        - Ruta: ‘/login’
        - Elementos:
            - Usuario
            - Contraseña
            - Botón de iniciar sesión
            - Validación de campos
        - Estados:
            - Loading durante autenticación
            - Error en credenciales inválidas
            - Redirección tras login exitoso
    - Dashboard principal
        - Componente: Home
        - Ruta: ‘/home’
        - Elementos:
            - Navegación principal
            - Botón para iniciar sesión
            - Botón para registrarse
    - Predicciones
        - Componente: Predict
        - Ruta: ‘/predict’
        - Elementos:
            - Nitrógeno (N): `input[type="number"]`
            - Fósforo (P): `input[type="number"]`
            - Potasio (K): `input[type="number"]`
            - Temperatura: `input[type="number"]` (°C)

- Humedad: `input[type="number"]` (%)
- pH del suelo: `input[type="number"]` (0-14)
- Precipitación: `input[type="number"]` (mm)
- Botón para realizar la predicción
- Resultado de predicción

■ Validaciones:
- Campos requeridos
- Rangos válidos por campo
- Formato numérico

4. Vista de Información (Information View)

4.1 Diseño de base de datos

a. Modelo Entidad-Relación:

- Entidades principales: users contiene información de usuario y autenticación; crops es el catálogo de cultivos reconocidos por el sistema; user_climate_data: datos que determinan el cultivo, input de la IA; crop_predictions: predicción de la IA, asociada a los datos (user_climate_data) y usuarios (users).
- Relaciones: Un usuario tiene muchos registros de datos climáticos; cada registro de datos climáticos pertenece a un usuario; cada predicción está asociada a un registro de datos climáticos, a un usuario y a un cultivo predicho; los cultivos son referenciados tanto en los datos de entrenamiento como en las predicciones.

b. Descripción de Tablas

- users

Campo	Tipo	Restricciones	Descripción
user_id	INT	PK, AUTO_INCREMENT	Identificador único de usuario
username	VARCHAR(50)	NOT NULL, UNIQUE	Nombre de usuario
email	VARCHAR(100)	NOT NULL, UNIQUE	Correo electrónico
password_hash	VARCHAR(255)	NOT NULL	Contraseña en hash
full_name	VARCHAR(100)		Nombre completo
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Fecha de creación

- crops

Campo	Tipo	Restricciones	Descripción
crop_id	INT	PK, AUTO_INCREMENT	Identificador único de cultivo
crop_name	VARCHAR(50)	NOT NULL, UNIQUE	Nombre común de cultivo
crop_label	INT	NOT NULL, UNIQUE	Etiqueta usada por el modelo de IA
scientific_name	VARCHAR(100)		Nombre científico
description	TEXT		Descripción
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Fecha de alta

- user_climate_data

Campo	Tipo	Restricciones	Descripción
data_id	INT	PK, AUTO_INCREMENT	Identificador único
user_id	INT	FK (users)	Usuario dueño del registro
nitrogen	DECIMAL(8,2)	NOT NULL	Nitrógeno
phosphorus	DECIMAL(8,2)	NOT NULL	Fósforo
potassium	DECIMAL(8,2)	NOT NULL	Potasio
temperature	DECIMAL(5,2)	NOT NULL	Temperatura en °C
humidity	DECIMAL(5,2)	NOT NULL	Humedad %
ph_level	DECIMAL(4,2)	NOT NULL	pH del suelo
rainfall	DECIMAL(8,2)	NOT NULL	Precipitación mm
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Fecha de registro

- crop_predictions

Campo	Tipo	Restricciones	Descripción
prediction_id	INT	PK, AUTO_INCREMENT	Identificador de predicción
data_id	INT	FK (user_climate_data), NOT NULL, UNIQUE	Datos usados
user_id	INT	FK (users), NOT NULL	Usuario que realiza la consulta
predicted_crop_id	INT	FK (crops), NOT NULL	Cultivo predicho
confidence_score	DECIMAL(5,4)	NOT NULL	Confianza del modelo (0-1)
model_architecture	VARCHAR(20)	DEFAULT 'deep'	Arquitectura utilizada

created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Fecha de predicción

c. Vistas Definidas
    - v_prediction_details: Une las predicciones con los datos de usuario y detalles de cultivo.
    - v_user_stats: Estadísticas básicas por usuario, como total de predicciones, confianza promedio y cultivo más frecuente.

d. Normalización

El diseño de esta base de datos, cumple con la cuarta forma normal (4FN) ya que cumple con los siguientes requisitos: Todos los atributos son atómicos y no repetitivos (1FN). Todas las tablas tienen PK y los campos dependen completamente de la PK (2FN). No hay dependencias transitivas; toda la información depende únicamente de la PK de cada tabla (3FN). No existen dependencias multivaluadas no triviales distintas de la dependencia funcional de la PK (4FN).

4.2 Justificación

Consideramos que la base de datos es adecuada para las necesidades de nuestro proyecto ya que es relacional, lo cual nos garantiza que exista consistencia y un modelo fijo que mantiene coherencia y organización para la recopilación de datos en el sistema. Además el que cumpla con la normalización a la 4ta forma nos asegura que se están implementado efectivamente las relaciones entre las diferentes entidades. Asimismo, las entidades escogidas permiten que se puedan agregar más variables climáticas o nuevos cultivos sin alterar la estructura básica.

5. Decisiones de Diseño

5.1 Justificación de tecnologías

- Next.js: se utilizó next js por la facilidad de la programación, uso de componentes y del server side para poder conectarlo con los demás componentes.
- Tailwind: se utilizó este framework para facilitar el diseño, personalización y la codificación de las interfaces de la página web.
- Flask: framework que permite trabajar con python para poder poder hacer APIs RESTful de manera más fácil y poder integrar los archivos pickle de la IA
- CORS: Permite la comunicación entre el frontend que esta en next.js y el backend que

está hecho en python y flask.

- Pickle: permite hacer la serialización de los modelos de inteligencia artificial para poder utilizar en otros framework como flask en este caso.
- Jwt: se usó para hacer la autenticación de los usuarios de forma segura ya que permite la validación de las sesiones.
- PyTorch: librerías para el desarrollo de modelos de inteligencia artificial de forma más fácil y con mayor accesibilidad.
- Python: Lenguaje de programación de forma sencilla para poder generar los programas e importar librerías más fácil.
- Scikit-learn:Cuenta con herramientas de preprocesamiento, métricas y modelos de aprendizaje que facilitaron el desarrollo del modelo de inteligencia artificial, el entrenamiento del mismo y su compatibilidad con PyTorch.
- MySQL: sistema de gestión de base de datos relacionales que facilita la búsqueda, el ordenamiento y la integridad de los datos.
- Docker: con esto se pudieron hacer contenedores para permitir la ejecución de los archivos y servicios del backend y del load balancer, lo que permitió su construcción y el despliegue de los mismo en las instancias de la nube privada y facilitando su comunicación.

5.2 Trade-offs principales

- Algoritmo de round-robin para el load balancer: se decidió utilizar el algoritmo de round-robin para poder hacer el balanceo de cargas debido a que se puede hacer una distribución uniforme entre los algoritmos de backend, se puede aplicar de manera más simple y con un upstream de direcciones ip más sencillas.

6. Vista Algorítmica

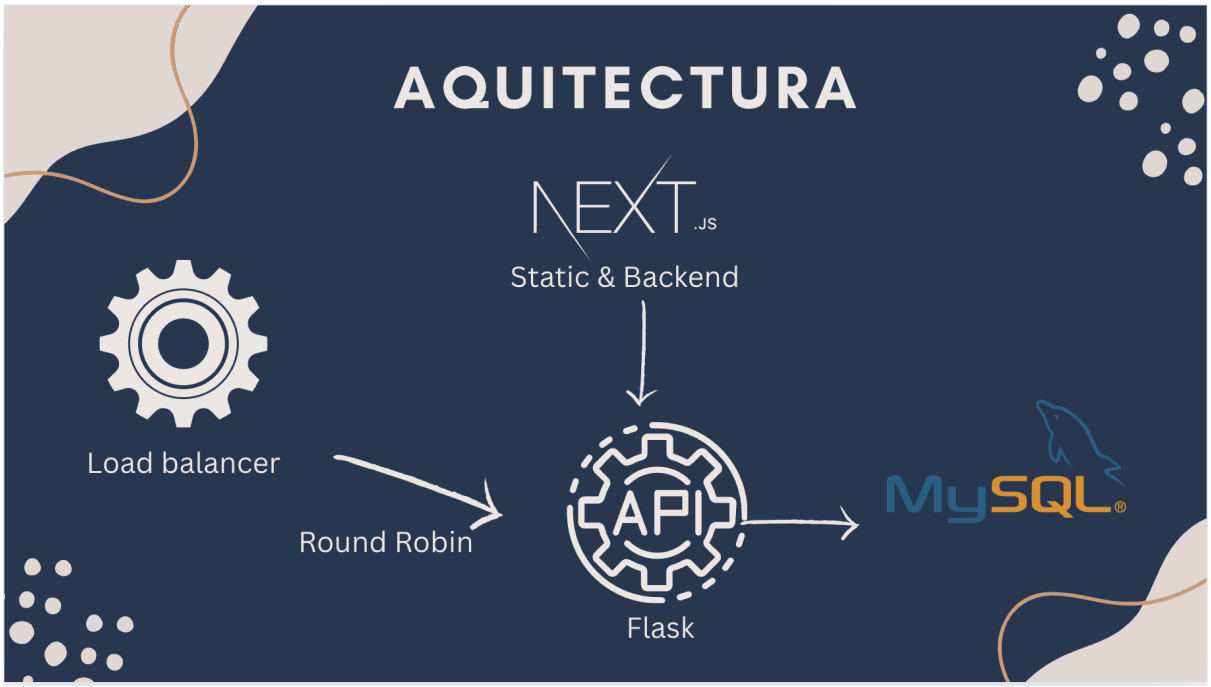
El modelo de inteligencia artificial para la recomendación de cultivos con base en las condiciones climatológicas del usuario se hizo con redes neuronales multi clase de forma supervisada con valores de verificación de los parámetros ingresados para poder tener un mayor control y certeza del modelo. Se utilizaron 3 capas ocultas para el entrenamiento, con la primera capa de entrada con 7 neuronas que son las condiciones climatológicas y la salida de 22 los cuales son los tipos de cultivo. Se utilizó una base de datos con alrededor de 22000 datos, los cuales se dividieron en un 70% se utilizaron para el entrenamiento del modelo, 15% para la validación y 15% para las pruebas. Como función de activación se utilizó ReLu para poder normalizar los valores e introducir no linealidad al sistema, y en la última capa se

utilizó la función de softmax para poder convertir las salidas del modelo en probabilidades de cada cultivo. Utilizamos la función de optimización de ADAM ya que cuenta con el momentum que nos permite salir de mínimos locales e intentar llegar al mínimo global lo que nos da una mayor accuracy en el modelo y RMS Prop que nos permite alterar el aprendizaje haciéndolo más eficiente. Utilizamos cross-entropy para poder calcular el error y realizar la backpropagation la cual nos permite actualizar los parámetros de peso y bias por época y bash. Se utilizó el early stopping para evitar el sobre-entrenamiento del modelo y ahorrar recursos del sistema cuando la variabilidad de los resultados por época alcanzaba la paciencia indicada. Con esto pudimos obtener un modelo de recomendación con certezas entre un 90-99% para nuestra página web.

7. Apéndices

A. Diagramas: Arquitectura del sistema, flujo de datos

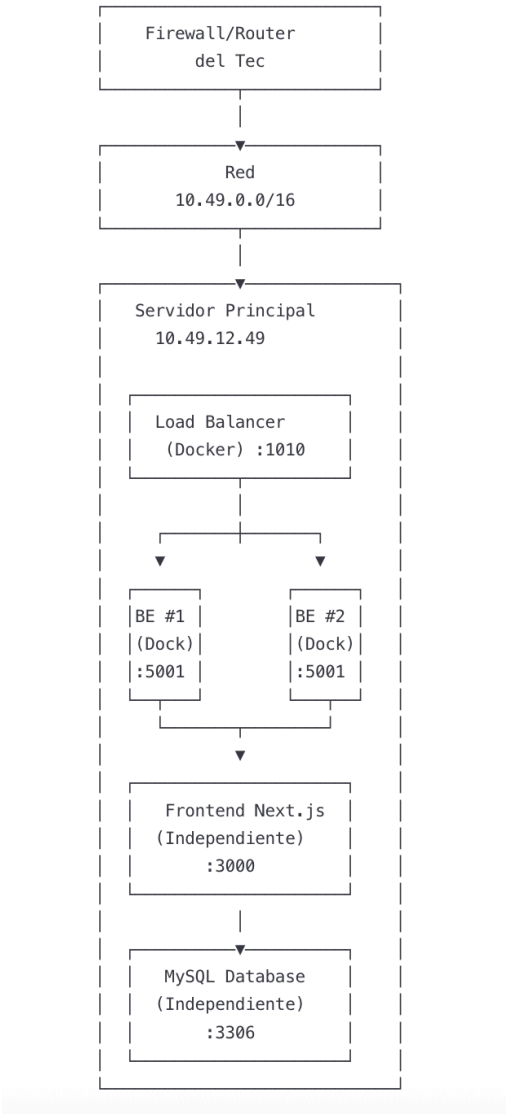
I. Arquitectura general



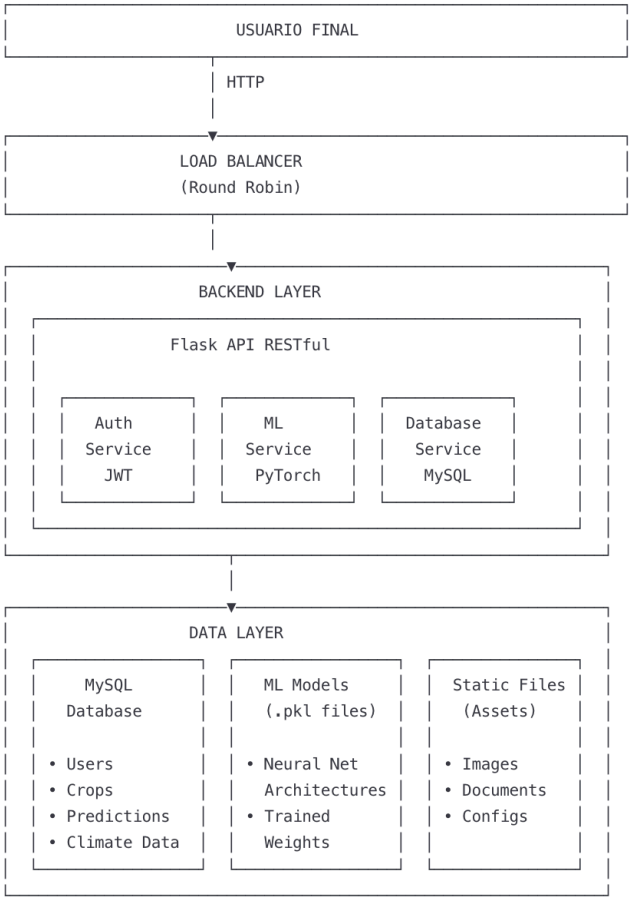
II. Topología de red

Para conectarnos a la nube privada del Tec usamos una computadora (cómputo) que contiene nuestro backend y el modelo, ésta se conecta a la nube privada por medio de un switch conectado a un router que sirve de gateway para dar salida a ésta.

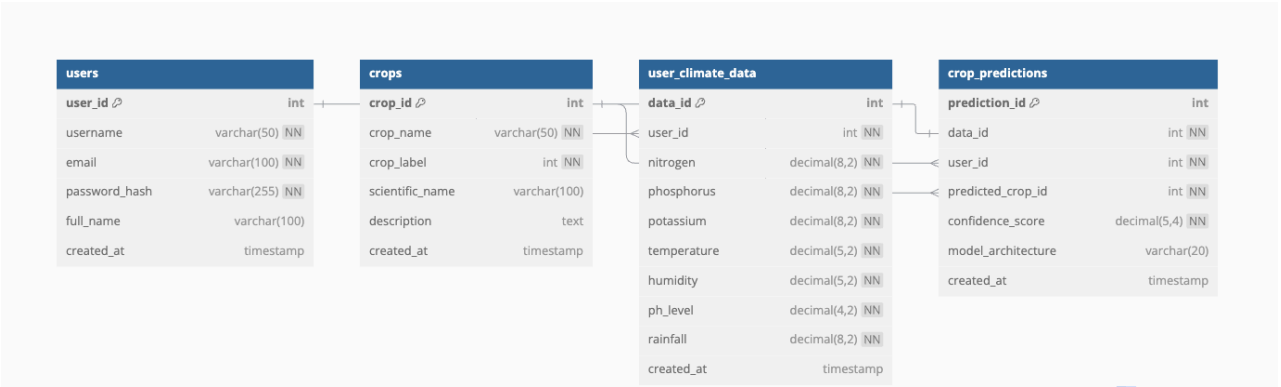
Del otro lado de la nube, la topología está puesta de la siguiente forma:



III. Arquitectura de Alto Nivel



B. Esquemas de base de datos: Diagrama ER



C. Ejemplos de API: Requests/responses típicos

I. Predicción

```
❯ curl -X POST http://localhost:5001/predict-simple \
  -H "Content-Type: application/json" \
  -d '{"N": 90, "P": 42, "K": 43, "temperature": 20.9, "humidity": 82.0, "ph": 6.5, "rainfall":
202.9}'
{
  "confidence": 0.9999936819076538,
  "crop_label": 15,
  "input_features": [
    90.0,
    42.0,
    43.0,
    20.9,
    82.0,
    6.5,
    202.9
  ],
  "predicted_crop": "muskmelon",
  "success": true
}
```

II. Verificar salud

```
❯ curl http://localhost:5001/health
{
  "crops_available": 22,
  "database": "connected",
  "model": "loaded",
  "status": "healthy",
  "timestamp": "2025-06-03T20:27:23.455801"
}
```

III. Inicio de sesión (login)

```
❯ curl -X POST http://localhost:5001/api/auth/login -H "Content-Type: application/json"
 -d '{"username": "jane", "password": "jane123"}'
{
  "message": "Login exitoso",
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoyLCJ1c2VybmFtZSI6Imphbm
UifQ.NSLuSsCBG9O0ZuYV-s-0SQz9pAKS3jDDNVBqLVHVDuI",
  "user": {
    "full_name": "Jane Doe",
    "user_id": 2,
    "username": "jane"
  }
}
```

IV. Cantidad de cultivos

```
❯ curl http://localhost:5001/crops
{
  "count": 22,
  "crops": [
    {
      "crop_id": 1,
      "crop_label": 0,
      "crop_name": "apple",
      "description": "Temperate fruit tree",
      "scientific_name": "Malus domestica"
    },
    {
      "crop_id": 2,
      "crop_label": 1,
      "crop_name": "banana",
      "description": "Tropical fruit crop",
      "scientific_name": "Musa acuminata"
    },
    {
      "crop_id": 3,
      "crop_label": 2,
      "crop_name": "blackgram",
      "description": "Black lentil crop",
      "scientific_name": "Vigna mungo"
    },
    {
      "crop_id": 4,
      "crop_label": 3,
      "crop_name": "chickpea",
      "description": "Legume crop rich in protein",
      "scientific_name": "Cicer arietinum"
    },
    {
      "crop_id": 5,
      "crop_label": 4,
      "crop_name": "coconut",
      "description": "Palm tree with multiple uses",
      "scientific_name": "Cocos nucifera"
    },
```

D. Configuración de los servidores en el load balancer

I. Upstream de las instancias del backend

```
1  upstream backend_upstream {
2
3      server 172.22.0.240:5001;
4      server 172.22.0.216:5001;
5  }
6
```

II. Configuración del server

```
server {
    listen 80;

    # Frontend Next.js APIs (auth, etc.)
    location /api/auth/ {
        proxy_pass http://172.22.0.65:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Flask APIs (ML, predictions, etc.)
    location /api/v1/ {
        proxy_pass http://backend_upstream/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Todo lo demás va al frontend
    location / {
        proxy_pass http://172.22.0.65:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

III. Configuración de nginx

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}
```