

UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

FACULTAD DE INGENIERÍA DE SISTEMAS



"PAPABOT: PLATAFORMA INTELIGENTE DE RECONOCIMIENTO Y CONSULTA DE VARIEDADES DE PAPA"

ASIGNATURA:
DESARROLLO DE APLICACIONES WEB

DOCENTE:
ING. JAIME SUASNABAR TERREL

INTEGRANTES:

- FERNANDEZ ROJAS, GABRIEL
- ISIDRO YARANGA, PIERO ALESSANDRO
- LAIZA SOTACURO, JENIFER FIORELLA
- MACURI LOPEZ, MAYRA SHADAY
- TAFUR HUAYTA, MARJORIE

HUANCAYO – PERÚ

2025

PAPABOT: PLATAFORMA INTELIGENTE DE RECONOCIMIENTO Y CONSULTA DE VARIEDADES DE PAPA

1. Resumen

El presente proyecto consiste en el desarrollo de una aplicación web inteligente orientada al reconocimiento y análisis de texto e imágenes, implementada con tecnologías modernas como Python (Flask) en el backend, React en el frontend y MySQL como sistema de base de datos relacional.

La aplicación permite a los usuarios registrarse, iniciar sesión de forma segura mediante JWT y acceder a funcionalidades protegidas como el análisis inteligente de texto y el reconocimiento de imágenes de papa. Para esto, se integran modelos de Inteligencia Artificial, entre ellos uno para análisis de texto (como análisis de sentimientos o resumen automático) y otro basado en ResNet50, entrenado para identificar especies de papa a partir de imágenes. Adicionalmente a esto, se incluye un dashboard administrativo que permite visualizar resultados de análisis de la cantidad de imágenes y textos subidos, variedad común de papas, un historial con el tipo de análisis, el resultado y la fecha en la que se realizó.

Toda la arquitectura se construye siguiendo el enfoque RESTful para la comunicación entre cliente y servidor, con especial atención en la seguridad, la protección de rutas y la experiencia del usuario en dispositivos móviles y de escritorio. Este desarrollo se organiza mediante la metodología SCRUM, distribuyendo roles como Product Owner, Scrum Master, Developers y QA Tester, gestionando el trabajo por sprints e historias de usuario priorizadas.

Como resultado final, se desarrolló una plataforma web inteligente caracterizada por su diseño moderno, responsive y modular. Esta solución integra tecnologías avanzadas tanto de frontend como de backend, junto con componentes de inteligencia artificial para el procesamiento inteligente de datos. Además, se aplicaron buenas prácticas de desarrollo ágil mediante metodologías colaborativas, lo que permitió construir un sistema robusto, seguro y escalable, la plataforma está orientada a brindar un servicio eficaz y accesible para el análisis automatizado e identificación precisa de información, optimizando la experiencia del usuario y facilitando la toma de decisiones.

2. Objetivos

Objetivo General

- Desarrollar una aplicación web inteligente, utilizando Python (Flask), MySQL y React, que permita el análisis automático de texto e imágenes mediante modelos de inteligencia artificial, con un enfoque en la seguridad, escalabilidad y experiencia de usuario.

Objetivos Específicos

- Implementar una API RESTful con autenticación segura utilizando JWT.
- Desarrollar componentes frontend en React que incluyan registro, inicio de sesión y módulos de análisis.
- Incorporar modelos de IA entrenados para análisis de texto y clasificación de imágenes de especies de papa.
- Diseñar un dashboard para visualizar los resultados procesados por el modelo de IA.
- Aplicar buenas prácticas de programación, documentación y diseño responsive.
- Coordinar el trabajo en equipo utilizando la metodología SCRUM y sus roles.

3. Equipo de Desarrollo SCRUM

Rol	Integrante	Responsabilidades
Product Owner	Mayra Macuri	Define los requisitos del sistema, prioriza tareas en el backlog y valida funcionalidades entregadas.
Scrum Master	Gabriel Fernández	Facilita las reuniones SCRUM, elimina bloqueos del equipo y asegura el cumplimiento de la metodología.
Developer – Backend	Piero Isidro	Desarrolla el backend con Flask y MySQL, implementa rutas de API, integración con modelo de IA.
Developer – Frontend	Fiorella Laiza	Crea los componentes de React: login, registro, análisis y visualización de resultados. Implementa diseño responsive.
QA Tester	Marjorie Tafur	Realiza pruebas funcionales de cada componente, verifica flujos del sistema, valida integración entre frontend y backend.

4. Backlog del Producto (Historias de Usuario)

ID	Historia de Usuario	Prioridad	Sprint
HU01	Como usuario, quiero registrarme para acceder al sistema	Alta	1
HU02	Como usuario, quiero iniciar sesión con seguridad (JWT)	Alta	1
HU03	Como usuario, quiero recuperar mi cuenta en caso de olvidar la contraseña	Alta	1
HU04	Como usuario autenticado, quiero visualizar una pantalla de bienvenida con mis datos	Media	2
HU05	Como usuario, quiero ingresar texto con características de la papa para obtener su especie mediante IA	Alta	2
HU06	Como usuario, quiero subir una imagen de papa y obtener su especie mediante el modelo entrenado (ResNet50)	Alta	2
HU07	Como usuario, quiero ver los resultados del análisis de imagen claramente organizados y explicados	Media	2
HU08	Como usuario autenticado, quiero ver mi perfil protegido por token	Media	2
HU09	Como administrador, quiero acceder a un dashboard con todos los resultados del análisis	Media	3
HU10	Como usuario, quiero acceder a un chatbot interactivo para ingresar datos fácilmente	Baja	3

5. Arquitectura General del Proyecto

El presente proyecto se ha desarrollado bajo una arquitectura cliente-servidor con división clara entre las capas de presentación, lógica de negocio, persistencia de datos e inteligencia artificial. Esta estructura modular permite un desarrollo colaborativo eficiente, mantenable y escalable, conforme a buenas prácticas de desarrollo ágil y los principios de la metodología SCRUM.

5.1. Frontend (Cliente)

Tecnologías: React.js, HTML5, CSS3, JavaScript ES6, TailwindCSS

Funcionalidades:

- Implementación de componentes dinámicos reutilizables para cada vista (registro, login, análisis por texto, análisis por imagen, dashboard).

- Manejo del estado y navegación entre rutas mediante React Router.
- Gestión del token JWT desde el almacenamiento local (localStorage) para el acceso a rutas protegidas.
- Estilos responsive mediante TailwindCSS, permitiendo una experiencia óptima tanto en móviles como en computadoras.
- Validación básica de formularios y retroalimentación al usuario sobre acciones realizadas (registro, análisis, etc.).

Interacción:

- Se comunica con el backend a través de peticiones HTTP (fetch o axios) hacia los endpoints definidos en la API RESTful.
- Recibe respuestas del modelo de inteligencia artificial y las presenta de manera clara al usuario final.

5.2. Backend (Servidor)

Tecnologías: Python 3, Flask, Flask-JWT-Extended, Flask-CORS, Flask-RESTful

Funcionalidades:

- Gestión completa de usuarios (registro, login con seguridad mediante JWT).
- Implementación de rutas protegidas por token para funcionalidades seguras como visualización de perfil y análisis de datos.
- Interacción con la base de datos MySQL para almacenar y recuperar datos de usuarios y análisis realizados.
- Lógica de negocio encargada de validar, procesar y canalizar los datos entre el frontend, el modelo de IA y la base de datos.
- Integración de modelos de aprendizaje automático (IA), que procesan tanto texto como imágenes, y retornan resultados precisos.

Rutas clave de la API:

- POST /register: Registro de usuarios con validación.
- POST /login: Inicio de sesión con generación de JWT.
- POST /analyze-text: Recibe texto y devuelve análisis IA.
- POST /analyze-image: Recibe imagen y devuelve clasificación con modelo entrenado.
- GET /profile: Ruta protegida que devuelve datos del usuario.
- GET /dashboard: Ruta protegida para observar los resultados del modelo.

5.3. Base de Datos

Sistema gestor: MySQL 8.x

Estructura:

- Tablas para usuarios, registros de análisis, logs de actividad, tokens inválidos y configuración del sistema.
- Uso de claves primarias y foráneas para asegurar la integridad referencial.
- Optimización de consultas mediante índices y uso adecuado de tipos de datos.

Conexión:

Flask se conecta a la base de datos utilizando una capa de abstracción personalizada o con ayuda de SQLAlchemy (opcional).

5.4. Módulo de Inteligencia Artificial

Modelos integrados:

5.4.1. Clasificación de imágenes de papa:

Se usó ResNet50 entrenado con un dataset de tipos de papas.

Proceso general:

- Se organiza el dataset de acuerdo a los tipos de papas que se tiene, en este caso son 4 tipos los cuales son: huayro, peruanita, blanca,yana-imilla.
- Se redimensiona las imágenes a un tamaño de 224 x 224 px

```
from PIL import Image
import os

# Ruta correcta del dataset
DATASET_DIR = 'tipo_papa/'
OUTPUT_DIR = 'tipo_papa_redimensionado/'
IMAGE_SIZE = (224, 224)

os.makedirs(OUTPUT_DIR, exist_ok=True)

for clase in os.listdir(DATASET_DIR):
    ruta_clase = os.path.join(DATASET_DIR, clase)
    ruta_salida = os.path.join(OUTPUT_DIR, clase)

    if os.path.isdir(ruta_clase):
        os.makedirs(ruta_salida, exist_ok=True)
        for i, archivo in enumerate(os.listdir(ruta_clase)):
            if archivo.lower().endswith('.jpg', '.jpeg', '.png'):
                ruta_imagen = os.path.join(ruta_clase, archivo)
                nueva_ruta = os.path.join(ruta_salida, f"{clase}_{i+1:03}.jpg")
                try:
                    img = Image.open(ruta_imagen).convert("RGB")
                    img = img.resize(IMAGE_SIZE)
                    img.save(nueva_ruta)
                    print(f"Guardado: {nueva_ruta}")
                except Exception as e:
                    print(f"Error al procesar {ruta_imagen}: {e}")
```

```

except Exception as e:
    print(f"Error en {ruta_imagen}: {e}")

```

Librerías: Pillow

- Para mejorar la calidad de las imágenes, se elimina el fondo

```

from rembg import remove
from PIL import Image
import os

# Ruta a las imágenes originales
INPUT_DIR = 'tipo_papa/'
# Ruta de salida donde se guardarán las imágenes sin fondo y con fondo
blanco
OUTPUT_DIR = 'tipo_papa_sinfondo/'

# Crea la carpeta de salida si no existe
os.makedirs(OUTPUT_DIR, exist_ok=True)

# Recorre todas las carpetas de clases dentro de tipo_papa/
for clase in os.listdir(INPUT_DIR):
    ruta_clase = os.path.join(INPUT_DIR, clase)
    ruta_salida = os.path.join(OUTPUT_DIR, clase)

    if os.path.isdir(ruta_clase):
        os.makedirs(ruta_salida, exist_ok=True)

    for i, archivo in enumerate(os.listdir(ruta_clase)):
        if archivo.lower().endswith('.jpg', '.jpeg', '.png'):
            ruta_imagen = os.path.join(ruta_clase, archivo)
            nueva_ruta = os.path.join(ruta_salida, f'{clase}_{i+1:03}.jpg')

            try:
                # Abre la imagen y remueve el fondo
                with Image.open(ruta_imagen) as img:
                    sin_fondo = remove(img)

                    # Crea una imagen con fondo blanco
                    fondo_blanco = Image.new("RGBA", sin_fondo.size, (255,
255, 255, 255))
                    resultado = Image.alpha_composite(fondo_blanco,
sin_fondo.convert("RGBA"))

                    # Convierte a RGB (sin canal alfa) y guarda como JPG
                    resultado.convert("RGB").save(nueva_ruta)
                    print(f"Guardado: {nueva_ruta}")

            except Exception as e:
                print(f"Error en {ruta_imagen}: {e}")

```

Librerías: rembg y Pillow

- En base al dataset, empezamos a entrenar el modelo

```

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras import layers, models, optimizers
import os

# Rutas
DATASET_DIR = 'tipos_final/'
IMG_SIZE = (224, 224)
BATCH_SIZE = 16
EPOCHS = 10

# Preprocesamiento de imágenes
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2, # 80% entrenamiento, 20% validación
    horizontal_flip=True,
    rotation_range=10,
    zoom_range=0.1
)

train_generator = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

# Cargar modelo base ResNet50 (sin top)
base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
base_model.trainable = False # Congelar pesos

# Construir modelo final
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(train_generator.num_classes, activation='softmax')
])

# Compilar

```

```

model.compile(
    optimizer=optimizers.Adam(),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Entrenar
history = model.fit(
    train_generator,
    epochs=EPOCHS,
    validation_data=val_generator
)

# Guardar modelo
model.save('modelo_resnet_papas.h5')
print(" Modelo guardado como modelo_resnet_papas.h5")

```

Librerías: Tensorflow y Keras

CNN(red neuronal convolucional) del tipo ResNet50

- Los datos (imagen) son enviados desde el frontend y backend.
- El backend procesa la información y pasa los datos al modelo correspondiente.
- El modelo devuelve un resultado (el tipo de papa) y se envía al cliente.

5.4.2 Clasificación de papa por características:

Se realizó una base de datos de las características de algunos tipos de papas.

Proceso general:

- Se organiza el dataset de acuerdo a los tipos de papas que se tiene, en este caso son 4 tipos los cuales son: huayro, peruanita, blanca,yana-imilla, se muestra las principales características de ellas.


```

pipeline.fit(X_train, y_train)

# Evaluación
y_pred = pipeline.predict(X_test)
report = classification_report(y_test, y_pred, output_dict=True)
report_df = pd.DataFrame(report).transpose()
report_df.to_excel(output_report_path)

# Guardar modelo
joblib.dump(pipeline, output_model_path)
print(f"Modelo guardado en: {output_model_path}")
print(f"Reporte guardado en: {output_report_path}")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Entrena un modelo de
clasiﬁcación de papas')
    parser.add_argument('--excel', required=True, help='Ruta del archivo
Excel con columnas descripción y variedad')
    parser.add_argument('--model', default='modelo_entrenado.joblib',
help='Ruta para guardar el modelo entrenado')
    parser.add_argument('--report', default='reporte_modelo.xlsx', help='Ruta
para guardar el reporte de evaluación')

    args = parser.parse_args()
    main(args.excel, args.model, args.report)

```

Librerías: Pandas, Scikit-Learn y Openpyxl

- Para realizarse la conexión con el backend se usó

```

from flask import Flask, request, jsonify
from flask_cors import CORS
import joblib

# Cargar el modelo
modelo = joblib.load("modelo.joblib")

# Crear app y habilitar CORS
app = Flask(__name__)
CORS(app, resources={r"/predict": {"origins": "*"}})

@app.route("/predict", methods=["POST"])
def predict():
    data = request.get_json()
    if not data or "descripción" not in data:
        return jsonify({"error": "Se requiere una descripción"}), 400

    descripción = data["descripción"]
    predicción = modelo.predict([descripción])[0]
    proba = modelo.predict_proba([descripción])
    clase_index = list(modelo.classes_).index(predicción)
    confianza = float(proba[0][clase_index])

```

```
return jsonify({
    "descripcion": descripcion,
    "variedad": prediction,
    "confianza": round(confidence, 4)
})

if __name__ == "__main__":
    app.run(debug=True)
```

Librerías: Flask

5.5. API RESTful

La API sirve como puente entre todas las capas del sistema. Es la encargada de facilitar una comunicación estructurada y segura entre el cliente y el servidor. Se siguen buenas prácticas REST como:

- Uso de verbos HTTP adecuados (GET, POST, etc.)
- Gestión de códigos de estado HTTP para respuestas exitosas o fallidas
- Rutas protegidas con autenticación basada en tokens JWT
- Validación robusta de datos recibidos

5.6. Seguridad

JWT (JSON Web Token): Utilizado para autenticar usuarios de manera segura y mantener la sesión sin necesidad de cookies.

CORS (Cross-Origin Resource Sharing): Configurado para permitir acceso únicamente desde dominios permitidos.

Hash de contraseñas: Las contraseñas de los usuarios se almacenan de forma encriptada (por ejemplo, con werkzeug.security o bcrypt).

Protección de rutas: Algunas rutas están restringidas a usuarios autenticados o a roles específicos como "admin".

6. Estructura y rutas del Backend

6.1. Estructura de carpetas

```
backend/
└── __pycache__/
    ├── .venv/
    └── database/
        └── __pycache__/
```

```

    └── db.py
    └── models/
        ├── __pycache__/
        ├── modelo_resnet_papas.h5
        ├── modelo_resnet50_papas.h5
        ├── modelo.joblib
        └── prediction_model.py
    └── routes/
        ├── __pycache__/
        ├── auth_routes.py
        ├── image_routes.py
        ├── profile_routes.py
        └── text_routes.py
    └── utils/
        ├── __pycache__/
        └── helpers.py
    └── app.py
    └── config.py
    └── requirements.txt

```

6.2. Rutas Backend

Método	Ruta	Descripción	Protegida
POST	/register	Registrar un nuevo usuario	No
POST	/login	Iniciar sesión, retorna token JWT	No
GET	/profile	Devuelve datos del perfil	Sí
POST	/analyze-text	Recibe texto y aplica el modelo IA	Sí
POST	/predict	Recibe una imagen de papa, predice la clase usando modelo ResNet y guarda	Si
POST	/forgot-password	Envia un código al correo electrónico registrado	No
POST	/reset-password	Permite cambiar la contraseña usando un código enviado por correo	No

GET	/admin/stats	Muestra estadísticas generales del sistema solo para usuarios con rol "admin"	Si
GET	/user/predictions	Muestra las predicciones del usuario autenticado	Si

7. Base de Datos (MySQL)

Creación de la base de datos

```
CREATE DATABASE IF NOT EXISTS papa_db;
USE papa_db;
```

Tabla Usuarios

```
-- Tabla de usuarios
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    full_name VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('user', 'admin') DEFAULT 'user',
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Tabla para códigos de recuperación

```
-- Tabla para códigos de recuperación (referencia a user_id es más robusta)
CREATE TABLE password_reset_codes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    code VARCHAR(6) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP DEFAULT (CURRENT_TIMESTAMP + INTERVAL 1 HOUR),
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Tabla de predicciones

```
-- Tabla de predicciones (IA)
CREATE TABLE predictions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    input_type ENUM('image', 'text') NOT NULL,
    input_data TEXT NOT NULL,
    result TEXT NOT NULL,
    confidence FLOAT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

8. Componentes y estructura del Frontend (React)

8.1. Estructura de carpetas

```
FRONTEND-PAPA-IA/
├── node_modules/      # Dependencias del proyecto
├── public/            # Archivos públicos accesibles directamente
│   ├── img/           # Imágenes usadas en la interfaz
│   ├── favicon.ico    # Icono del navegador
│   ├── index.html     # HTML base donde se monta la app React
│   ├── logo192.png
│   ├── logo512.png
│   └── manifest.json  # Configuración para apps PWA
└── robots.txt         # Instrucciones para buscadores
├── src/               # Código fuente principal
    ├── components/    # Componentes de la interfaz
    │   ├── analysis/   # Funcionalidad de análisis IA
    │   │   ├── Analysis.jsx
    │   │   ├── Dashboard.jsx
    │   │   ├── ImageAnalysis.jsx
    │   │   ├── Results.jsx
    │   │   └── TextAnalysis.jsx
    │   ├── auth/        # Autenticación de usuarios
    │   │   ├── ForgotPassword.jsx
    │   │   ├── Login.jsx
    │   │   ├── PerfilUsuario.jsx
    │   │   └── RequestAccount.jsx
    │   ├── AuthRoute.jsx
    │   ├── Layout.jsx
    │   ├── ProtectedRoute.jsx
    │   └── Unauthorized.jsx
    └── context/
        └── AuthContext.js # Contexto global para autenticación
```

8.2. Descripción

public/

index.html: Punto de entrada del proyecto. Aquí se monta todo el contenido de React.

img/: Carpeta para logos e imágenes visibles.

manifest.json y robots.txt: Para configurar la apariencia y comportamiento en navegadores o buscadores.

src/components/

analysis/ – Módulo de Análisis con IA

Dashboard.jsx: Vista principal del análisis, donde se muestran los resultados.

ImageAnalysis.jsx: Componente que carga y analiza imágenes de papas.

TextAnalysis.jsx: Análisis de texto asociado a los resultados (opcional si aplica).

Results.jsx: Componente que muestra los resultados del modelo IA.

Analysis.jsx: Componente contenedor que une todo el flujo de análisis.

auth/ – Módulo de Autenticación

Login.jsx / Register.jsx: Formularios de acceso y registro.

PerfilUsuario.jsx: Muestra los datos del usuario logueado.

ForgotPassword.jsx / RequestAccount.jsx: Recuperación de acceso y solicitudes.

ProtectedRoute.jsx: Componente que protege rutas para usuarios logueados.

Unauthorized.jsx: Muestra un mensaje si el usuario no tiene permisos.

Layout.jsx / AuthRoute.jsx: Estructura general y rutas de autenticación.

context/

AuthContext.js: Maneja el estado global de autenticación con React Context API. Permite que toda la aplicación sepa si el usuario está autenticado o no.

8.3. Componentes Principales

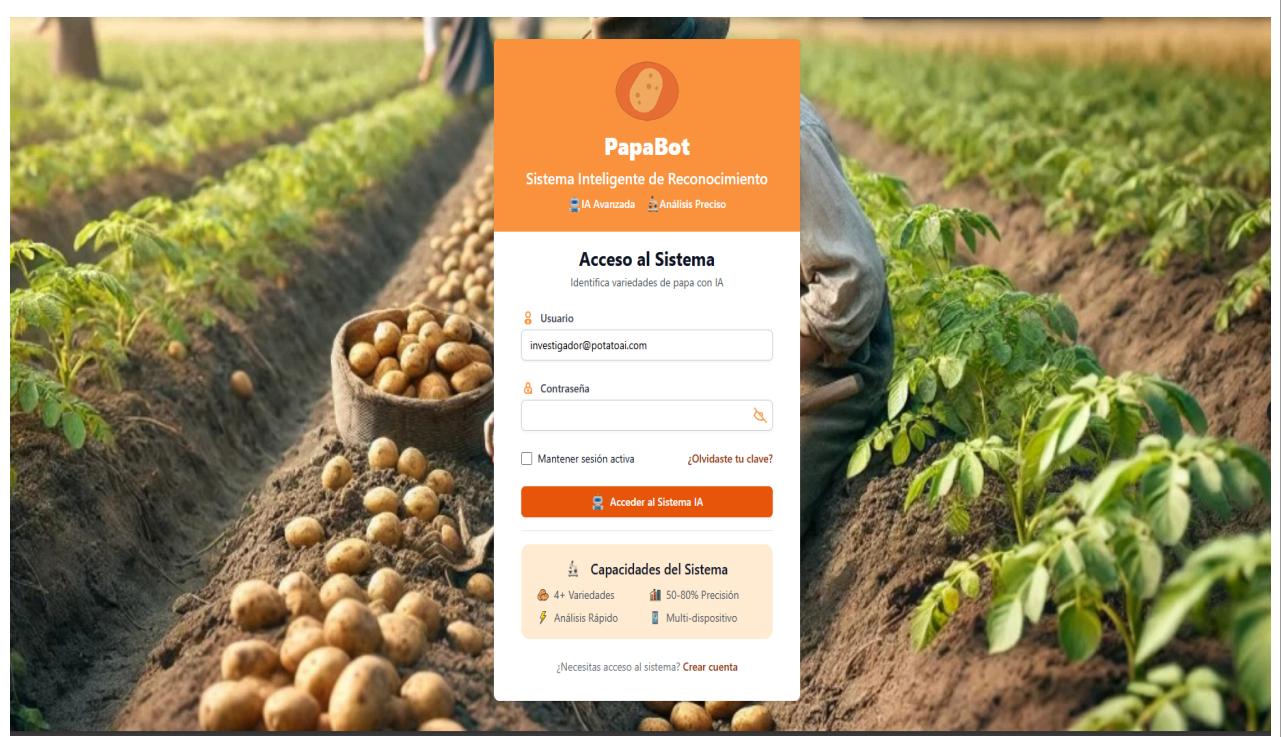
Descripción de componentes:

Componente	Descripción
------------	-------------

Login.jsx	Formulario de inicio de sesión que autentica y genera un token.
ForgotPassword.jsx	Permite recuperar la contraseña a través del correo registrado.
RequestAccount.jsx	Solicitud de creación de cuenta para usuarios nuevos (puede requerir aprobación).
PerfilUsuario.jsx	Muestra información del usuario como nombre, correo, rol y organización.
Dashboard.jsx	Muestra resultados del análisis de imágenes o textos procesados.
Analysis.jsx	Contenedor general del análisis que integra los subcomponentes.
ImageAnalysis.jsx	Permite al usuario cargar y analizar imágenes de papas.
TextAnalysis.jsx	Permite ingresar texto para ser analizado por el modelo.
Results.jsx	Presenta los resultados del modelo después del análisis.
AuthRoute.jsx	Controla el acceso a rutas públicas dependiendo del estado de autenticación.
Layout.jsx	Estructura base de la interfaz
ProtectedRoute.jsx	Protege rutas para que solo usuarios autenticados puedan acceder.
Unauthorized.jsx	Página mostrada cuando un usuario no tiene permisos suficientes.
AuthContext.js	Archivo de contexto que gestiona el estado global de autenticación.

Vista de componentes:

Login.jsx



ForgotPassword.jsx



RequestAccount.jsx



PapaBot
Crear Cuenta

Información Personal

Nombre de usuario *

Nombre completo *

Ej: Juan Pérez

Correo Electrónico *

Credenciales de Acceso

Contraseña *

Mínimo 8 caracteres, incluyendo mayúsculas, minúsculas y números

Confirmar Contraseña *

Tipo de Cuenta *

Usuario Normal

Registrarse

¿Ya tienes una cuenta? [Iniciar sesión](#)

Al registrarte, aceptas nuestros Términos de Servicio y Política de Privacidad.
© 2025 PapaBot. Todos los derechos reservados.



PerfilUsuario.jsx



PapaBot
Análisis Inteligente

Regresar



Perfil del Usuario

Nombre de usuario Jenifer Fiorella	Nombre completo	Rol Admin
Profesión Ej. Ingeniera de sistemas	Institución Ej. Universidad Nacional del Centro del Perú	Número de celular Ej. +51 999 123 456
LinkedIn https://linkedin.com/in/...		
Otros Cualquier otro dato adicional...		

Dashboard.jsx

Panel de Control del Administrador

Resumen del sistema y estadísticas generales

Ir al análisis

Fecha actual
21/7/2025

Usuarios Registrados
0

Imágenes Analizadas
0

Papa más común
0

Horas de uso del modelo
0

GRÁFICAS PAPABOT

Distribución de Características

Carácteristica	Valor
Tamaño	45
Color	30
Textura	18
Forma	15

Estado de las Papas Analizadas

Estado	Porcentaje
Sanas	70%
Defectuosas	30%

Analysis.jsx

PapaBot
Análisis Inteligente

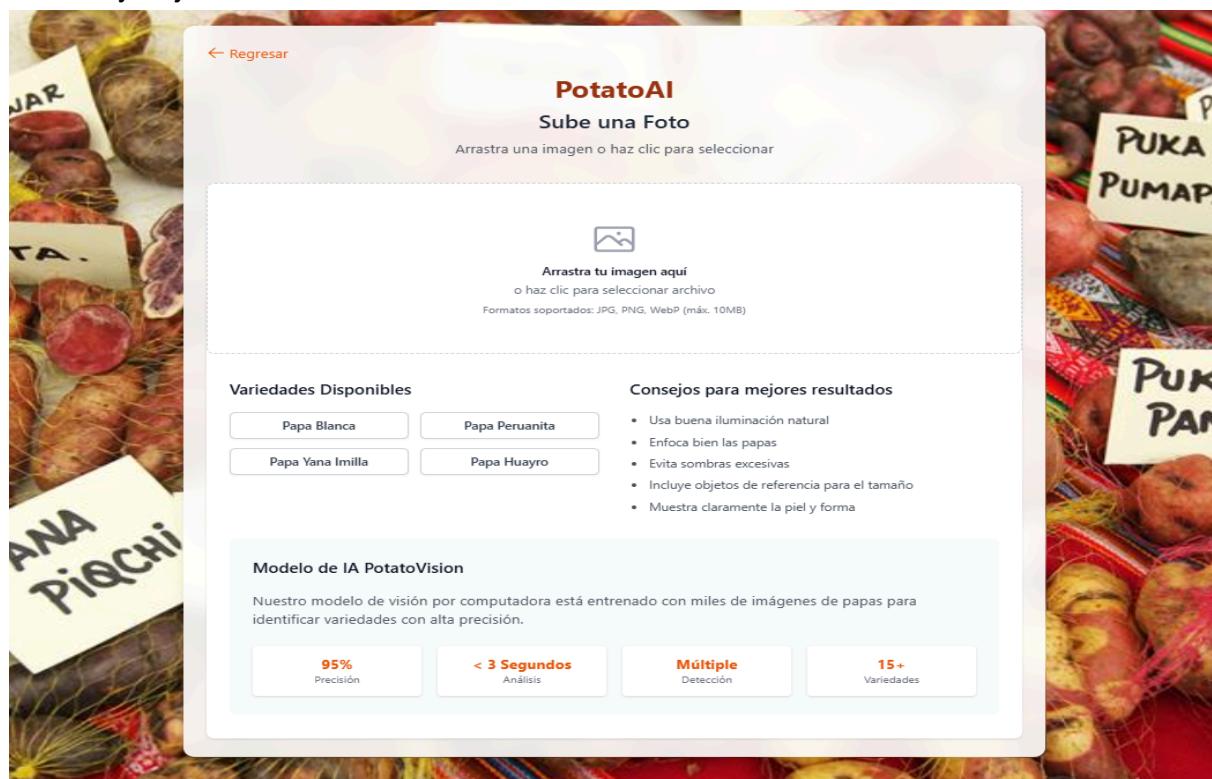
Bienvenido a PapaBot

La plataforma más avanzada para identificación y análisis de variedades de papa usando inteligencia artificial

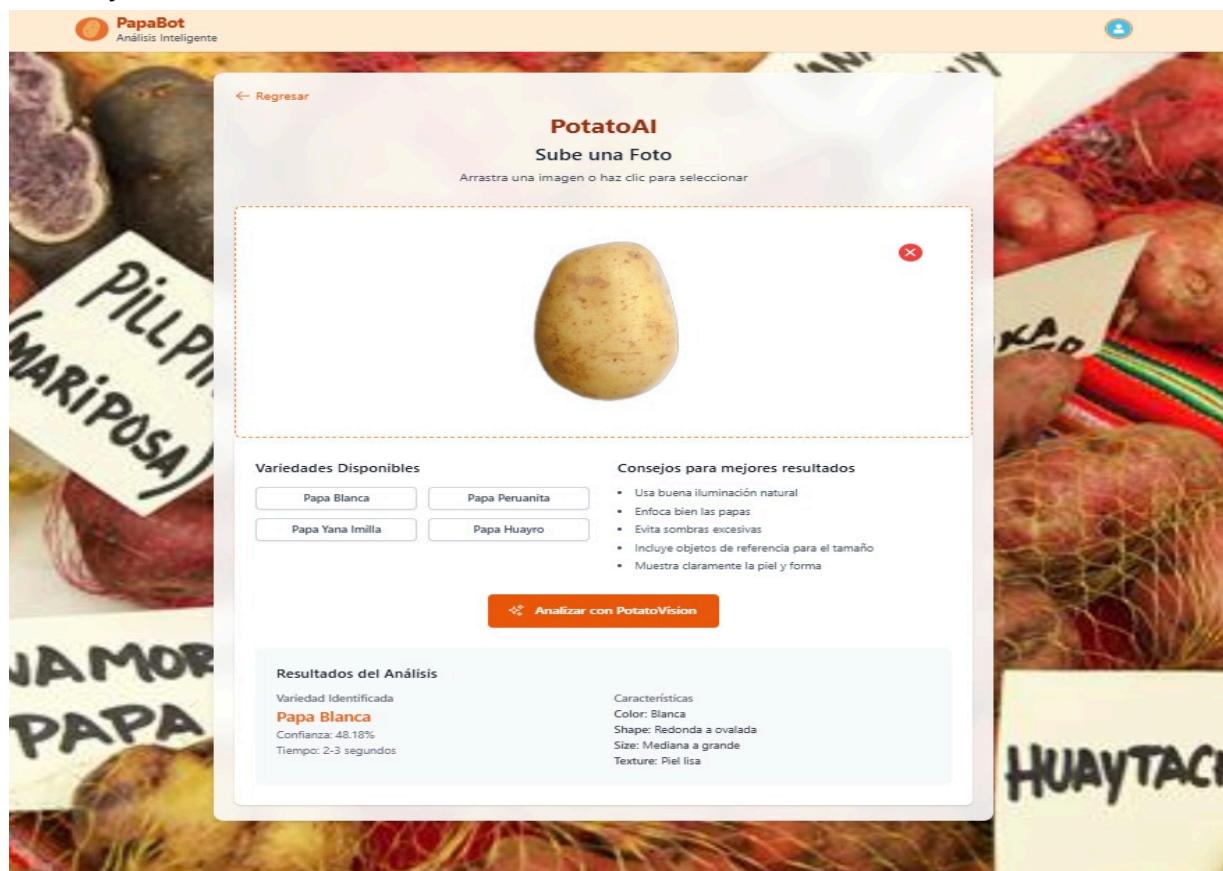
Análisis por texto **Análisis por Imagen** **Ver Dashboard**

ImageAnalysis.jsx

TextAnalysis.jsx



Results.jsx



9. Autenticación y Seguridad

- Uso de JWT (token) para proteger rutas en el backend.
- Almacenamiento del token en localStorage del navegador.
- Middleware en Flask para validar token

9.1. Backend

Función	Ubicación	Descripción
Generación de token JWT	routes/auth_routes.py → función login()	Al iniciar sesión, se crea un token con create_access_token(identity=usuario), que contiene la identidad del usuario autenticado.
Protección de rutas	Archivos de rutas como prediction_routes.py, user_routes.py, admin_routes.py	Se usa el decorador @jwt_required() para proteger rutas que requieren autenticación. El backend solo responde si el token es válido.
Identificación de usuario	Dentro de funciones protegidas	Se obtiene el usuario autenticado con get_jwt_identity() y se usa para acceder a sus datos o verificar su rol.
Control de acceso por rol	Ej. en /admin/stats	Se compara el rol (user['rol']) para permitir o denegar acceso a rutas específicas. Ejemplo: solo 'admin' puede ver estadísticas globales.
Middleware JWT	Configuración en app.py con JWTManager(app)	Configura el manejo de tokens y errores relacionados (token expirado, inválido, etc.).

9.2. Frontend

Elemento	Ubicación	Descripción

Protección de rutas	components/ProtectedRoute.jsx	Verifica si el usuario está autenticado (a través del contexto). Si no lo está, redirige al login. Si está cargando, muestra "Cargando...".
Almacenamiento del token	context/AuthContext.jsx	Cuando el usuario inicia sesión, el token JWT recibido se guarda en localStorage para mantener la sesión persistente.
Verificación automática de sesión	context/AuthContext.jsx	Al cargar la aplicación, se revisa si hay un token en localStorage. Si existe, se decodifica y se usa para establecer el estado del usuario.
Cierre de sesión (logout)	En un botón o menú (por ejemplo, Navbar.jsx)	Elimina el token del localStorage y limpia el estado de autenticación, redirigiendo al login.
Consumo de rutas protegidas	Cualquier componente fetch/axios con al backend	Se incluyen los headers con Authorization: Bearer TOKEN en las peticiones para acceder a las rutas protegidas por JWT.

10. Diseño y UX

- Prototipado de interfaces: Para el diseño de las interfaces de usuario se hizo el prototipado con las herramientas que nos ofrece Canva. Esta es una fase previa al desarrollo del frontend para garantizar que las interfaces sean visualmente atractivas e intuitivas en su manejo.
- Interfaz responsive con Tailwind CSS: La aplicación está construida sobre Tailwind CSS, lo que permite una adaptación fluida a diferentes tamaños de pantalla: móviles, tablets y escritorios.
- Diseño limpio, moderno y accesible: Se prioriza la usabilidad y accesibilidad del sistema, con un enfoque minimalista que mejora la experiencia de usuario (UX). Se utilizaron principios como jerarquía visual, espacio coherente y navegación intuitiva.
- Paleta de colores personalizada (papa-orange): En el archivo tailwind.config.js, se definió una paleta de colores bajo el nombre papa-orange. Esta paleta se aplicó en

toda la interfaz para mantener consistencia visual y reforzar la identidad del proyecto.

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {
      colors: {
        'papa-orange': {
          100: '#fff7ed', // naranja muy claro
          200: '#ffedd5', // crema anaranjado
          300: '#fed7aa', // durazno suave
          400: '#fdbd74', // durazno intenso
          500: '#fb923c', // naranja medio
          600: '#f97316', // naranja brillante
          700: '#ea580c', // naranja oscuro
          800: '#c2410c', // naranja quemado
          900: '#9a3412', // marrón anaranjado
        },
      },
    },
  },
  plugins: [],
}
```

- Los estilos fueron aplicados directamente en componentes de React (.jsx), utilizando clases utilitarias de Tailwind, lo cual permitió una rápida personalización de cada vista manteniendo el control central de la identidad visual.
- Consistencia visual en todos los elementos: Para los botones, textos, contenedores y elementos interactivos comparten la misma lógica de color y estilo, lo cual genera una experiencia homogénea para el usuario.

11. Conclusión

Este proyecto integró de manera efectiva tecnologías modernas de desarrollo web con modelos de inteligencia artificial, en un entorno de trabajo colaborativo basado en la metodología ágil SCRUM. A lo largo del desarrollo, se abordaron desafíos técnicos relacionados con la autenticación segura, la integración de modelos de IA para el análisis de texto e imágenes, y la construcción de una interfaz responsive y amigable para el usuario.

Gracias a una planificación por sprints, una correcta distribución de roles (Product Owner, Scrum Master, Developers y QA Tester), y una comunicación constante entre los integrantes del equipo, se logró construir una plataforma web robusta, funcional y escalable, cumpliendo con todos los requisitos definidos inicialmente: registro e inicio de sesión, protección de rutas, análisis automatizado de entradas, visualización de resultados y gestión de perfiles.

Este trabajo no solo permitió aplicar conocimientos técnicos, sino también reforzar habilidades de trabajo en equipo, resolución de problemas y documentación profesional. El resultado final es una aplicación lista para ser escalada, integrada con nuevas funcionalidades o adaptada a otros dominios.

ANEXOS:

MANUAL DE INSTALACIÓN

1. Requisitos Previos

Tener instalado:

- Python 3.10+
- Node.js (16 o superior)
- MySQL Server
- Git
- Un editor de código como VS Code

2. Clonar el repositorio

Para acceder a los archivos del backend y el frontend

```
git clone https://github.com/usuario/proyecto.git  
cd FINAL_v2
```

o también realizar la descarga directa del archivo comprimido, exactamente la carpeta es “Final_v2”, abrirlo en el Visual Code el backend y el frontend.

3. Configuración de la base de datos MySQL

En un query de Mysql ingresar el script que creará las tablas y creará un usuario

```
-- Crear la base de datos y seleccionarla  
CREATE DATABASE IF NOT EXISTS papa_db;  
USE papa_db;  
  
-- Tabla de usuarios  
CREATE TABLE IF NOT EXISTS users (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(50) UNIQUE NOT NULL,      -- Nombre de usuario  
    email VARCHAR(100) UNIQUE NOT NULL,        -- Correo electrónico  
    full_name VARCHAR(100) NOT NULL,           -- Nombre completo  
    password VARCHAR(255) NOT NULL,             -- Contraseña (hasheada en producción)  
    role ENUM('user', 'admin') DEFAULT 'user', -- Rol del usuario  
    is_active BOOLEAN DEFAULT TRUE,  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```

        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- Tabla de códigos de recuperación
CREATE TABLE IF NOT EXISTS password_reset_codes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(100) NOT NULL,
    code VARCHAR(6) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP DEFAULT (CURRENT_TIMESTAMP + INTERVAL 1 HOUR),
    FOREIGN KEY (email) REFERENCES users(email) ON DELETE CASCADE
);

-- Tabla de predicciones (IA)
CREATE TABLE IF NOT EXISTS predictions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    input_type ENUM('image', 'text') NOT NULL,
    input_data TEXT NOT NULL,          -- Texto descriptivo o ruta de imagen
    result TEXT NOT NULL,            -- Resultado de la IA
    confidence FLOAT,                -- Nivel de confianza (opcional)
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Insertar usuarios: 1 admin y 1 usuario normal
INSERT INTO users (username, email, full_name, password, role)
VALUES
('admin_user', 'admin@correo.com', 'Administrador del Sistema', 'admin1234', 'admin'),
('juan_perez', 'juan@correo.com', 'Juan Pérez', '12345678', 'user');

-- Insertar códigos de recuperación para ambos usuarios
INSERT INTO password_reset_codes (email, code)
VALUES
('admin@correo.com', 'ADM001'),
('juan@correo.com', 'USR002');

-- Insertar predicciones para el usuario normal (id = 2)
INSERT INTO predictions (user_id, input_type, input_data, result, confidence)
VALUES
(2, 'text', 'Papa amarilla de forma redonda y piel suave', 'Papa Amarilla', 0.91),
(2, 'image', '/uploads/papas/huayro.jpg', 'Papa Huayro', 0.93);

```

4. Configurar el Backend (Flask)

```

cd backend
python -m venv venv
venv\Scripts\activate

```

Si existe conflictos con el .env, eliminar el actual y crear uno nuevo

```
python -m venv .venv  
#para la activación  
.venv\Scripts\activate
```

Las librerías para instalar en el backend

```
# Instalar dependencias  
pip install -r requirements.txt
```

Para la ejecución del backend, ya dentro del entorno colocar:

```
python app.py
```

5. Configurar el Frontend (React)

Ingresar a la carpeta del frontend e instalar para agregar la carpeta de node modules

```
cd ../frontend  
npm install
```

Para ejecutarlo

```
npm start
```