Manual Técnico – App de Fútbol Híbrida (AWS + Azure)

1. Objetivos del Proyecto

- Desarrollar una aplicación de fútbol que permita a los usuarios consultar partidos, estadísticas de jugadores, equipos y recibir notificaciones en tiempo real.
- Implementar una **arquitectura híbrida en la nube**, utilizando servicios de **AWS y Azure** para optimizar cómputo, almacenamiento, análisis y notificaciones.
- Garantizar escalabilidad, disponibilidad y seguridad de la aplicación, permitiendo su crecimiento para manejar miles de usuarios simultáneamente.
- Fomentar la innovación mediante el análisis de datos en tiempo real y el uso de Machine Learning para generar estadísticas predictivas de partidos y jugadores.

2. Descripción del Proyecto

La App de Fútbol es una plataforma interactiva para fanáticos, jugadores y equipos, que permite:

- Registro y autenticación de usuarios (admin, jugador, fan).
- Consulta de partidos, resultados, estadísticas de jugadores y equipos.
- Subida y almacenamiento de fotos y videos de partidos.
- Generación de alertas y notificaciones en tiempo real sobre eventos importantes (goles, inicio de partidos, resultados).
- Análisis de datos y generación de reportes sobre rendimiento de equipos y jugadores.

El proyecto combina servicios de **AWS** para cómputo y base de datos con **Azure** para análisis avanzado y almacenamiento multimedia, logrando una **solución híbrida** eficiente y flexible.

3. Arquitectura Implementada del Proyecto

Proyecto

App de Fútbol

Arquitectura y diseño de la app

Usuario -> [Internet] -> [Application Load Balancer] -> [Grupo de Auto-Scaling de EC2] -> [RDS (MySQL/PostgreSQL)] & [S3] & [Rekognition/Translate]

. Integración del Traductor (Translate):

- En cualquier página que muestre estadísticas (por ejemplo, la ficha de un jugador), añade un selector de idioma.
- Cuando el usuario elija un idioma, tu aplicación en EC2 tomará los textos (como "Goles", "Asistencias", "Posición") y los enviará a **Translate** para obtener la traducción.
- **No traduzcas los datos dinámicos de la base de datos** (nombres propios, números), solo las etiquetas fijas de la interfaz.

Paso – código

Para Cambiar de Idioma AWS Traslate – para la pagina completa

Configurar AWS SDK en el Backend

1.1 Instalar dependencias AWS en el backend

cd backend

npm install @aws-sdk/client-translate @aws-sdk/client-rekognition

Configurar AWS en el backend (backend/src/config/aws.js)

Servicio de Traducción Real (backend/src/services/translateService.js)

// POST /api/translate/batch - Para traducir múltiples textos

Método de traducción de múltiples textos (Batch)

Este se refiere a la operación de Traducción Asíncrona por Lotes, que se inicia con StartTextTranslationJob.

Característica	Descripción
Uso Principal	Traducción de grandes volúmenes de datos, como documentos completos o una colección de archivos.
Modo de Procesamiento	Traducción asíncrona (Batch). Tú inicias un trabajo (Job) y el proceso se ejecuta en segundo plano. Recibirás una notificación cuando el trabajo haya terminado.
API	StartTextTranslationJob, DescribeTextTranslationJob, ListTextTranslationJobs, etc.
Caso de Uso Típico	Traducción de documentos a gran escala, como: * Manuales de usuario completos,* Grandes colecciones de archivos (por ejemplo, en Amazon S3),* Informes, libros, o bases de datos de contenido.
Entrada/Salida	Los documentos de entrada deben estar almacenados en un bucket de Amazon S3. Amazon Translate coloca los archivos traducidos en otra ubicación de Amazon S3 especificada por ti.
Ventajas Adicionales	Puede manejar la traducción de múltiples documentos a múltiples idiomas de destino en un solo trabajo, e incluso puede detectar automáticamente el idioma de origen de cada documento.

Método principal de traducción singular (Tiempo Real)

Este se refiere a la operación TranslateText en la API de Amazon Translate.

Característica	Descripción
Uso Principal	Traducción de pequeños textos o frases individuales.
Modo de Procesamiento	Traducción en tiempo real (Síncrona). La solicitud se envía y la traducción se recibe inmediatamente.
API	TranslateText.
Caso de Uso Típico	Interacciones de usuario en vivo, como: * Chats en línea,* Traducción de mensajes de correo electrónico cortos,* Aplicaciones móviles donde el usuario espera una respuesta instantánea.
Entrada/Salida	El texto de entrada se pasa directamente en la llamada a la API y el texto traducido se devuelve en la respuesta.

Característica	Descripción
Límites	Tiene límites en el tamaño del texto de entrada (por ejemplo, 5.000 bytes).

https://864899875112.signin.aws.amazon.com/console

User name

futbol-app-user

Console password

futbol-App-user

Retrieve access keys

Access key : e8kZqnRj3Bb29SvXhZ1L9jQp2LLXqRcscS2vZrru

En Frontend :

Para consultar dependiendo el idioma

ntend > src > services > realApiService.js > realApiService

```
// Para configuracion de la API REAL
const API_BASE_URL = 'http://localhost:3000/api';

export const realApiService = {
  // Traducir texto usando el backend con AWS Translate
  translateText: async (text, targetLanguage) => {···
  },

  // Traducir múltiples textos
  translateMultiple: async (texts, targetLanguage) => {···
  },

  // Traducir la carta y datos de un Jugador
  translatePlayerData: async (players, targetLanguage) => {···
  }

};
```

Lenguaje Context : es todo el texto de la pagina para que lo traduzca

```
translateText: async (text, targetLanguage) => {
  try {
    const response = await fetch(`${API_BASE_URL}/translate`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        text,
        targetLanguage
      })
    });

    if (!response.ok) {
      throw new Error(`Error ${response.status}: ${response.statusText}`);
    }

    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error en traducción:', error);
    return {
      success: false,
      translatedText: text, // Texto original como fallback
      error: error.message
    };
  }
}
```

para traslado del Texto

Para manejar los datos de jugadorea se busca en Seachbar y se en api y lo retorna al los gif card

3.1 Diagrama de Arquitectura Híbrida AWS + Azure

Flujo general:

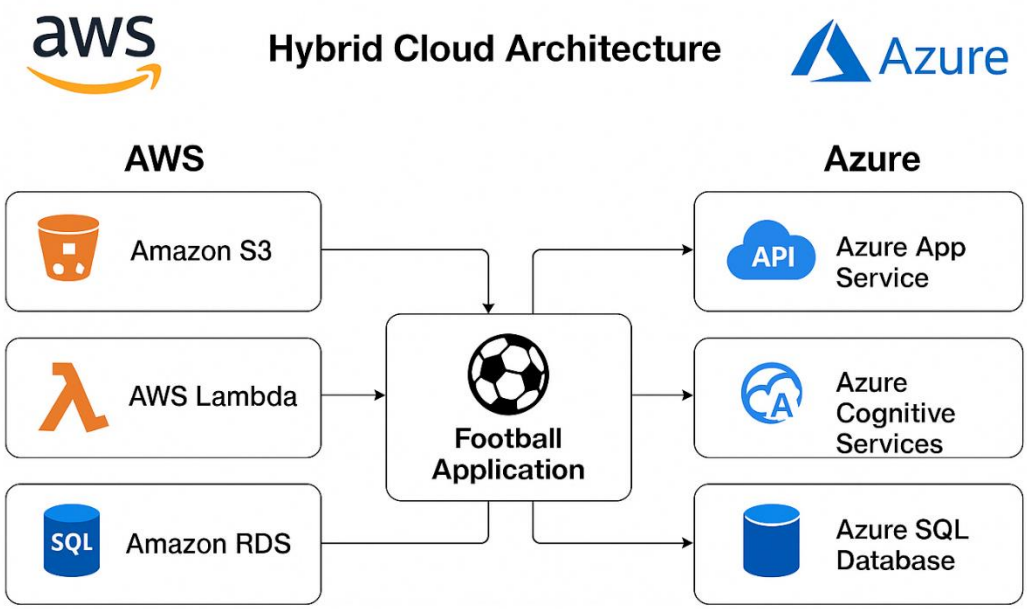
1. El usuario inicia sesión → autenticación con AWS Cognito y Azure AD B2C.
2. Consulta de datos → API en AWS Lambda / EC2 consulta RDS (AWS).
3. Almacenamiento multimedia → S3 (AWS) y replicación a Blob Storage (Azure).
4. Análisis de estadísticas → Azure Synapse / Azure Machine Learning consume datos de RDS.
5. Notificaciones push → AWS SNS y Azure Notification Hubs.

Servicios principales utilizados:

Función	AWS	Azure
Backend / API	EC2 / Lambda	App Service
Base de Datos	RDS (MySQL/PostgreSQL)	SQL Database (para análisis)
Almacenamiento de videos/fotos	S3	Blob Storage
Autenticación	Cognito	Azure AD B2C
Notificaciones push	SNS	Notification Hubs
Análisis de datos / ML	SageMaker	Azure Machine Learning / Synapse

Nota: La arquitectura híbrida permite que cada proveedor maneje lo que mejor sabe hacer: AWS para operaciones críticas y base de datos, Azure para análisis y servicios multimedia.

3.2 Diagrama ER (Base de Datos)



Entidades principales:

- **Usuarios**: id_usuario, nombre, email, contraseña, rol.
- **Equipos**: id_equipo, nombre, ciudad, entrenador.
- **Jugadores**: id_jugador, nombre, posición, id_equipo, estadísticas.
- **Partidos**: id_partido, fecha, id_equipo_local, id_equipo_visitante, marcador.
- **EstadísticasPartido**: id_estadística, id_partido, id_jugador, goles, asistencias, tarjetas.
- **Notificaciones**: id_notificación, id_usuario, mensaje, fecha.

Relaciones:

- Un equipo tiene muchos jugadores.
- Un partido involucra dos equipos.
- Un partido tiene muchas estadísticas de jugadores.
- Un usuario puede recibir múltiples notificaciones.

4. Presupuesto del Proyecto

Concepto	Servicio	Costo estimado mensual (USD)	Observaciones
Backend / API	AWS EC2 / Lambda	Gratis bajo instancia ec2 con poco trafico ,aws lambda es más economico, pero lueso **$0.20** por cada millo de solicitudes. Y tiene costo fijo por hora mientras la instancia este encendida.	Depende del tráfico de usuarios
Base de Datos	RDS MySQL	Por ejemplo, una instancia t4g.micro con 20 GB de almacenamiento puede costar alrededor de **$13.98** al mes, mientras que otras configuraciones pueden ser mucho más caras	Alta disponibilidad y backups
Almacenamiento multimedia	S3 + Azure Blob	depende de la cantidad de datos, la clase de almacenamiento elegida (frecuente, esporádico, archivo) y la cantidad de operaciones de lectura/escritura. Por ejemplo, el almacenamiento estándar en S3 cuesta alrededor de **$0,023/GB**/mes, mientras que el nivel "Hot" (acceso frecuente) en Azure Blob Storage es de aproximadamente **$0,018/GB**/mes.	Videos y fotos de partidos
Autenticación	Cognito + Azure AD B2C	Azure AD B2C ofrece un nivel gratuito para los primeros 50,000 MAU, con un costo de **$0.00325** por MAU adicional. En contraste, Amazon Cognito tiene una tarifa de **$0.015** por MAU que inician sesión	Por usuario activo

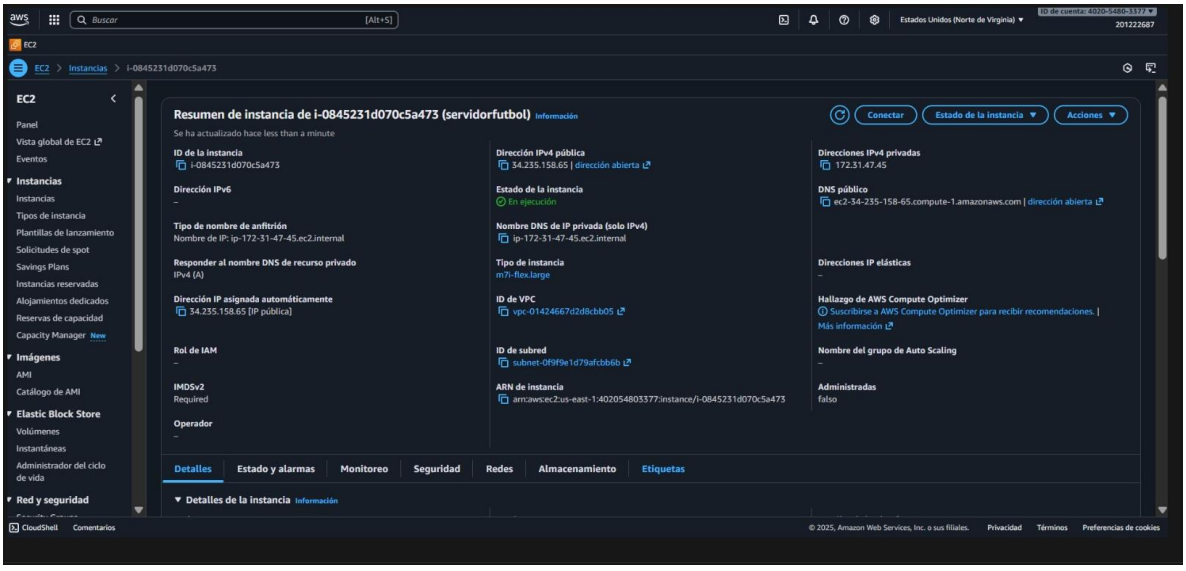
		a través de federación (SAML u OIDC), y hay costos separados para la autenticación multifactor a través de SMS o correo electrónico	
Notificaciones	SNS + Notification Hubs	• Notificaciones push móviles: **$0.50** USD por millón de notificaciones, después de la capa gratuita de 1 millón de notificaciones. • Email o Email-JSON**: $2.00** USD por 100,000 notificaciones, después de la capa gratuita de 1,000 notificaciones. • HTTP/s: **$0.60** USD por millón de notificaciones, después de la capa gratuita de 100,000 notificaciones. • Mensajes SMS de entrada: **$0.0075** USD por mensaje. • Suscripciones: **$0.01** USD por cada millón de suscripciones, más **$0.001** USD por GB de datos de carga útil.	Push y alertas
Análisis de datos	Azure ML / Synapse		Para estadísticas y ML
Total estimado	—	190–330 USD/mes	Aproximado según uso

Costo mensual de análisis de datos

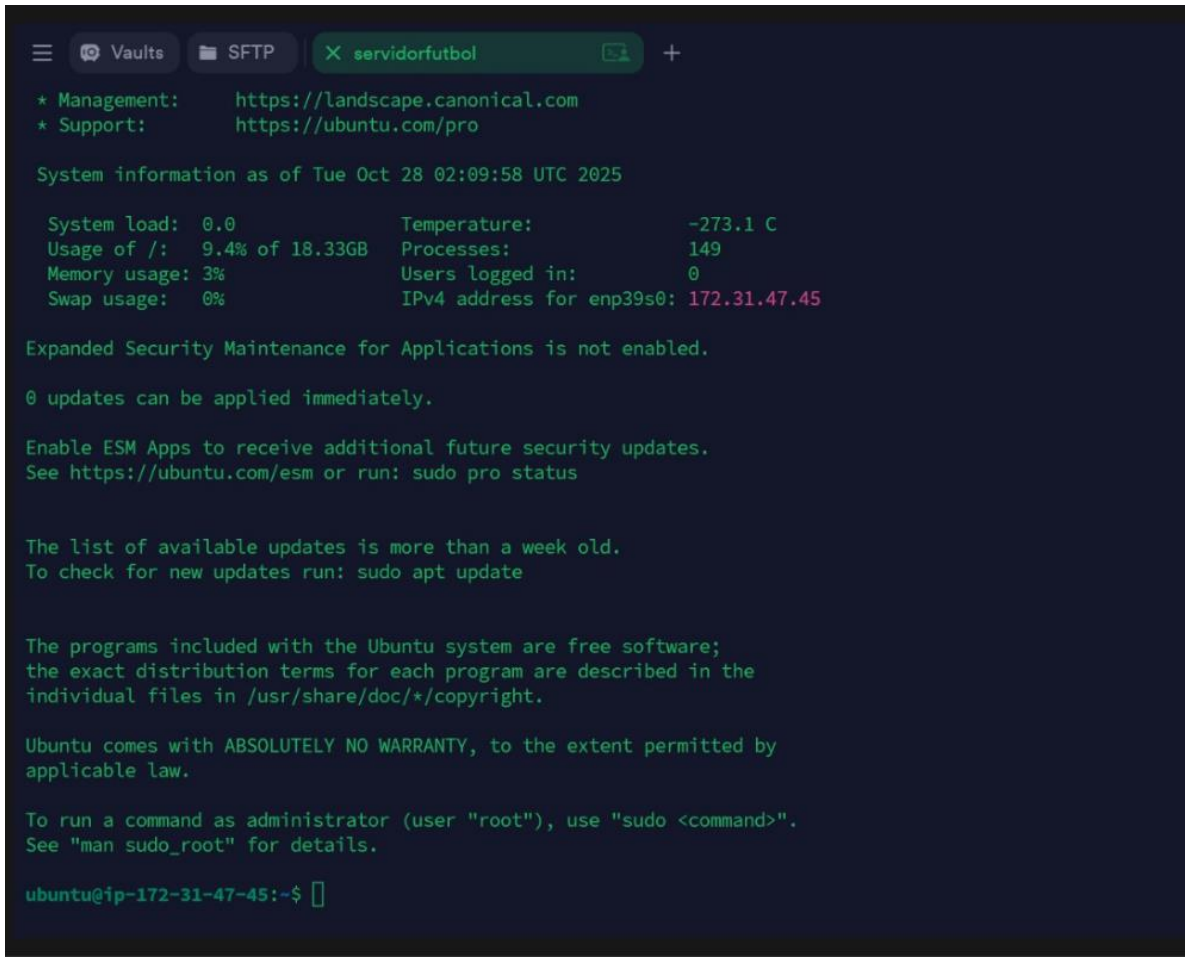
Nivel	Unidades de confirmación de Synapse (SCU)	% de descuento	Precio
1	5.000	6%	$4700
2	10.000	8%	$9200
3	24.000	11%	$21.360
4	60.000	16%	$50.400
5	150.000	22%	$117.000
6	360.000	28%	$259.200

Nota: **Los costos pueden variar según la cantidad de usuarios, almacenamiento requerido y tráfico de la app.**

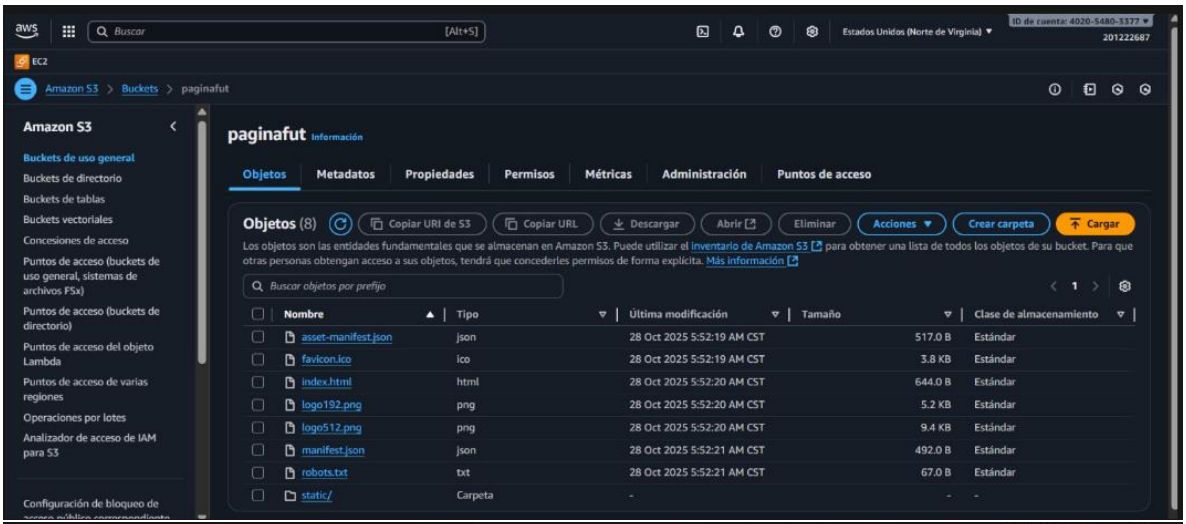
CONFIGURACION DE LA IC2

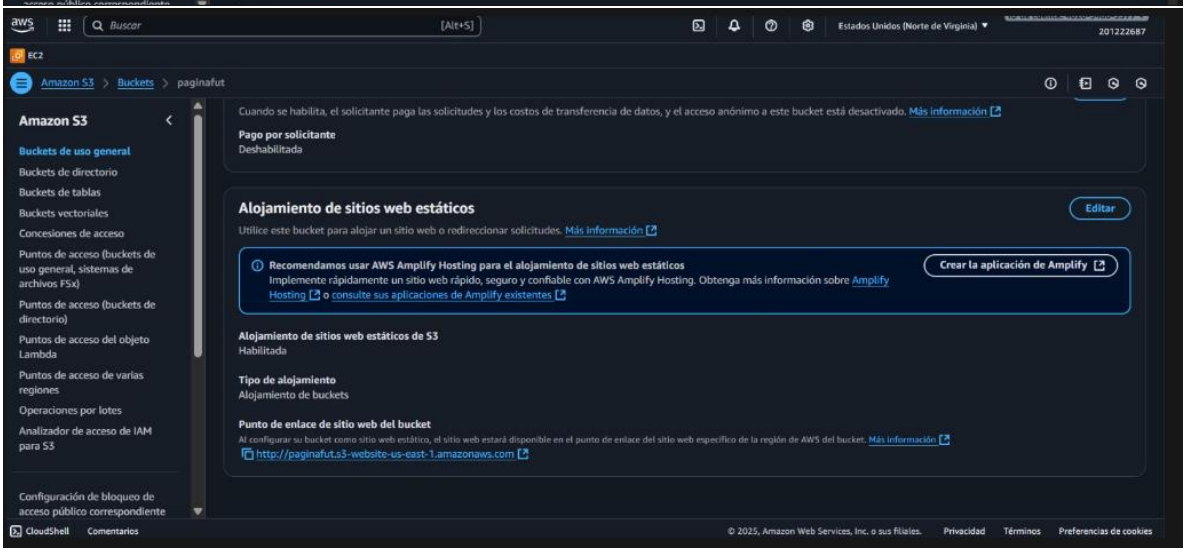


CORRIENDO

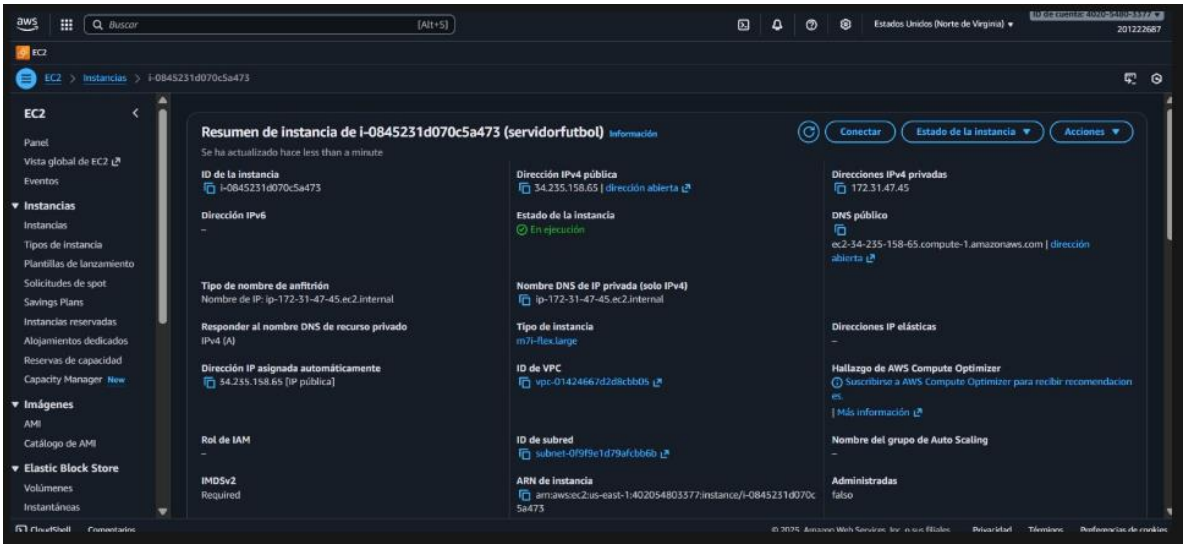

Vaults
SFTP
servidorfutbol
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Tue Oct 28 02:09:58 UTC 2025
System load: 0.0
Usage of /: 9.4% of 18.33GB
Memory usage: 3%
Swap usage: 0%
Temperature: -273.1 C
Processes: 149
Users logged in: 0
IPv4 address for enp39s0: 172.31.47.45
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-47-45:~$

S3

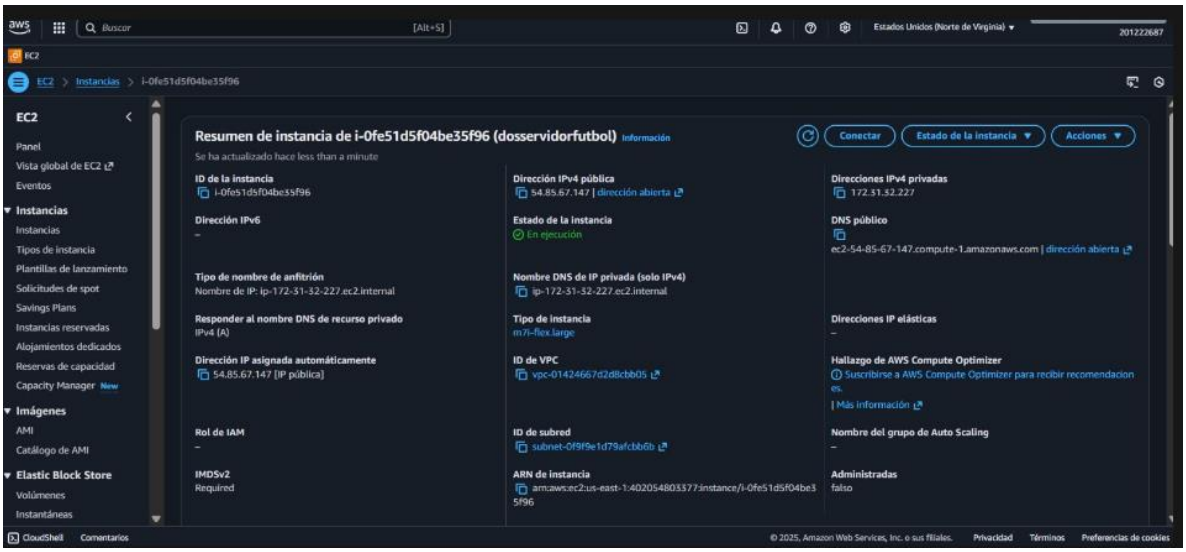




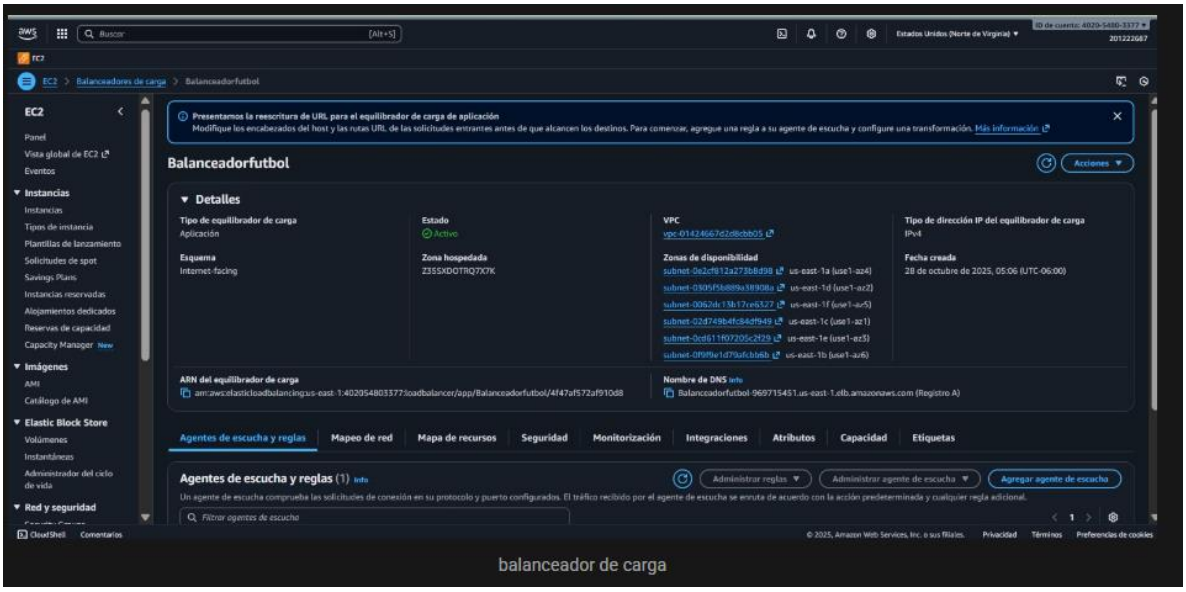
Servidor uno



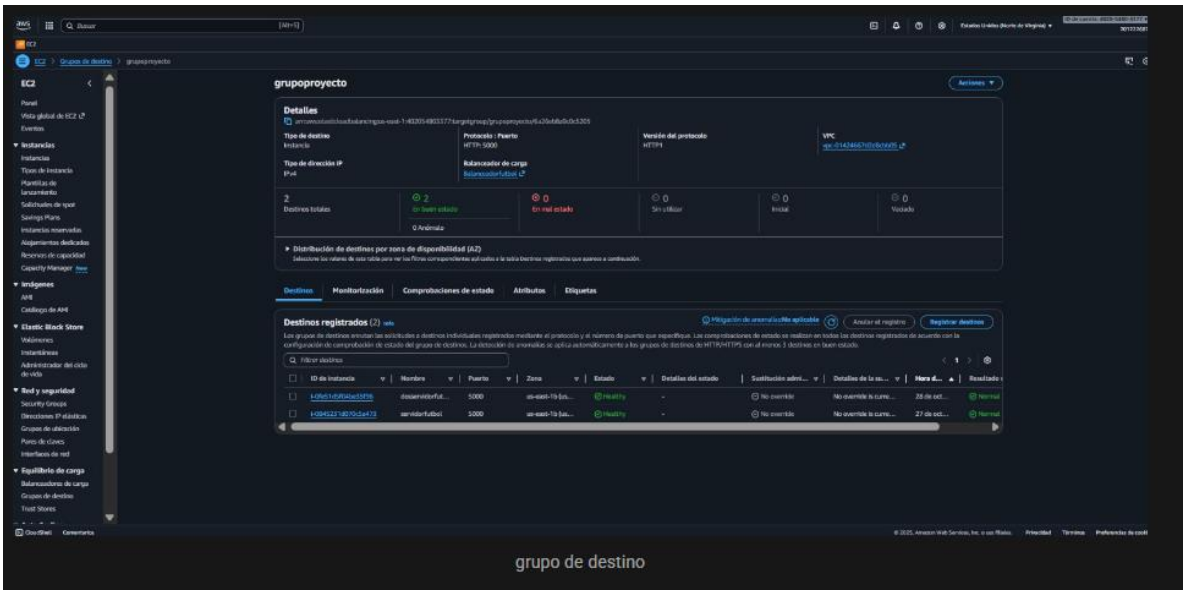
Servidor 2



Balanceador de carga

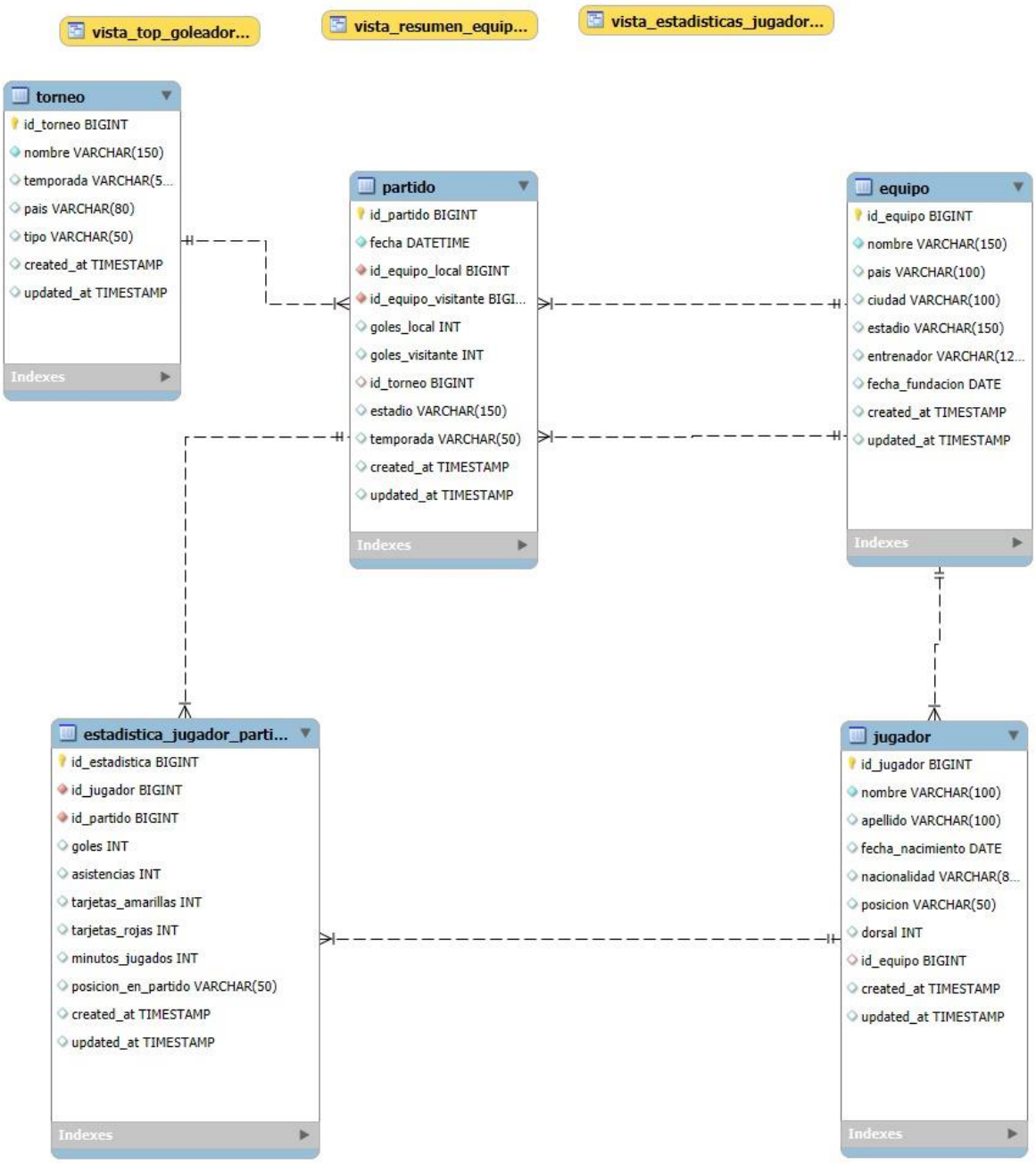


Grupo destino



grupo de destino

Diagramas



Script de la base de datos en RDS

Script SQL — App de Fútbol (MySQL / Amazon RDS)

```sql
-- ============================================================
-- SCRIPT: app_futbol_rds_mysql.sql
-- Descripción: Base de datos para aplicación de fútbol
-- Autor: GLS
-- Base: Amazon RDS (MySQL 8+)
-- ============================================================

-- Crear base de datos (opcional)
CREATE DATABASE IF NOT EXISTS app_futbol CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
USE app_futbol;

-- ==========================
-- 1. TABLAS PRINCIPALES
-- ==========================

-- Tabla: Equipo
CREATE TABLE IF NOT EXISTS equipo (
    id_equipo BIGINT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(150) NOT NULL,
    pais VARCHAR(100),
    ciudad VARCHAR(100),
    estadio VARCHAR(150),
    entrenador VARCHAR(120),
    fecha_fundacion DATE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- Tabla: Jugador
CREATE TABLE IF NOT EXISTS jugador (
    id_jugador BIGINT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100),
    fecha_nacimiento DATE,
    nacionalidad VARCHAR(80),
    posicion VARCHAR(50),
    dorsal INT,
    id_equipo BIGINT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    CONSTRAINT fk_jugador_equipo FOREIGN KEY (id_equipo) REFERENCES
equipo(id_equipo) ON DELETE SET NULL,
```

```
    CONSTRAINT uq_dorsal_equipo UNIQUE (id_equipo, dorsal)
);

-- Tabla: Torneo
CREATE TABLE IF NOT EXISTS torneo (
    id_torneo BIGINT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(150) NOT NULL,
    temporada VARCHAR(50),
    pais VARCHAR(80),
    tipo VARCHAR(50), -- Liga, Copa, Amistoso, etc.
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- Tabla: Partido
CREATE TABLE IF NOT EXISTS partido (
    id_partido BIGINT AUTO_INCREMENT PRIMARY KEY,
    fecha DATETIME NOT NULL,
    id_equipo_local BIGINT NOT NULL,
    id_equipo_visitante BIGINT NOT NULL,
    goles_local INT DEFAULT 0,
    goles_visitante INT DEFAULT 0,
    id_torneo BIGINT,
    estadio VARCHAR(150),
    temporada VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    CONSTRAINT fk_partido_local FOREIGN KEY (id_equipo_local) REFERENCES
equipo(id_equipo) ON DELETE RESTRICT,
    CONSTRAINT fk_partido_visitante FOREIGN KEY (id_equipo_visitante)
REFERENCES equipo(id_equipo) ON DELETE RESTRICT,
    CONSTRAINT fk_partido_torneo FOREIGN KEY (id_torneo) REFERENCES
torneo(id_torneo) ON DELETE SET NULL,
    CONSTRAINT chk_equipos_distintos CHECK (id_equipo_local <>
id_equipo_visitante)
);

-- Tabla: Estadísticas por jugador en cada partido
CREATE TABLE IF NOT EXISTS estadistica_jugador_partido (
    id_estadistica BIGINT AUTO_INCREMENT PRIMARY KEY,
    id_jugador BIGINT NOT NULL,
    id_partido BIGINT NOT NULL,
    goles INT DEFAULT 0,
    asistencias INT DEFAULT 0,
    tarjetas_amarillas INT DEFAULT 0,
```

```
    tarjetas_rojas INT DEFAULT 0,
    minutos_jugados INT DEFAULT 0,
    posicion_en_partido VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    CONSTRAINT fk_estadistica_jugador FOREIGN KEY (id_jugador) REFERENCES
jugador(id_jugador) ON DELETE CASCADE,
    CONSTRAINT fk_estadistica_partido FOREIGN KEY (id_partido) REFERENCES
partido(id_partido) ON DELETE CASCADE,
    CONSTRAINT uq_estadistica_unica UNIQUE (id_jugador, id_partido)
);

-- ============================
-- 2. DATOS DE EJEMPLO
-- ============================

-- Equipos
INSERT INTO equipo (nombre, pais, ciudad, estadio, entrenador, fecha_fundacion)
VALUES
('Real Madrid', 'España', 'Madrid', 'Santiago Bernabéu', 'Carlo Ancelotti', '1902-03-06'),
('FC Barcelona', 'España', 'Barcelona', 'Spotify Camp Nou', 'Xavi Hernández', '1899-11-29'),
('Manchester City', 'Inglaterra', 'Manchester', 'Etihad Stadium', 'Pep Guardiola', '1880-04-
16'),
('Liverpool FC', 'Inglaterra', 'Liverpool', 'Anfield', 'Jürgen Klopp', '1892-06-03');

-- Jugadores
INSERT INTO jugador (nombre, apellido, fecha_nacimiento, nacionalidad, posicion,
dorsal, id_equipo) VALUES
('Vinícius', 'Júnior', '2000-07-12', 'Brasil', 'Delantero', 7, 1),
('Jude', 'Bellingham', '2003-06-29', 'Inglaterra', 'Mediocampista', 5, 1),
('Pedri', 'González', '2002-11-25', 'España', 'Mediocampista', 8, 2),
('Erling', 'Haaland', '2000-07-21', 'Noruega', 'Delantero', 9, 3),
('Mohamed', 'Salah', '1992-06-15', 'Egipto', 'Delantero', 11, 4);

-- Torneos
INSERT INTO torneo (nombre, temporada, pais, tipo) VALUES
('UEFA Champions League', '2024-2025', 'Europa', 'Internacional'),
('LaLiga', '2024-2025', 'España', 'Liga');

-- Partidos
INSERT INTO partido (fecha, id_equipo_local, id_equipo_visitante, goles_local,
goles_visitante, id_torneo, estadio, temporada)
VALUES
('2025-09-10 20:00:00', 1, 2, 3, 1, 2, 'Santiago Bernabéu', '2024-2025'),
('2025-09-15 18:00:00', 3, 4, 2, 2, 1, 'Etihad Stadium', '2024-2025');
```

```sql
-- Estadísticas por jugador
INSERT INTO estadistica_jugador_partido (id_jugador, id_partido, goles, asistencias,
tarjetas_amarillas, minutos_jugados)
VALUES
(1, 1, 2, 1, 0, 90),
(2, 1, 1, 0, 1, 85),
(3, 1, 0, 1, 0, 90),
(4, 2, 2, 0, 0, 90),
(5, 2, 1, 1, 0, 90);

-- ============================
-- 3. VISTAS ÚTILES
-- ============================

-- Vista: Top goleadores por torneo
CREATE OR REPLACE VIEW vista_top_goleadores AS
SELECT
    j.id_jugador,
    CONCAT(j.nombre, ' ', j.apellido) AS jugador,
    e.nombre AS equipo,
    t.nombre AS torneo,
    SUM(es.goles) AS goles_totales
FROM estadistica_jugador_partido es
JOIN jugador j ON es.id_jugador = j.id_jugador
JOIN partido p ON es.id_partido = p.id_partido
JOIN equipo e ON j.id_equipo = e.id_equipo
LEFT JOIN torneo t ON p.id_torneo = t.id_torneo
GROUP BY j.id_jugador, jugador, equipo, torneo
ORDER BY goles_totales DESC;

-- Vista: Resumen de equipos en torneo
CREATE OR REPLACE VIEW vista_resumen_equipos AS
SELECT
    eq.id_equipo,
    eq.nombre AS equipo,
    t.nombre AS torneo,
    COUNT(p.id_partido) AS partidos_jugados,
    SUM(CASE WHEN p.id_equipo_local = eq.id_equipo THEN p.goles_local ELSE
p.goles_visitante END) AS goles_a_favor,
    SUM(CASE WHEN p.id_equipo_local = eq.id_equipo THEN p.goles_visitante ELSE
p.goles_local END) AS goles_en_contra
FROM equipo eq
JOIN partido p ON (p.id_equipo_local = eq.id_equipo OR p.id_equipo_visitante =
eq.id_equipo)
LEFT JOIN torneo t ON p.id_torneo = t.id_torneo
GROUP BY eq.id_equipo, eq.nombre, t.nombre;
```

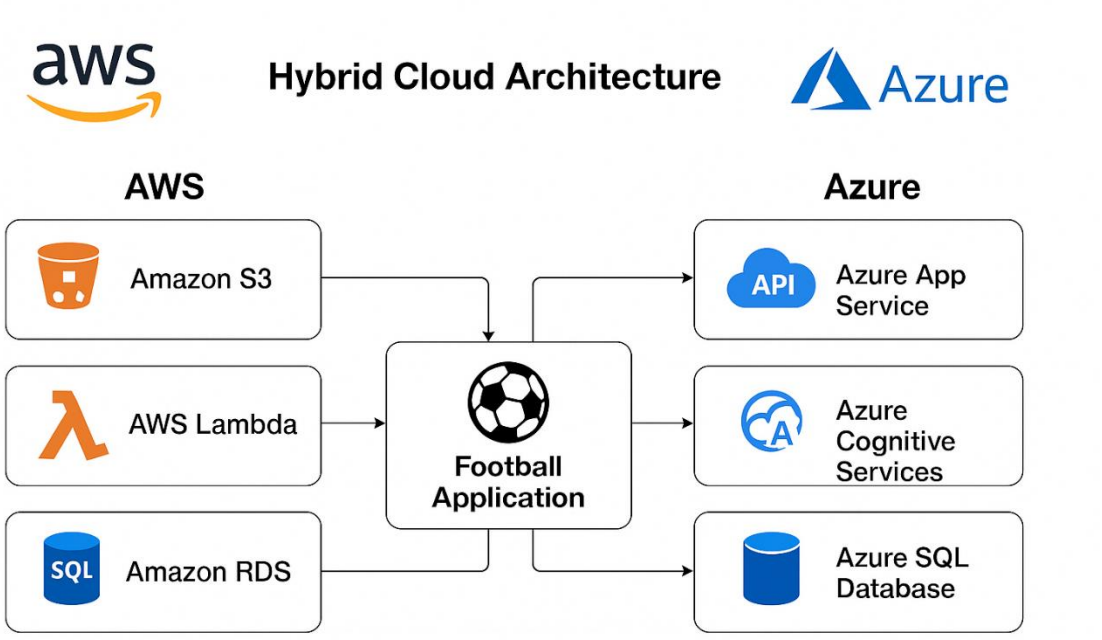
```sql
-- Vista: Estadísticas globales de jugadores
CREATE OR REPLACE VIEW vista_estadisticas_jugadores AS
SELECT
    j.id_jugador,
    CONCAT(j.nombre, ' ', j.apellido) AS jugador,
    e.nombre AS equipo,
    SUM(es.goles) AS total_goles,
    SUM(es.asistencias) AS total_asistencias,
    SUM(es.tarjetas_amarillas) AS amarillas,
    SUM(es.tarjetas_rojas) AS rojas,
    COUNT(es.id_estadistica) AS partidos
FROM estadistica_jugador_partido es
JOIN jugador j ON es.id_jugador = j.id_jugador
JOIN equipo e ON j.id_equipo = e.id_equipo
GROUP BY j.id_jugador, jugador, equipo
ORDER BY total_goles DESC;

-- ============================
-- 4. PRUEBAS RÁPIDAS
-- ============================

-- Consultar top goleadores
-- SELECT * FROM vista_top_goleadores;

-- Consultar resumen de equipos
-- SELECT * FROM vista_resumen_equipos;

-- Consultar estadísticas de jugadores
-- SELECT * FROM vista_estadisticas_jugadores;

-- ============================
-- FIN DEL SCRIPT
-- ============================
```

cómo usarlo en Amazon RDS

1. Crea una base de datos MySQL en RDS (ejemplo: app_futbol).
2. Conéctate con un cliente como:

mysql -h <endpoint> -u <usuario> -p app_futbol < app_futbol_rds_mysql.sql

3. Verifica:
   SHOW TABLES;
   SELECT * FROM vista_top_goleadores;



CONFIGURACION DE LA BASE DE DATOS

BASE DE DATOS. (MYSQL)

Identificador: bdfutbol

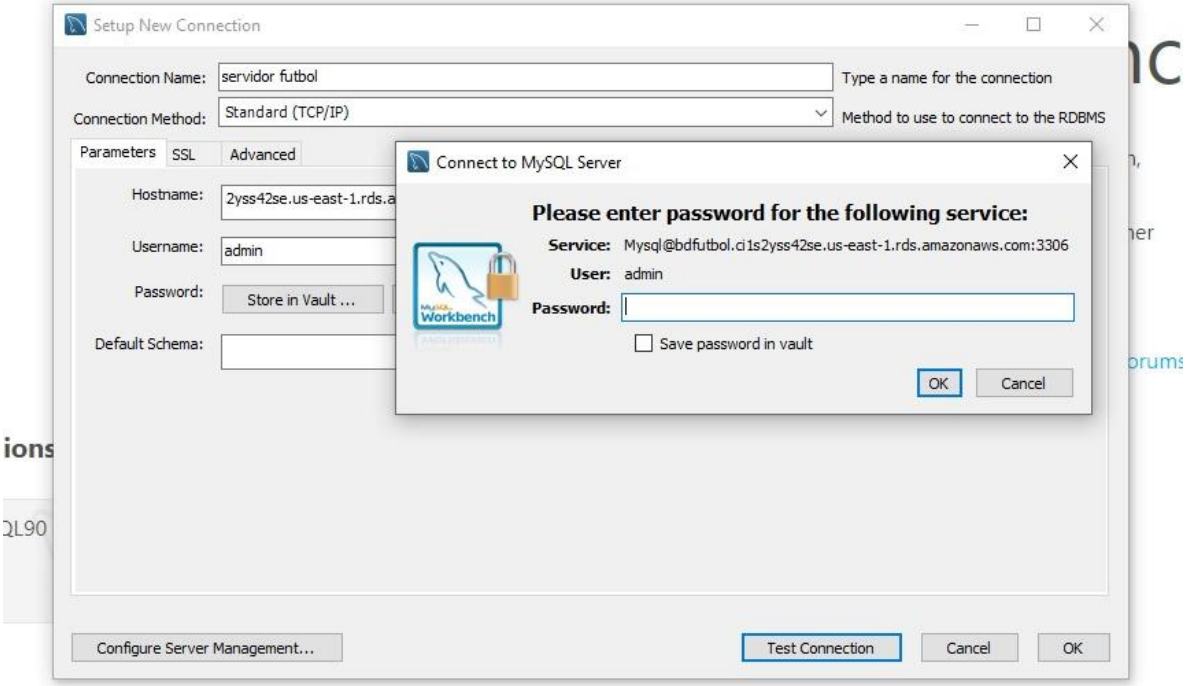
Nombre de usuario maestro: admin

Contraseña maestra: Pybdfutbol

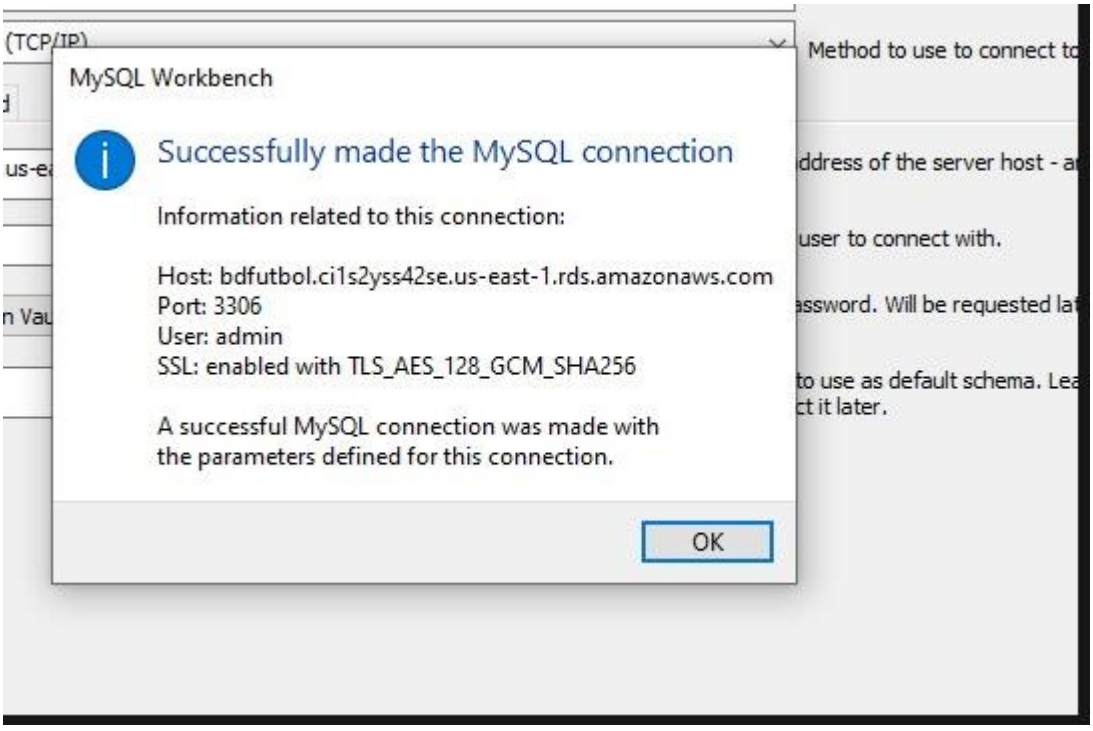
Punto de enlace: bdfutbol.ci1s2yss42se.us-east-1.rds.amazonaws.com

bdfutbol.ci1s2yss42se.us-east-1.rds.amazonaws.com

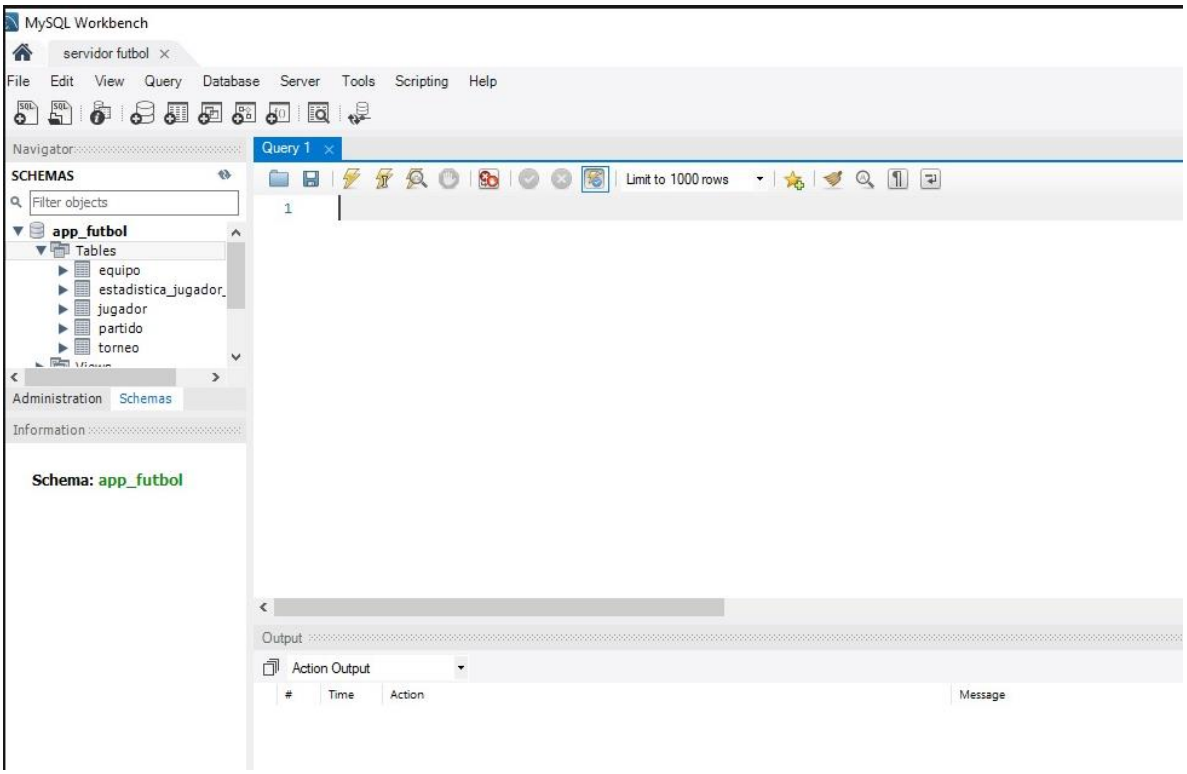
Con esto configuramos MySql Workbench



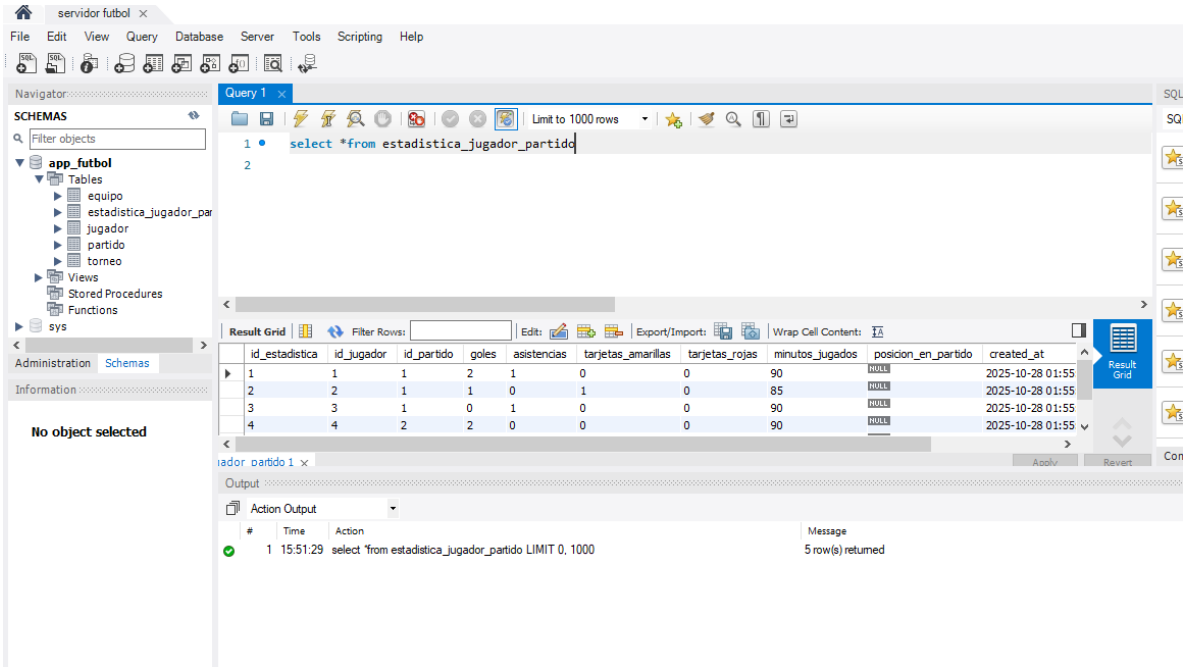
Conexion configurada exitosamente



Base de datos conectada



Algunas consultas efectuadas ya en la base de datos

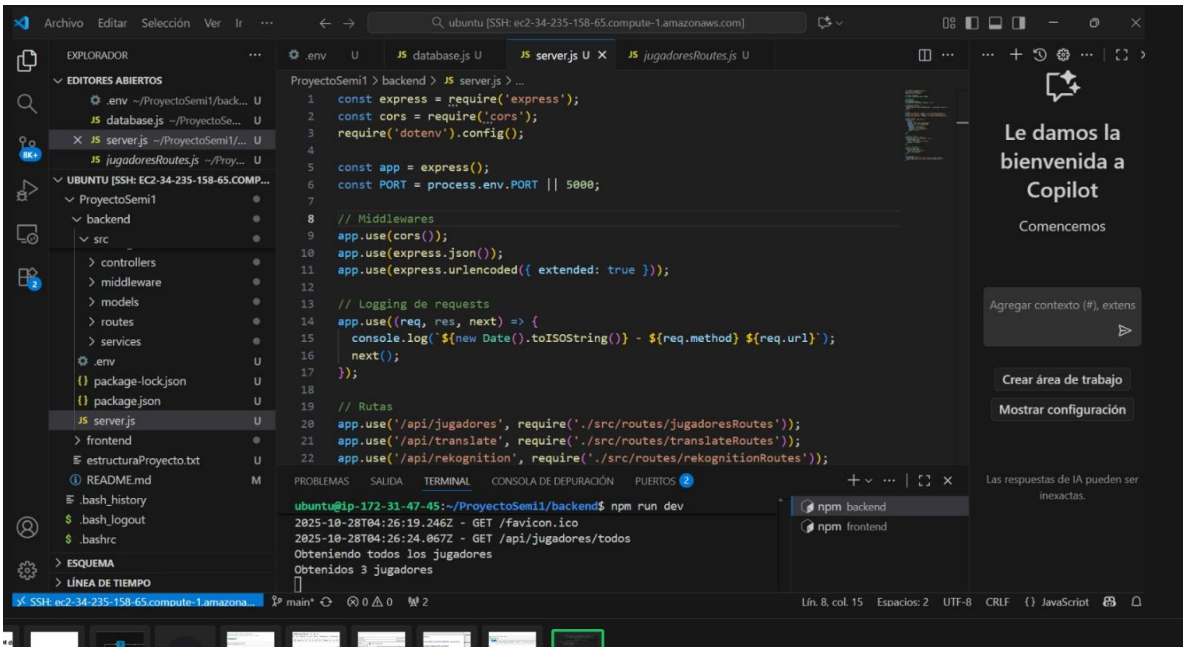


Breve Investigación de los Servicios Utilizados

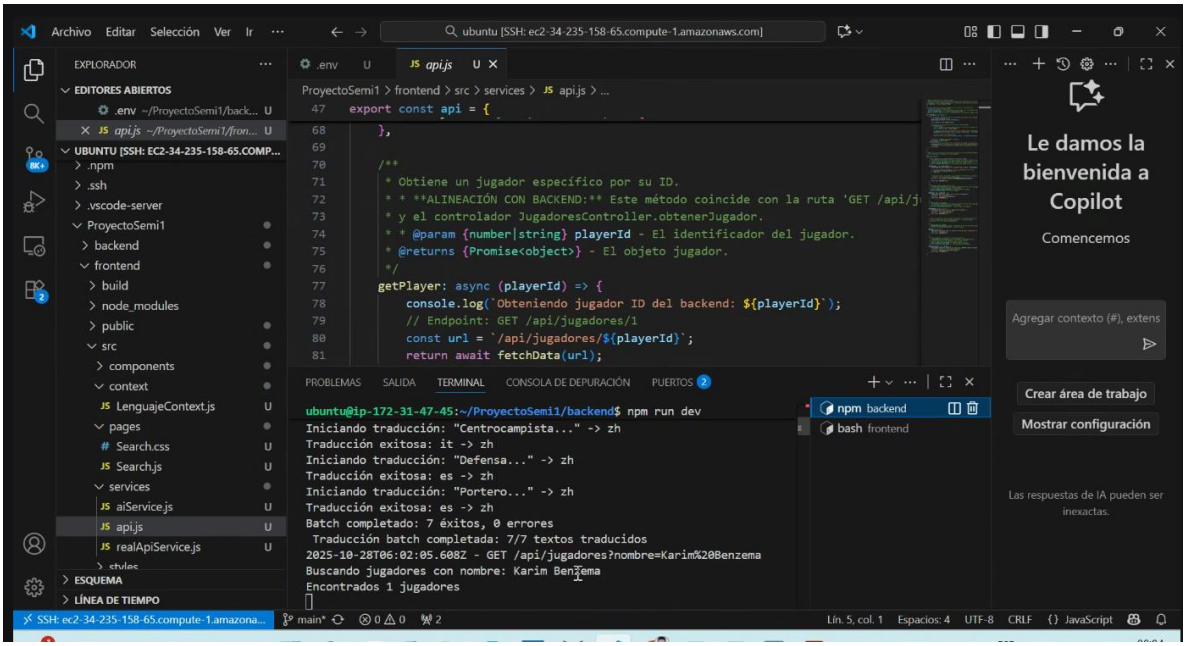
1Servicios de AWS

1. **AWS EC2 (Elastic Compute Cloud)**
   - Servicio de **computación en la nube** que permite ejecutar servidores virtuales escalables.
   - Útil para alojar la API de la aplicación, gestionar solicitudes de usuarios y procesar datos en tiempo real.
   - Permite **autoescalado** según la demanda del tráfico.
2. **AWS Lambda**
   - Servicio de **computación serverless** que ejecuta código sin necesidad de administrar servidores.
   - Ideal para funciones específicas como enviar notificaciones o procesar estadísticas de partidos.
   - Se cobra según el tiempo de ejecución y la cantidad de solicitudes.
3. **Amazon RDS (Relational Database Service)**
   - Base de datos **relacional administrada**, compatible con MySQL, PostgreSQL, etc.
   - Permite **alta disponibilidad, backups automáticos y escalabilidad vertical y horizontal**.
   - Aquí se almacenan usuarios, equipos, jugadores y partidos.
4. **Amazon S3 (Simple Storage Service)**
   - Servicio de **almacenamiento de objetos** en la nube.
   - Ideal para **videos, fotos y archivos multimedia** de los partidos.
   - Permite replicación a otras regiones y a Azure Blob Storage para solución híbrida.
5. **AWS Cognito**
   - Servicio de **gestión de identidad y autenticación** de usuarios.
   - Permite registro, inicio de sesión y control de acceso seguro.
   - Se integra con otros servicios como Lambda y API Gateway.
6. **AWS SNS (Simple Notification Service)**
   - Servicio de **notificaciones push y mensajes** en tiempo real.
   - Puede enviar alertas a usuarios de la app sobre goles, resultados o noticias importantes.

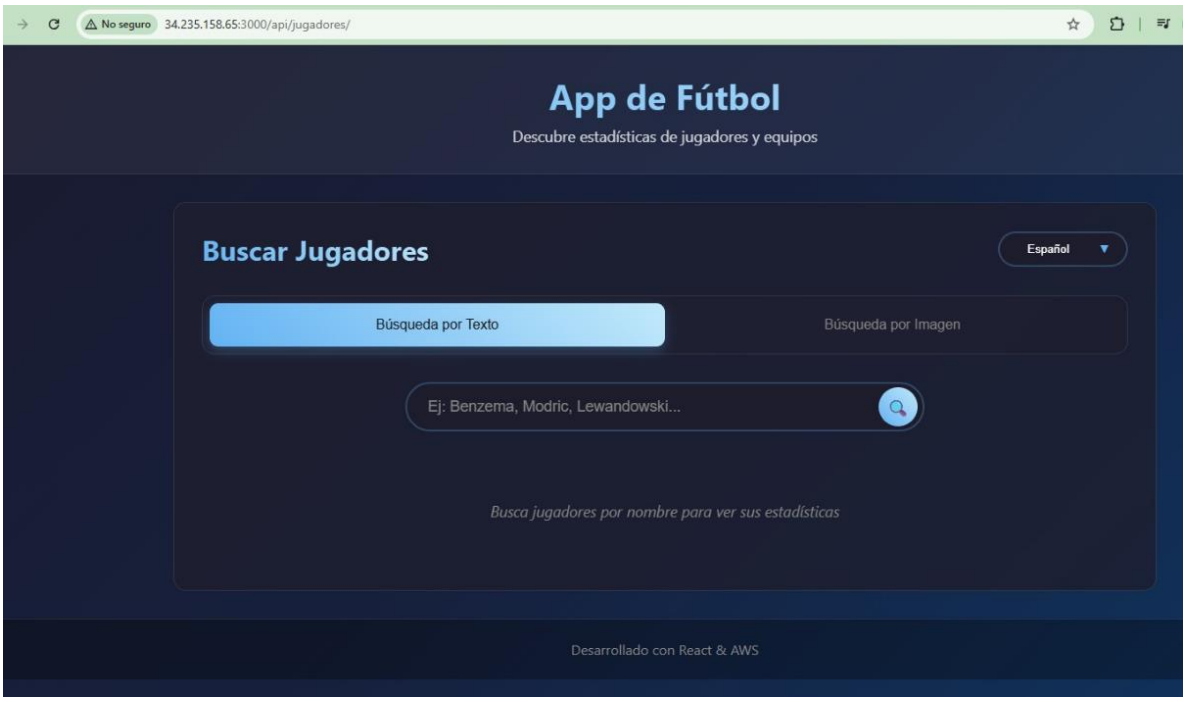
BackEnd



Buscando jugador desde el BackEnd



FrontEnd



BUSCAR JUGADORES