Universidad Internacional San Isidro Labrador Facultad de Ingeniería de Sistemas
Carrera: Ingeniería de Sistemas Curso: Programación Avanzada III Cuatrimestre 2025

PROYECTO 1 – PRE-DISEÑO VISUAL DEL SISTEMA

# "TASKFLOW - GESTOR INTELIGENTE DE TAREAS"

DOCENTE:
Estefanía Boza Villalobos

ESTUDIANTE(S):
Brayan Palacios García Cedula: 207510970

FECHA DE ENTREGA: 19/10/2025

# INTRODUCCIÓN

El presente documento corresponde al Pre-diseño Visual del Sistema "TaskFlow", desarrollado como parte del Proyecto 1 del curso [Nombre del curso].

TaskFlow es un gestor inteligente de tareas diseñado para ayudar a los usuarios a organizar su tiempo de manera eficiente, reduciendo el estrés y aumentando la productividad mediante una interfaz intuitiva y funcional.

El pre-diseño visual presentado a continuación sirve como guía preliminar para el equipo de desarrollo y stakeholders, estableciendo los lineamientos de estructura, navegación y apariencia general del sistema antes de proceder con la implementación técnica.

# ESTRUCTURA DEL SISTEMA

## PRE-DISEÑO VISUAL

**"TaskFlow" - Gestor Inteligente de Tareas**

**Slogan:**

*"Organiza tu tiempo de manera inteligente y sin estrés"*

**Estructura de Pantallas (Wireframes)**

### a) Login

- **Cabecera:** Logo + Nombre "TaskFlow"
- **Formulario centralizado:**
    - Campo "Usuario o Email"
    - Campo "Contraseña" (con icono de ojo para mostrar/ocultar)
    - Botón "Iniciar Sesión"
    - Enlace "¿Olvidaste tu contraseña?"
- **Pie:** "¿No tienes cuenta? **Regístrate aquí**"

### b) Panel Principal

- **Barra superior:** Logo, menú usuario (avatar + nombre), botón notificaciones, botón cerrar sesión.

- **Sidebar izquierdo:** Navegación con íconos:
    - Inicio
    - Mis Tareas
    - Proyectos
    - Calendario
    - Reportes
    - Configuración
- **Área central:**
    - Resumen rápido: "Tareas de hoy", "Próximas pendientes", "Completadas esta semana".
    - Lista de tareas recientes con checkboxes, etiquetas de prioridad y fechas.

### c) Formulario de Registro

- Campos:
    - Nombre completo
    - Email
    - Contraseña
    - Confirmar contraseña
    - Botón "Registrarse"
    - Enlace "¿Ya tienes cuenta? Inicia sesión"

### d) Menú de Navegación (Ejemplo: Mis Tareas)

- Filtros rápidos: "Hoy", "Esta semana", "Importante", "Completadas".
- Barra búsqueda.
- Botón "+ Nueva Tarea".

- Lista de tareas con:
    - Checkbox
    - Título
    - Fecha vencimiento
    - Prioridad (color)
    - Proyecto asignado
    - Opciones (editar, eliminar)

**e) Reportes**

- Gráfico de barras: "Tareas completadas por día"
- Gráfico circular: "Distribución por prioridad"
- Listado: "Productividad semanal"

**Paleta de colores visual**

---

- TURQUESA PRINCIPAL: [2EC4B6]
- TURQUESA CLARO: [CBF3F0]
- GRIS OSCURO: [2D3A3A]
- GRIS MEDIO: [7F8C8D]
- GRIS CLARO: [F5F7F7]
- ACENTO/URGENTE: [FF6B6B]

---

**Iconografía utilizada**

- TAREAS: ✓ [CheckSquare]
- PROYECTOS: 🗁 [Folder]
- CALENDARIO: 📅 [Calendar]
- REPORTES: 📊 [BarChart3]
- CONFIGURACIÓN: ⚙ [Settings]
- NOTIFICACIONES: 🔔 [Bell]
- USUARIO: 👤 [User]

  BÚSQUEDA: 🔍 [Search]

**Tipografía**

- **Fuente Principal:** Segoe UI (moderna, legible)
- **Tamaños:**
  - Títulos: 1.5rem - 2rem
  - Texto normal: 1rem
  - Texto pequeño: 0.875rem

**Elementos Visuales Adicionales**

- **Botones:**
  - Primario: Turquesa, texto blanco, bordes redondeados.
  - Secundario: Gris claro, texto gris oscuro.

- **Tarjetas (cards):** Sombra suave, fondo blanco, bordes redondeados.

**Prioridades:**

  - Alta: Rojo (#FF6B6B)

- Media: Naranja (#FF9E64)
- Baja: Turquesa (#2EC4B6)

**Diagrama de Navegación Principal**

Descripción de Módulos

- Login: Autenticación de usuarios
- Panel Principal: Vista resumen con métricas rápidas
- Mis Tareas: Gestión completa de tareas individuales
- Proyectos: Organización de tareas por proyectos
- Calendario: Visualización temporal de actividades
- Reportes: Análisis de productividad y métricas
- Configuración: Personalización del sistema

# WIREFRAMES

## Pantalla de Login

| |
|---|
| **[Logo TaskFlow]** |
| |
| **Usuario o Email** |
| |
| **Contraseña [👁]** |
| |
| **Iniciar Sesión** |
| |
| **¿Olvidaste tu contraseña?** |
| |
| **¿No tienes cuenta?**<br>**Regístrate \|** |
| |

## WIREFRAME - PANEL PRINCIPAL

| [Logo] TaskFlow | [🔔] | [👤 María] | [Salir] |
|---|---|---|---|
| [🏠] Inicio | 📅 Hoy: 5 tareas | | |
| [√] Tareas | ⭐ Importante: 2 | | |
| [📁] Proyectos | ✅ Completadas: 3 | | |
| [📅] Calendario | | | |
| [📊] Reportes | | | |
| [⚙] Config | \| ✓ Revisar informe \| | | |
| Reunión equipo | | | |
| Llamar cliente | | | |
| Entregar diseño | | | |

**WIREFRAME - FORMULARIO DE REGISTRO**

| |
|---|
| **[Logo TaskFlow]** |
| **Nombre completo** |
| |
| **Email** |
| |
| **Contraseña** |
| |
| **Confirmar contraseña** |
| |
| **[ REGISTRARSE ]** |
| **¿Ya tienes cuenta? Inicia sesión** |

## WIREFRAME - SECCIÓN "MIS TAREAS"

| | | | |
|---|---|---|---|
| [Logo] TaskFlow | | [👤] | [Salir] |
| 🔍 Buscar tareas... | | | |
| [🏠] Inicio | | | |
| [√] Tareas | Filtros: | [Hoy] | [Semana] |
| [📁] Proyectos | [Importante] | [Todas] | |
| [📅] Calendario | | | |
| [📊] Reportes | [+ Nueva Tarea] | | |
| [⚙] Config | | | |
| **Diseñar wireframes** | | | |
| 🕐 Hoy | → | 🔴 Alta | |
| **Revisar documentación** | | | |
| 📅 28 Oct | → | 🟠 Media | |
| **✓ Completar informe** | | | |
| 📅 30 Oct | → | ✅ Hecha | |

## WIREFRAME - SECCIÓN DE REPORTES

| | | | |
|---|---|---|---|
| [Logo] TaskFlow | | [👤] | [Salir] |
| 📊 Reportes de Productividad | | | |
| [🏠] Inicio | | | |
| [√] Tareas | Filtros: | [Hoy] | [Semana] |
| [🗀 ] Proyectos | [Importante] | [Todas] | |
| [📅] Calendario | | | |
| [📊] Reportes | [+ Nueva Tarea] | | |
| [⚙] Config | | | |

| Productividad | semanal: |
|---|---|
| Lunes | 8 tareas ✅ |
| Martes | 6 tareas ✅ |
| Miércoles | 7 tareas ✅ |

# DEFINICIÓN DE LA BASE DE DATOS

## Tipo de Base de Datos

**Base de datos relacional**:

Se eligió este tipo debido a la naturaleza estructurada de los datos y las relaciones definidas entre entidades como usuarios, proyectos y tareas.

**Propósito dentro del Sistema**

La base de datos tiene como propósito principal:

- Almacenar información de usuarios registrados
- Gestionar proyectos y tareas asociadas
- Mantener el estado y progreso de cada tarea
- Clasificar tareas por categorías y prioridades
- Facilitar consultas y reportes del sistema

**Motor de Base de Datos Elegido**

**MySQL 8.0**

Justificación:

- Software de código abierto y gratuito
- Amplia documentación y comunidad de soporte
- Compatibilidad con frameworks web modernos
- Buen desempeño para aplicaciones de tamaño medio
- Soporte para transacciones ACID

# ELEMENTOS DE DISEÑO

**Tipos de Datos por Campo**

Tabla: Usuarios

| Campo | Tipo | Longitud | Restricciones | Descripción |
|---|---|---|---|---|
| id | INT | - | PRIMARY KEY, AUTO_INCREMENT | Identificador único |
| nombre | VARCHAR | 100 | NOT NULL | Nombre completo del usuario |
| email | VARCHAR | 150 | UNIQUE, NOT NULL | Correo electrónico del usuario |
| password_hash | VARCHAR | 255 | NOT NULL | Contraseña cifrada o encriptada |
| fecha_creacion | DATETIME | - | DEFAULT CURRENT_TIMESTAMP | Fecha de registro en el sistema |

Tabla: Proyectos

| Campo | Tipo | Longitud | Restricciones | Descripción |
|---|---|---|---|---|
| id | INT | - | PRIMARY KEY, AUTO_INCREMENT | Identificador único |
| nombre | VARCHAR | 100 | NOT NULL | Nombre del proyecto |
| descripcion | TEXT | - | NULL | Descripción general o detallada del proyecto |
| usuario_id | INT | - | FOREIGN KEY | Referencia al usuario creador |
| fecha_creacion | DATETIME | - | DEFAULT CURRENT_TIMESTAMP | Fecha de creación del proyecto |

Tabla: Tareas (Principal)

| Campo | Tipo | Longitud | Restricciones | Descripción |
|---|---|---|---|---|
| id | INT | - | PRIMARY KEY, AUTO_INCREMENT | Identificador único |
| titulo | VARCHAR | 200 | NOT NULL | Título de la tarea |
| descripcion | TEXT | - | NULL | Descripción detallada de la tarea |
| fecha_creacion | DATETIME | - | DEFAULT CURRENT_TIMESTAMP | Fecha de creación |
| fecha_vencimiento | DATE | - | NULL | Fecha límite de cumplimiento |
| prioridad | ENUM | - | ('Baja', 'Media', 'Alta') | Nivel de prioridad asignada |

| | | | | |
|---|---|---|---|---|
| usuario_id | INT | - | FOREIGN KEY, NOT NULL | Usuario responsable de la tarea |
| proyecto_id | INT | - | FOREIGN KEY, NULL | Proyecto al que pertenece la tarea |
| categoria_id | INT | - | FOREIGN KEY, NULL | Categoría o clasificación de la tarea |
| estado_id | INT | - | FOREIGN KEY, DEFAULT 1 | Estado actual de la tarea (pendiente, en proceso, completada, etc.) |

**Llaves Primarias y Foráneas**

Llaves Primarias

- **Usuarios:** `id` (INT AUTO_INCREMENT)
- **Proyectos:** `id` (INT AUTO_INCREMENT)
- **Tareas:** `id` (INT AUTO_INCREMENT)
- **Categorías:** `id` (INT AUTO_INCREMENT)
- **Estados:** `id` (INT AUTO_INCREMENT)

Llaves Foráneas y Relaciones sql

-- Relación: Usuarios → Proyectos ALTER

TABLE Proyectos

ADD FOREIGN KEY (usuario_id) REFERENCES Usuarios(id) ON

DELETE CASCADE;

-- Relación: Usuarios → Tareas ALTER

TABLE Tareas

ADD FOREIGN KEY (usuario_id) REFERENCES Usuarios(id) ON

DELETE CASCADE;

-- Relación: Proyectos → Tareas

```
ALTER TABLE Tareas
ADD FOREIGN KEY (proyecto_id) REFERENCES Proyectos(id) ON
DELETE SET NULL;

-- Relación: Categorías → Tareas ALTER
TABLE Tareas
ADD FOREIGN KEY (categoria_id) REFERENCES Categorias(id) ON
DELETE SET NULL;

-- Relación: Estados → Tareas ALTER
TABLE Tareas
ADD FOREIGN KEY (estado_id) REFERENCES Estados(id);
```

## Normalización hasta Tercera Forma Normal (3FN)

Primera Forma Normal (1FN)

- ✓ Todos los campos contienen valores atómicos
- ✓ No hay grupos repetitivos de datos
- ✓ Cada tabla tiene una clave primaria única
- ✓ Los datos se almacenan sin repeticiones innecesarias

Segunda Forma Normal (2FN)

- ✓ Cumple con 1FN

- ✓ Todos los atributos no clave dependen completamente de la clave primaria
- ✓ Ejemplo: titulo, descripcion en Tareas dependen completamente de id
- ✓ No hay dependencias parciales

Tercera Forma Normal (3FN)

- ✓ Cumple con 2FN
- ✓ No hay dependencias transitivas entre atributos no clave
- ✓ Los datos de categoría, estado y usuario están en tablas separadas
- ✓ Eliminación de redundancias mediante relaciones

Normalización aplicada:

Los nombres de categorías no se repiten en cada tarea, sino que se referencian mediante categoria_id

Los estados del sistema están centralizados en una tabla para consistencia

## Funciones, Vistas y Triggers

Vista: Tareas Próximas a Vencer

```
CREATE VIEW tareas_proximas_vencer AS SELECT
    t.id, t.titulo,
    t.fecha_vencimiento, u.nombre as
    usuario, p.nombre as proyecto,
    DATEDIFF(t.fecha_vencimiento, CURDATE()) as dias_restantes FROM
Tareas t
```

```
JOIN Usuarios u ON t.usuario_id = u.id

LEFT JOIN Proyectos p ON t.proyecto_id = p.id

WHERE t.fecha_vencimiento BETWEEN CURDATE() AND DATE_ADD(CURDATE(),
INTERVAL 7 DAY)

AND t.estado_id != (SELECT id FROM Estados WHERE nombre = 'Completado') ORDER
BY t.fecha_vencimiento ASC;
```

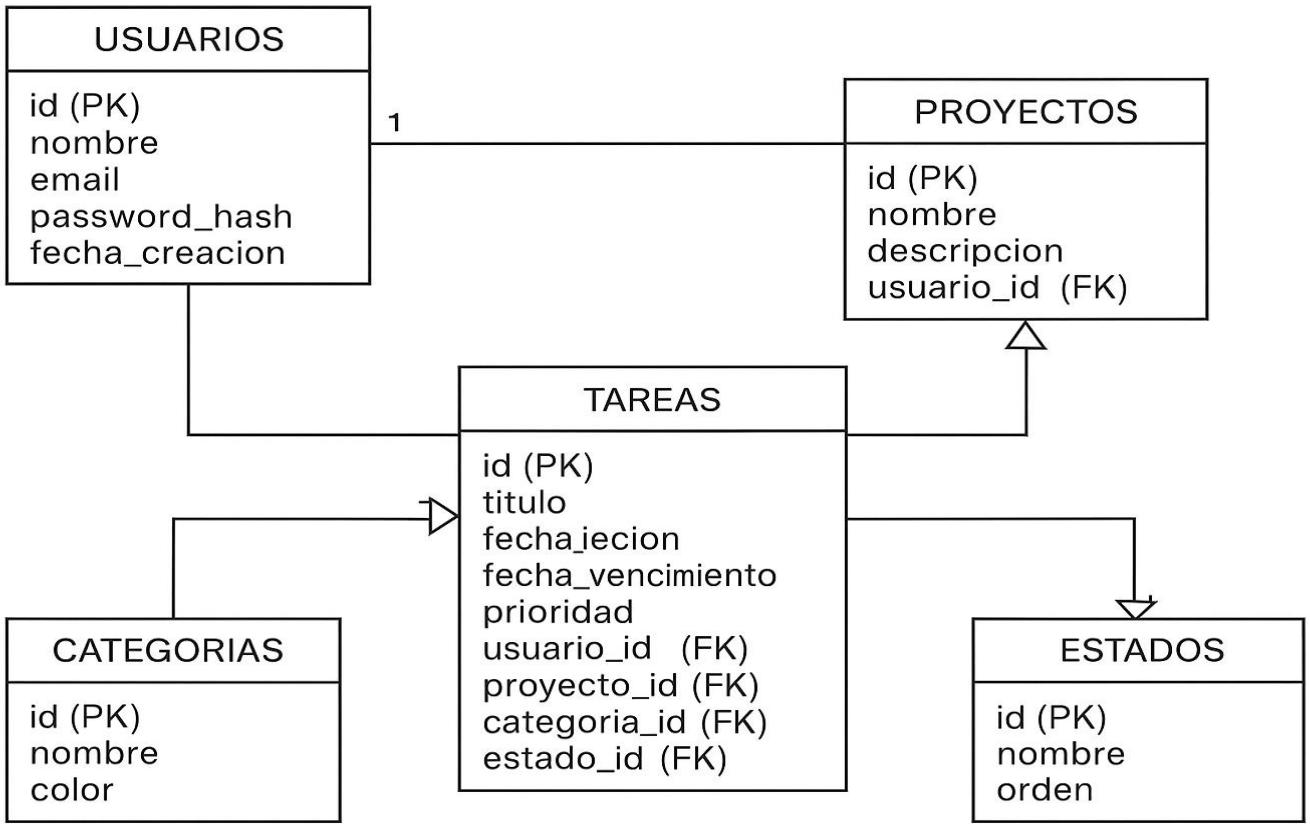
**Trigger: Auditoría de Cambios**

```
DELIMITER //

CREATE TRIGGER auditoria_cambios_tarea AFTER
    UPDATE ON Tareas
    FOR EACH ROW BEGIN
    IF OLD.estado_id != NEW.estado_id THEN

        INSERT INTO Auditoria (tarea_id, campo, valor_anterior, valor_nuevo, fecha_cambio)

        VALUES (NEW.id, 'estado_id', OLD.estado_id, NEW.estado_id, NOW()); END IF;
END// DELIMITER ;
```

# ESQUEMA Y DIAGRAMA

Diagrama Entidad-Relación (ERD)

SISTEMA DE GESTIÓN DE TAREAS - DIAGRAMA ERD



## Descripción de Entidades y Relaciones

**Entidad: Usuarios**

- **Descripción:** Almacena la información de los usuarios del sistema
- **Atributos clave:** id, nombre, email, password_hash
- **Relaciones:** Un usuario puede crear múltiples proyectos y tareas

**Entidad: Proyectos**

- **Descripción:** Agrupa tareas relacionadas bajo un objetivo común
- **Atributos clave:** id, nombre, descripcion, usuario_id
- **Relaciones:** Pertenece a un usuario, contiene múltiples tareas

**Entidad: Tareas**

- **Descripción:** Representa las actividades individuales a realizar
- **Atributos clave:** id, titulo, fecha_vencimiento, prioridad
- **Relaciones:** Relacionada con usuario, proyecto, categoría y estado

**Entidad: Categorías**

- **Descripción:** Clasifica las tareas por tipo o área
- **Atributos clave:** id, nombre, color
- **Relaciones:** Una categoría puede aplicarse a múltiples tareas

**Entidad: Estados**

- **Descripción:** Define el flujo de trabajo de las tareas
- **Atributos clave:** id, nombre, orden
- **Relaciones:** Un estado puede aplicarse a múltiples tareas

# CONEXIÓN CON LA APLICACIÓN

**1. Tecnología de Conexión**

**Flask SQLAlchemy** - ORM (Object-Relational Mapping) para Python

**Justificación de la elección:**

- Facilita la interacción con la base de datos mediante objetos Python
- Proporciona abstracción del motor de base de datos
- Ofrece seguridad contra inyecciones SQL
- Permite migraciones de esquema sencillas

**2. Configuración de Conexión Código:**

**Configuración Principal**

```
# config.py - Configuración de la base de datos import os
from flask import Flask
from flask_sqlalchemy import SQLAlchemy from
flask_migrate import Migrate


app = Flask(__name__)

# Configuración de la base de datos
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql+pymysql://usuario:password@localhost/gestion_tareas'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SECRET_KEY'] = os.environ.get('SECRET_KEY', 'clave-secreta-
predeterminada')


db = SQLAlchemy(app) migrate =
Migrate(app, db)
```

### 3. Código: Modelos SQLAlchemy

```
# models.py - Definición de modelos
from datetime import datetime from
config import db
class Usuario(db.Model):
    __tablename___= 'Usuarios'
    id = db.Column(db.Integer, primary_key=True) nombre =
    db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(150), unique=True, nullable=False) password_hash =
    db.Column(db.String(255), nullable=False) fecha_creacion =
    db.Column(db.DateTime, default=datetime.utcnow)

    # Relaciones
    proyectos = db.relationship('Proyecto', backref='usuario', lazy=True, cascade='all, delete-
orphan')
    tareas = db.relationship('Tarea', backref='usuario', lazy=True, cascade='all, delete-
orphan')

class Proyecto(db.Model):
    __tablename___= 'Proyectos'
    id = db.Column(db.Integer, primary_key=True) nombre =
    db.Column(db.String(100), nullable=False) descripcion =
    db.Column(db.Text)
    usuario_id = db.Column(db.Integer, db.ForeignKey('Usuarios.id'), nullable=False)
```

```
    fecha_creacion = db.Column(db.DateTime, default=datetime.utcnow)


    # Relaciones

    tareas = db.relationship('Tarea', backref='proyecto', lazy=True)
class Tarea(db.Model):


    __tablename___= 'Tareas'

    id = db.Column(db.Integer, primary_key=True) titulo =
    db.Column(db.String(200), nullable=False) descripcion =
    db.Column(db.Text)
    fecha_creacion = db.Column(db.DateTime, default=datetime.utcnow) fecha_vencimiento
    = db.Column(db.Date)
    prioridad = db.Column(db.Enum('Baja', 'Media', 'Alta'), default='Media')
    # Llaves foráneas

    usuario_id = db.Column(db.Integer, db.ForeignKey('Usuarios.id'), nullable=False)

    proyecto_id = db.Column(db.Integer, db.ForeignKey('Proyectos.id')) categoria_id =
    db.Column(db.Integer, db.ForeignKey('Categorias.id')) estado_id =
    db.Column(db.Integer, db.ForeignKey('Estados.id'), default=1)
    # Relaciones

    categoria = db.relationship('Categoria', backref='tareas') estado =
    db.relationship('Estado', backref='tareas')
class Categoria(db.Model):

    __tablename___= 'Categorias'

    id = db.Column(db.Integer, primary_key=True) nombre =
    db.Column(db.String(50), nullable=False) color =
    db.Column(db.String(7), default='#007bff')
```

```
class Estado(db.Model):
    __tablename___= 'Estados'
    id = db.Column(db.Integer, primary_key=True) nombre =
    db.Column(db.String(30), nullable=False)
    orden = db.Column(db.Integer, default=0)
```

**4. Código: Ejemplo de Uso en Rutas**

```
# routes.py - Ejemplos de operaciones con la base de datos from flask
import render_template, request, jsonify
from models import db, Usuario, Tarea, Proyecto

@app.route('/tareas') def
listar_tareas():
    usuario_id = 1  # En una app real, esto vendría de la sesión tareas =
    Tarea.query.filter_by(usuario_id=usuario_id).all() return
    render_template('tareas.html', tareas=tareas)


@app.route('/tarea/nueva', methods=['POST']) def
crear_tarea():
    try:
        nueva_tarea = Tarea( titulo=request.form['titulo'],
            descripcion=request.form['descripcion'],
            fecha_vencimiento=request.form['fecha_vencimiento'],
            prioridad=request.form['prioridad'],
            usuario_id=1, proyecto_id=request.form.get('proyecto_id'),
            categoria_id=request.form.get('categoria_id')
```

```
        )
        db.session.add(nueva_tarea)
        db.session.commit()
        return jsonify({'mensaje': 'Tarea creada exitosamente'}), 201 except
    Exception as e:
        db.session.rollback()
        return jsonify({'error': str(e)}), 400

@app.route('/api/tareas/proximas-vencer') def
tareas_proximas_vencer():
    from sqlalchemy import text query =
    text("""
        SELECT titulo, fecha_vencimiento, dias_restantes FROM
        tareas_proximas_vencer
        WHERE usuario_id = :usuario_id """)
    resultado = db.session.execute(query, {'usuario_id': 1}) tareas =
    [dict(row) for row in resultado]
    return jsonify(tareas)
```

**5. Evidencia de Conexión Funcional**

**Prueba de Conexión**

```
# test_connection.py - Script de prueba from config
import app, db

def test_database_connection(): try:
        with app.app_context():
            db.session.execute('SELECT 1')
```

```
            print("✅ Conexión a la base de datos exitosa") return True
        except Exception as e:
            print(f"❌ Error de conexión: {e}") return False


if __name__ == '__main__':
    test_database_connection()
```

# DESARROLLO DEL BACKEND

## Arquitectura seleccionada: Arquitectura por Capas (MVC)

**Justificación:**

TaskFlow es un sistema de gestión de tareas de complejidad media, que requiere una clara separación entre la lógica de negocio, la presentación y los datos. La arquitectura por capas (Modelo-Vista-Controlador) facilita el mantenimiento, la escalabilidad y la colaboración entre desarrolladores. Además, se integra de forma natural con el frontend ya diseñado en las pantallas de login, registro, tareas y reportes.

**Estructura y Modularidad del Código**

Estructura de carpetas del backend:

```
/src
  /controllers     # Manejo de solicitudes HTTP
  /models          # Modelos de datos (Usuario, Tarea, Proyecto)
  /services        # Lógica de negocio
  /middlewares     # Autenticación, validaciones, logs
  /routes          # Definición de endpoints
  /config          # Configuración de BD y entorno
  /utils           # Helpers (encriptación, fechas)
```

/tests # Pruebas unitarias y de integración

**Descripción de componentes:**

- **Controllers:** authController.js, taskController.js, projectController.js, reportC ontroller.js
- **Models:** User.js, Task.js, Project.js
- **Services:** userService.js, taskService.js, reportService.js
- **Middlewares:** authMiddleware.js, validationMiddleware.js, errorHandler.js

**Diagrama de flujo de interacción:**

Cliente → Routes → Middlewares (auth/validation) → Controllers → Services → Models → BD (MySQL/MongoDB)

**Seguridad**

Medidas implementadas:

- Autenticación mediante JWT (JSON Web Tokens).
- Cifrado de contraseñas con bcrypt.
- Validación de entrada con express-validator.
- Protección de rutas con middlewares de autorización por rol.
- Uso de variables de entorno para claves sensibles.

**Gestión de Errores y Pruebas**

Manejo de errores:

- Middleware centralizado de errores.

- Logs con **morgan** y **winston**.
- Respuestas JSON estandarizadas para errores (ej: { error: "Mensaje" }).

Pruebas realizadas:

- Pruebas unitarias con **Jest** para servicios y modelos.
- Pruebas de integración para endpoints de autenticación y gestión de tareas.

Ejemplo de prueba unitaria (TaskService):

```
test("Debería crear una tarea con prioridad alta", async () => { const task =
  await taskService.createTask({
    title: "Revisar informe", priority:
    "alta", dueDate: "2025-11-01"
  });
  expect(task.priority).toBe("alta");
});
```

## Conexión con Base de Datos y Frontend

Conexión a la base de datos:

- Uso de **MongoDB** con Mongoose (o MySQL con Sequelize según elección).
- Modelos definidos para Usuario, Tarea y Proyecto.

Endpoints implementados (ejemplos):

- POST /auth/register → Registro de usuario
- POST /auth/login → Inicio de sesión
- GET /tasks → Listar tareas del usuario
- POST /tasks → Crear nueva tarea

- PUT /tasks/:id → Actualizar tarea
- GET /reports/weekly → Reporte semanal de productividad Ejemplo de conexión con frontend (desde React):

```
// Ejemplo: Obtener tareas
const response = await fetch("/api/tasks", { headers: {
  "Authorization": `Bearer ${token}` }
});
const tasks = await response.json();
```

# CONCLUSIONES

El desarrollo del proyecto “TaskFlow – Gestor Inteligente de Tareas” representa un esfuerzo integral que combina diseño visual, modelado de base de datos, arquitectura backend y conexión efectiva entre los distintos componentes del sistema. El trabajo realizado demuestra una comprensión sólida de los principios del desarrollo de software moderno, aplicando buenas prácticas de programación, organización modular y normalización de datos.

Desde su concepción, el pre-diseño visual proporcionó una base clara para la estructuración del sistema, priorizando la usabilidad, la simplicidad y la eficiencia del flujo de trabajo. Las interfaces propuestas, con una distribución lógica de elementos y un esquema de colores coherente, promueven una experiencia de usuario intuitiva, moderna y accesible, lo cual es esencial para un sistema orientado a la productividad personal y profesional.

En el ámbito de la base de datos, el modelo diseñado bajo MySQL 8.0 garantiza integridad, coherencia y rendimiento mediante una arquitectura relacional robusta. La aplicación de la tercera forma normal (3FN) y la definición de claves primarias, foráneas y restricciones de integridad aseguran una administración óptima de los datos, minimizando redundancias y errores. Además, la inclusión de vistas y triggers, como la auditoría de cambios y la vista de tareas próximas a vencer, fortalece la trazabilidad y el control de las operaciones dentro del sistema.

El backend, implementado bajo la arquitectura MVC (Modelo–Vista–Controlador) y con soporte de Flask SQLAlchemy, ofrece una estructura modular, mantenible y segura. La

separación de responsabilidades entre controladores, modelos y servicios facilita el escalamiento del sistema y permite incorporar nuevas funcionalidades —como recordatorios automáticos, integración con calendarios externos o generación de reportes avanzados— sin alterar la estabilidad general del código.

En cuanto a la seguridad y confiabilidad, el sistema incorpora medidas esenciales como el cifrado de contraseñas, autenticación con JWT, validaciones de entrada y gestión centralizada de errores. Estas estrategias consolidan a TaskFlow como una solución profesional y alineada con los estándares modernos de desarrollo seguro.