

UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD DE INGENIERÍA

INVESTIGACION DE OPERACIONES II



PROYECTO EVALUACIÓN PARCIAL 1

ALUMNOS:

367770 — JAZMÍN CRUZ GONZÁLEZ

367977 — SARAHY CHAPARRO RAMÍREZ

PROFESOR:

JOSÉ SAÚL DE LIRA MIRAMONTES

CHIHUAHUA, CHIHUAHUA A 01 DE OCTUBRE DEL 2025.

Introducción

El presente documento expone el desarrollo de un Sistema de Gestión de Blog, diseñado como una herramienta que facilita la creación, organización y administración de publicaciones en línea. Para su construcción se empleó Python Flask, encargado de la interacción con el usuario, y Oracle Database con PL/SQL, responsable del manejo y la gestión de la información. Con esta integración se busca ofrecer una solución práctica, eficiente y adaptable a las necesidades actuales de publicación digital.

Objetivo del proyecto

En la era digital actual, los sistemas de gestión de contenidos requieren arquitecturas escalables y mantenibles. Este proyecto aborda este desafío mediante la implementación de un blog completo que separa claramente las responsabilidades entre:

- Frontend/API: Desarrollado en Python Flask, proporcionando endpoints RESTful
- Lógica de Negocio: Implementada en PL/SQL dentro de la base de datos Oracle
- Persistencia: Gestión de datos mediante tablas relacionales optimizadas

Alcance del sistema

El sistema desarrollado permite la gestión integral de contenidos blog, incluyendo:

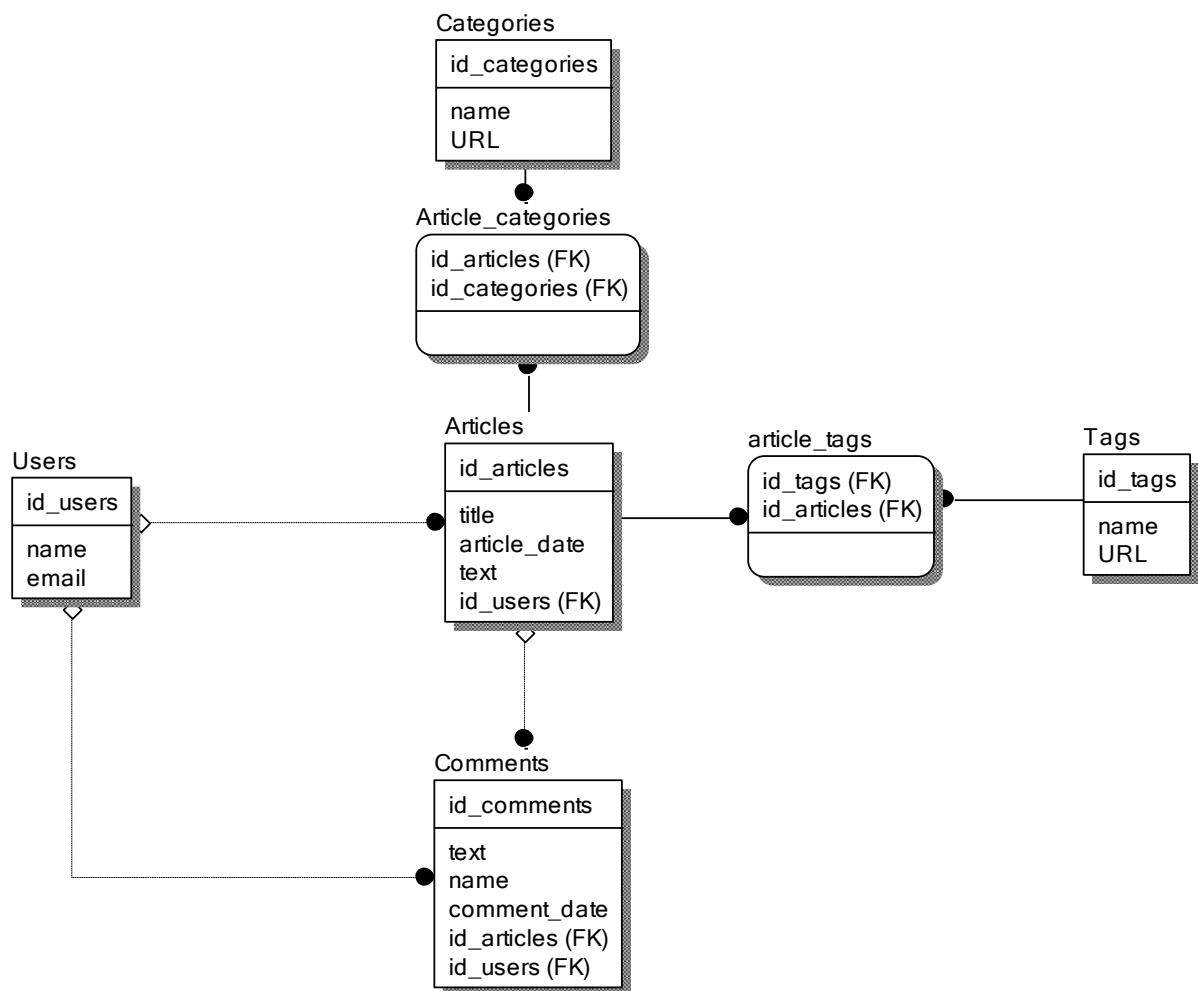
- Gestión de Usuarios: Registro y administración de autores
- Publicación de Artículos: Creación, edición y eliminación de contenido
- Sistema de Comentarios: Interacción con los lectores
- Categorización Avanzada: Organización mediante tags y categorías
- Relaciones Complejas: Implementación de relaciones muchos-a-muchos

Metodología y Tecnologías

La solución emplea una arquitectura de tres capas que garantiza escalabilidad y mantenibilidad:

- Capa de Presentación: API REST con Flask y JSON
- Capa de Lógica de Negocio: Procedimientos y funciones PL/SQL
- Capa de Datos: Oracle Database con tablas relacionales

Modelo entidad-relación



Esquema de tablas

- **Tablas principales**

USERS: Información de usuarios/autores

CREATE TABLE users (id, name, email);

ARTICLES: Contenido principal del blog

CREATE TABLE articles (id, title, article_date, text, user_id);

COMMENTS: Comentarios de lectores

CREATE TABLE comments (id, article_id, name, text, comment_date);

TAGS: Etiquetas de clasificación

CREATE TABLE tags (id, name, url);

CATEGORIES: Categorías de organización

CREATE TABLE categories (id, name, url);

- **Relaciones N a N**

ARTICLE_TAGS: Relación artículos ↔ tags

CREATE TABLE article_tags (article_id, tag_id);

ARTICLE_CATEGORIES: Relación artículos ↔ categorías

CREATE TABLE article_categories (article_id, category_id);

```
create table users (  
    id number primary key,  
    name varchar2(100) not null,  
    email varchar2(150)  
);  
  
create table articles (  
    id number primary key,  
    title varchar2(200) not null,  
    article_date date default sysdate,  
    text clob,  
    user_id number references users(id)  
);  
  
create table comments (  
    id number primary key,  
    article_id number references articles(id),  
    name varchar2(100) not null,  
    text clob,  
    comment_date date default sysdate  
);  
  
create table tags (  
    id number primary key,  
    name varchar2(100) not null,  
    url varchar2(200)  
);  
  
create table categories (  
    id number primary key,  
    name varchar2(100) not null,  
    url varchar2(200)  
);  
  
create table article_tags (  
    article_id number references articles(id),  
    tag_id number references tags(id),  
    primary key(article_id, tag_id)  
);  
  
create table article_categories (  
    article_id number references articles(id),  
    category_id number references categories(id),  
    primary key(article_id, category_id)  
);
```

```
Table USERS creado.  
  
Table ARTICLES creado.  
  
Table COMMENTS creado.  
  
Table TAGS creado.  
  
Table CATEGORIES creado.  
  
Table ARTICLE_TAGS creado.  
  
Table ARTICLE_CATEGORIES creado.
```

Flujo de datos

1. Cliente envía petición HTTP a endpoint Flask
2. Flask valida datos y llama a función/procedimiento PL/SQL
3. PL/SQL ejecuta lógica de negocio en la base de datos
4. Oracle retorna resultados a Flask
5. Flask formatea respuesta JSON para el cliente

FUNCIONES Y PROCEDIMIENTOS PL/SQL

1. register_user

Registra nuevos usuarios en el sistema. Genera automáticamente un ID único y retorna el ID del usuario creado.

2. create_tag

Crea nuevas etiquetas para categorizar artículos. Retorna el ID del tag creado, con URL opcional.

3. create_category

Crea nuevas categorías para organizar el contenido del blog. Retorna el ID de la categoría creada.

4. add_article

Publica nuevos artículos en el blog. Asigna fecha automática y retorna el ID del artículo creado.

5. count_comments

Cuenta los comentarios de un artículo específico. Retorna el número total de comentarios para estadísticas.

6. get_articles

Obtiene todos los artículos con información del autor. Retorna un cursor con los datos ordenados por fecha.

PROCEDIMIENTOS

1. add_comment

Agrega comentarios a los artículos. Obtiene automáticamente el nombre del usuario que comenta.

2. add_tag_to_article

Asocia tags a artículos. Maneja relaciones muchos a muchos e ignora duplicados silenciosamente.

3. add_category_to_article

Asocia categorías a artículos. Gestiona relaciones N a N entre artículos y categorías.

4. delete_article_complete

Elimina artículos completamente con todas sus dependencias. Borra tags, categorías y comentarios relacionados en cascada.

-- FUNCIONES Y PROCEDIMIENTOS PL/SQL

```
create or replace function register_user(
    p_name in varchar2,
    p_email in varchar2
) return number is
    v_id number;
begin
    select nvl(max(id),0)+1 into v_id from users;
    insert into users (id, name, email) values (v_id, p_name, p_email);
    return v_id;
end register_user;
/

create or replace function create_tag(
    p_name in varchar2,
    p_url in varchar2 default null
) return number is
    v_id number;
begin
    select nvl(max(id),0)+1 into v_id from tags;
    insert into tags (id, name, url) values (v_id, p_name, p_url);
    return v_id;
end create_tag;
/

create or replace function create_category(
    p_name in varchar2,
    p_url in varchar2 default null
) return number is
    v_id number;
begin
    select nvl(max(id),0)+1 into v_id from categories;
    insert into categories (id, name, url) values (v_id, p_name, p_url);
    return v_id;
end create_category;
/
```



```

create or replace function add_article(
    p_title in varchar2,
    p_text in clob,
    p_user_id in number
) return number is
    v_id number;
begin
    select nvl(max(id),0)+1 into v_id from articles;
    insert into articles (id, title, text, article_date, user_id)
    values (v_id, p_title, p_text, sysdate, p_user_id);
    return v_id;
end add_article;
/

create or replace procedure add_comment(
    p_article_id in number,
    p_user_id in number,
    p_text in clob
) is
    v_user_name varchar2(100);
    v_comment_id number;
begin
    select name into v_user_name from users where id = p_user_id;
    select nvl(max(id),0)+1 into v_comment_id from comments;
    insert into comments (id, article_id, name, text, comment_date)
    values (v_comment_id, p_article_id, v_user_name, p_text, sysdate);
end add_comment;
/

create or replace procedure add_tag_to_article(
    p_article_id in number,
    p_tag_id in number
) is
begin
    insert into article_tags (article_id, tag_id) values (p_article_id, p_tag_id);
exception
    when dup_val_on_index then null;
end add_tag_to_article;

```

```

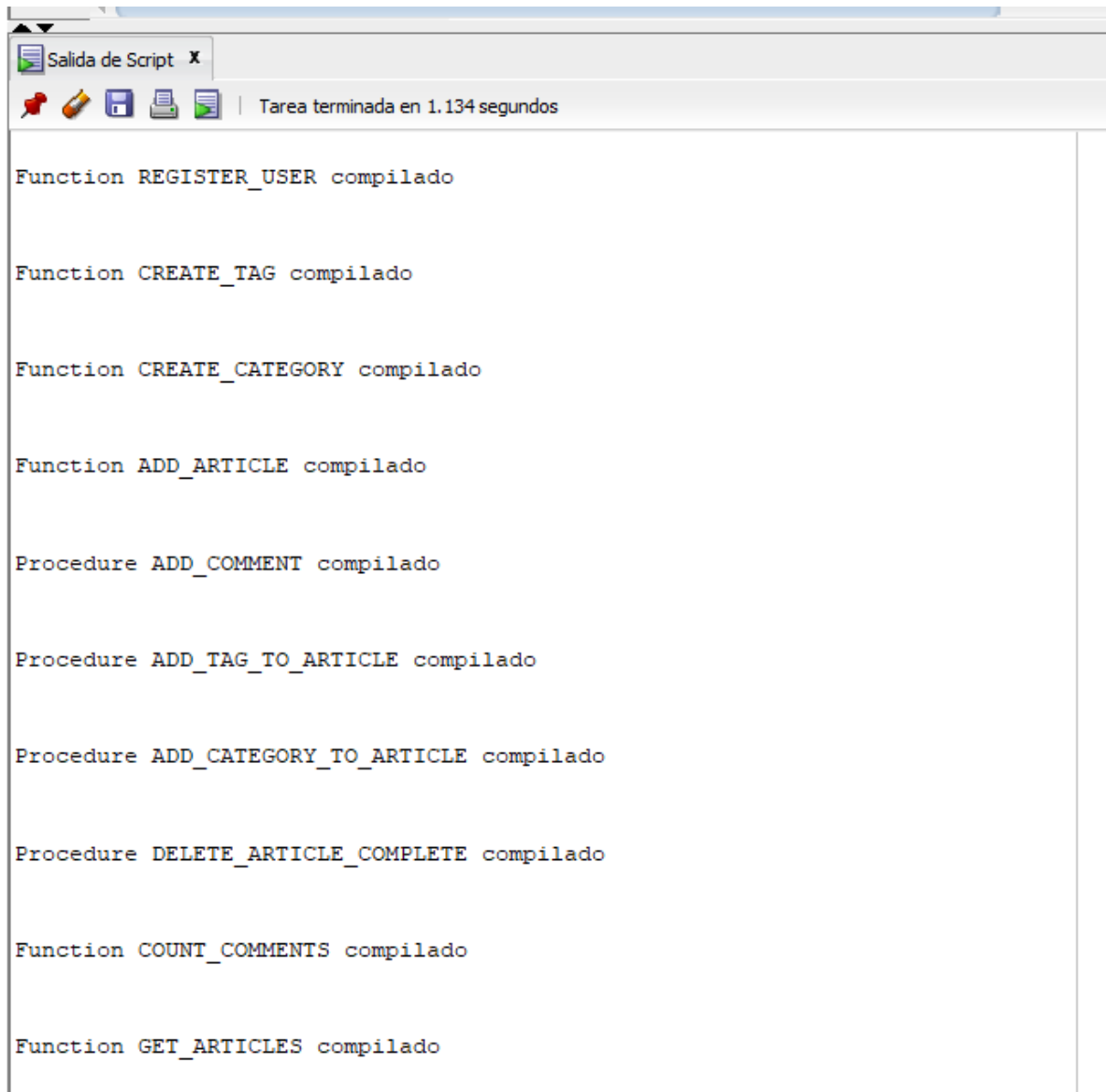
create or replace procedure add_category_to_article(
    p_article_id in number,
    p_category_id in number
) is
begin
    insert into article_categories (article_id, category_id) values (p_article_id,
exception
    when dup_val_on_index then null;
end add_category_to_article;
/

create or replace procedure delete_article_complete(
    p_article_id in number
) is
begin
    delete from article_tags where article_id = p_article_id;
    delete from article_categories where article_id = p_article_id;
    delete from comments where article_id = p_article_id;
    delete from articles where id = p_article_id;
    commit;
end delete_article_complete;
/

create or replace function count_comments(p_article_id in number)
return number is
    v_count number;
begin
    select count(*) into v_count from comments where article_id = p_article_id;
    return v_count;
end count_comments;
/

create or replace function get_articles return sys_refcursor is
    v_cursor sys_refcursor;
begin
    open v_cursor for
select a.id, a.title, a.article_date, u.name as author
from articles a, users u
where a.user_id = u.id
order by a.article_date desc;
    return v_cursor;
end get_articles;
/

```



```
Function REGISTER_USER compilado

Function CREATE_TAG compilado

Function CREATE_CATEGORY compilado

Function ADD_ARTICLE compilado

Procedure ADD_COMMENT compilado

Procedure ADD_TAG_TO_ARTICLE compilado

Procedure ADD_CATEGORY_TO_ARTICLE compilado

Procedure DELETE_ARTICLE_COMPLETE compilado

Function COUNT_COMMENTS compilado

Function GET_ARTICLES compilado
```

MANUAL DE INSTALACIÓN Y USO

- Requisitos previos

- Python 3.8+
- Oracle SQL developer
- Librerías: flask, oracledb, flask-cors

- Configuración de la base de datos

1. Descargar la aplicación desde el repositorio:
2. En Oracle SQL developer hacer crear un usuario: BLOG1/blog1

3. Abrir los scripts y ejecutar en orden, siendo el primero BLOG1 para la creación de tablas y en segundo BLOG~2 para datos de prueba

- Configuración de aplicación

En app.py - Configurar conexión

```
dsn = oracledb.makedsn("localhost", 1521, service_name="xepdb1")
```

```
conn = oracledb.connect(user="BLOG1", password="blog1", dsn=dsn)
```

- Ejecución

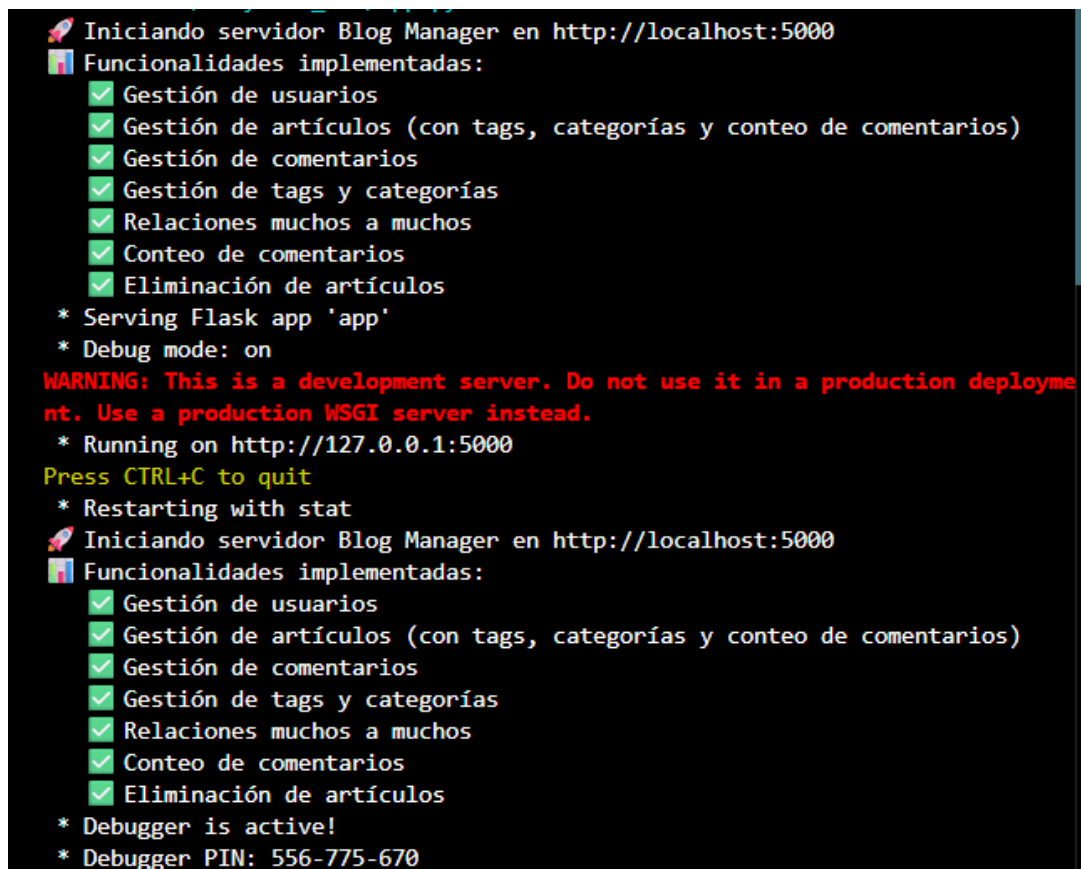
- Instalar dependencias

```
pip install flask oracledb flask-cors
```

- Ejecutar aplicación

```
python app.py
```

El servidor estará en: <http://localhost:5000>



```
🚀 Iniciando servidor Blog Manager en http://localhost:5000
📁 Funcionalidades implementadas:
  ✔ Gestión de usuarios
  ✔ Gestión de artículos (con tags, categorías y conteo de comentarios)
  ✔ Gestión de comentarios
  ✔ Gestión de tags y categorías
  ✔ Relaciones muchos a muchos
  ✔ Conteo de comentarios
  ✔ Eliminación de artículos
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
🚀 Iniciando servidor Blog Manager en http://localhost:5000
📁 Funcionalidades implementadas:
  ✔ Gestión de usuarios
  ✔ Gestión de artículos (con tags, categorías y conteo de comentarios)
  ✔ Gestión de comentarios
  ✔ Gestión de tags y categorías
  ✔ Relaciones muchos a muchos
  ✔ Conteo de comentarios
  ✔ Eliminación de artículos
* Debugger is active!
* Debugger PIN: 556-775-670
```

Ejecucion de Frontend

Para ejecutar el frontend que se encuentra en HTML se necesita ir a una nueva terminal donde se colocara la dirección de la carpeta en la que se encuentra guardado el archivo consecutivamente se colocara el siguiente comando:

- `python -m http.server 8000`
- Abrimos el navegador y accedemos a: <http://localhost:8000>

Funcionalidades Disponibles en la Interfaz:

- Visualizar todos los artículos del blog
- Crear nuevos artículos con título y contenido
- Gestionar usuarios del sistema
- Ver estadísticas de comentarios por artículo
- Navegar por la estructura completa del blog

88



Gestión de Usuarios

Nuevo Usuario

Nombre completo:

Correo electrónico:

 Guardar Usuario

Usuarios Registrados

 **Ana García**
 ana.garcia@email.com
ID: 1

 **Carlos López**
 carlos.lopez@email.com
ID: 2

 **María Rodríguez**
 maria.rodriguez@email.com
ID: 3

Gestión de Artículos

Nuevo Artículo

Título:

Contenido:

Usuario ID (autor):

 **Publicar**

Asignar Relaciones

Asignar Tag a Artículo:

Artículo ID

Tag ID

 **Asignar Tag**

Asignar Categoría a
Artículo:

Artículo ID

Categoría ID

 **Asignar Categoría**

Eliminar Artículo

ID del artículo a eliminar:

 **Eliminar**

Artículos Publicados

Cómo Mantener una Rutina de Ejercicios

 Pedro Martínez •  15/02/2024

Etiquetas: Deportes Salud

Categorías: Deportes Salud

Comentarios: 1

ID: 5



Blog Manager

Sistema de Gestión de Contenidos

 Usuarios

 Artículos

 Comentarios

 Tags

 Configuración

Comentarios del Blog

Nuevo Comentario

Artículo ID:

Usuario ID:


1

Comentario:

 Agregar Comentario


Todos los Comentarios

"Justo lo que necesitaba para mantener mi rutina de ejercicios. Muy motivador."

 Javier López en "Cómo Mantener una Rutina de Ejercicios"

 16/02/2024 00:00

"Los consejos sobre porciones fueron muy útiles. Estoy mejorando mis hábitos alimenticios."

 Carmen Ruiz en "Consejos para una Alimentación Balanceada"

 11/02/2024 00:00



Blog Manager

Sistema de Gestión de Contenidos

 Usuarios

 Artículos

 Comentarios

 Tags

 Configuración

Gestión de Tags y Categorías

Tags Existentes

Cocina

 cocina


ID: 2

Deportes

 deportes

ID: 3


Música

 musica

ID: 5


Categorías Existentes

Cocina

 cocina

ID: 2

Deportes

 deportes

ID: 3

Entretenimiento

 entretenimiento

ID: 5

Blog Manager


Sistema de Gestión de Contenidos

 Usuarios

 Artículos

 Comentarios

 Tags

 Configuración

Configuración del Blog

Nuevo Tag

Nombre del Tag:


URL (opcional):

 Crear Tag

Nueva Categoría

Nombre de Categoría:

URL (opcional):

 Crear Categoría

Conclusión

El Sistema de Gestión de Blog se implementó exitosamente utilizando una arquitectura de tres capas que integra Python Flask para la API REST y Oracle PL/SQL para la lógica de negocio.

Logros Principales:

- 6 funciones y 4 procedimientos PL/SQL implementados
- Gestión integral de usuarios, artículos, comentarios, tags y categorías

Resultado Final:

Se entregó un sistema robusto y funcional que demuestra las mejores prácticas en desarrollo web moderno, con capacidad de expansión para nuevas funcionalidades y optimizaciones futuras.

El proyecto cumple todos los objetivos establecidos y representa una base sólida para sistemas de gestión de contenidos empresariales.