Tecnología Superior en Desarrollo de Software

Fecha: 25/09/2024

INSTITUTO SUPERIOR TECNOLOGICO DEL AZUAY

PROGRAMACIÓN ORIENTADA A OBJETOS

**Proyecto Final**

**Automatización de Parqueadero mediante IA**

1. **Contexto del Proyecto Final:**

El **Proyecto de Automatización de Parqueadero mediante IA** tiene como objetivo implementar una solución integral que automatice la gestión del estacionamiento del TecAzuay utilizando tecnologías de inteligencia artificial, visión por computadora y sistemas de notificación en tiempo real. Este proyecto tiene como finalidad no solo agilizar los procesos de entrada y salida de vehículos, sino también optimizar el uso de los espacios disponibles, reducir la intervención humana, y mejorar la toma de decisiones para la administración del parqueadero mediante la creación de estructuras tipo informe generadas con los datos obtenidos, dichos informes podrán ser agendados de manera periódica o por causales inmediatas de ser necesario.

#### **Objetivos del Proyecto**

1. **Automatización del registro de entradas y salidas**: A través de cámaras equipadas con sistemas de reconocimiento de placas vehiculares, el parqueadero podrá identificar automáticamente los vehículos que ingresan y salen. Esto eliminará la necesidad de sistemas manuales de control o intervención humana, permitiendo una operación fluida y eficiente.
2. **Detección y monitoreo de espacios disponibles**: El uso de cámaras conectadas a un sistema de procesamiento de imágenes permitirá detectar automáticamente qué espacios están ocupados y cuáles están libres. Esta información será reflejada en tiempo real en una aplicación web y móvil para que los usuarios puedan tomar decisiones rápidas sobre el lugar donde estacionar.
3. **Notificaciones de entrada y salida**: A través de una aplicación móvil, los usuarios recibirán notificaciones instantáneas cuando su vehículo ingrese o salga del parqueadero, mejorando la seguridad y el control de acceso.
4. **Generación de informes para administración**: Un módulo de administración permitirá generar informes detallados sobre el uso del parqueadero, incluyendo estadísticas de ocupación, patrones de ingreso y salida, así como el registro de usuarios y vehículos.
5. **Gestión de usuarios y control de accesos**: El sistema incluirá un módulo para la gestión de usuarios registrados, permitiendo a los administradores otorgar y controlar permisos de acceso a diferentes áreas del parqueadero.

#### **Tecnologías Utilizadas**

Para la implementación de este proyecto se han utilizado las siguientes tecnologías:

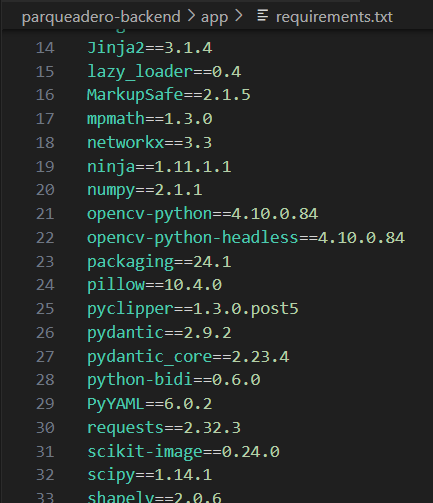
* **Lenguajes de Programación**: Python para el desarrollo de la IA y el procesamiento de imágenes, y JavaScript (Next.js, React) para la interfaz web.
* **Frameworks de Desarrollo**: Flask y Django en el backend, Next.js para la construcción del frontend.
* **Reconocimiento de Imágenes**: Se utilizará OpenCV y TensorFlow para el procesamiento y reconocimiento de imágenes captadas por cámaras web.
* **Base de Datos**: MongoDB para el registro de usuarios, vehículos y datos de ocupación de espacios.
* **Notificaciones en Tiempo Real**: Se utilizará Firebase para el envío de notificaciones push en la aplicación móvil.
* **Despliegue**: Docker y Kubernetes para la gestión de contenedores y el despliegue del sistema en entornos de producción.

#### **Propuestas de Mejora**

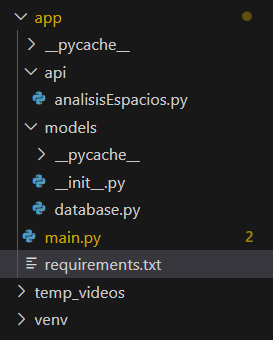
El proyecto, aunque completo en términos funcionales, presenta áreas donde pueden realizarse mejoras importantes tanto a nivel de código como en la arquitectura general. A continuación se detallan las áreas identificadas y las mejoras propuestas:

#### **Mejoras en el Código y el Sistema**

1. **Actualización de Versiones de Python y Dependencias**: Se recomienda actualizar Python a su versión más reciente para aprovechar las nuevas funcionalidades y optimizaciones en rendimiento. Además, bibliotecas como OpenCV, TensorFlow, Flask y las herramientas de reconocimiento de imágenes deberían mantenerse en sus versiones más recientes para evitar vulnerabilidades y mejorar la compatibilidad con nuevas tecnologías.



1. **Refactorización del Código**: Algunas funciones complejas encargadas del procesamiento de imágenes y la gestión de datos pueden dividirse en componentes más pequeños y manejables. Esto no solo mejoraría la legibilidad del código, sino también facilitaría su mantenimiento y futuras actualizaciones. Por ejemplo, separar el proceso de lectura de placa en un módulo y el proceso de identificación de espacios en otro independiente mejoraría su reutilización.



1. **Patrones de Diseño y Buenas Prácticas**: Implementar patrones de diseño como el patrón de **Observador** para las notificaciones, o el patrón **Factory** para la creación de instancias de diferentes tipos de usuarios (usuarios regulares, administradores) mejoraría la escalabilidad del sistema. El uso de **tipado estático** en Python a través de la herramienta **mypy** permitiría detectar errores en tiempo de desarrollo.

#### **Mejoras a la Robustez del Proyecto**

1. **Linter y Formateo Automático**: Utilizar herramientas como **pylint** o **flake8** para asegurar que el código sigue estándares de calidad y legibilidad. También se puede implementar **black** para un formateo automático que asegure consistencia en el estilo del código a lo largo del proyecto.
2. **Optimización de Funciones Críticas**: En el procesamiento de imágenes, se podrían identificar cuellos de botella, como la detección de contornos en las placas de vehículos. Aquí se propone el uso de técnicas de paralelización o el uso de GPUs para acelerar el procesamiento y reducir la latencia.

#### **Planes de Pruebas Completos**

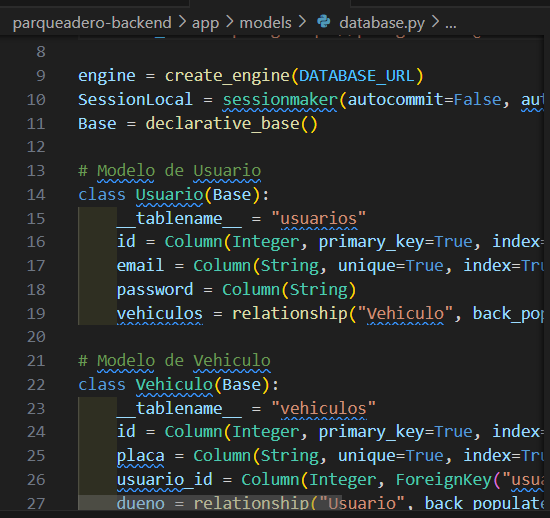
1. **Pruebas Unitarias**: Para garantizar la correcta funcionalidad de los módulos principales, se implementarán pruebas unitarias para los módulos de reconocimiento de placas, detección de espacios, y gestión de usuarios. Se recomienda el uso de bibliotecas como **unittest** o **pytest** para crear y ejecutar estas pruebas.
2. **Pruebas de Integración**: Además de las pruebas unitarias, es crucial verificar la interacción entre los diferentes componentes del sistema, como la integración entre el sistema de procesamiento de imágenes y la base de datos. **pytest-django** o **pytest-flask** son herramientas adecuadas para este tipo de pruebas.
3. **Pruebas de Extremo a Extremo (E2E)**: Para garantizar que el flujo completo de usuario (desde la entrada de un vehículo hasta la notificación en la app móvil) funciona sin problemas, se recomienda la implementación de pruebas E2E utilizando **Selenium**.

#### **Mejoras en el Pipeline de CI/CD**

1. **Automatización de Pruebas y Despliegue**: La automatización de pruebas a través de herramientas como **GitHub Actions** permitirá ejecutar las pruebas automáticamente cada vez que se suba código al repositorio. Además, se puede configurar un flujo de trabajo para desplegar automáticamente nuevas versiones del sistema en un entorno de staging.
2. **Contenerización**: Se recomienda contenerizar el proyecto utilizando **Docker**. Esto garantizará que el sistema sea fácil de desplegar y que funcione en cualquier entorno. También facilita la colaboración entre desarrolladores, quienes pueden trabajar en entornos idénticos independientemente de sus sistemas operativos.

#### **Acceso a Datos**

1. **Optimización del Acceso a Datos**: Se puede mejorar el acceso a datos mediante la implementación de **índices** en las colecciones de MongoDB para optimizar las búsquedas de usuarios o registros de vehículos. Además, el uso de **SQLAlchemy** puede ser evaluado si se requiere integrar una base de datos SQL en un futuro.



1. **Validación de Datos**: Para mejorar la robustez del sistema y evitar datos incorrectos o inconsistentes, se puede implementar **Pydantic** para validar los datos de entrada tanto en la API como en el frontend.

#### **Empaquetado y Despliegue**

1. **Empaquetado de la Aplicación**: El empaquetado con **Poetry** permite una gestión eficiente de dependencias, además de facilitar la creación de distribuciones del proyecto para diferentes entornos. Esto será especialmente útil para gestionar versiones y para integrar nuevas funcionalidades en el futuro.
2. **Despliegue en Kubernetes**: Utilizar **Kubernetes** para el despliegue en producción permitirá la escalabilidad automática del sistema según la demanda de usuarios. Además, garantiza una alta disponibilidad y recuperación ante fallos en los diferentes componentes del sistema.

#### **Tecnologías y Prácticas No Utilizadas**

1. **Bases de Datos SQL**: Aunque las bases de datos SQL son robustas y estructuradas, se ha decidido optar por **MongoDB** debido a su flexibilidad en la gestión de datos no estructurados, lo que se adapta mejor al flujo de trabajo en tiempo real del sistema.
2. **Deep Learning Extenso**: Aunque se ha considerado el uso de modelos de **Deep Learning** para el reconocimiento de imágenes, se ha optado por modelos más ligeros basados en redes neuronales convolucionales preentrenadas, debido a la complejidad y costo computacional que representaría entrenar y desplegar modelos más avanzados.

#### **Conclusión**

Las mejoras propuestas están orientadas a incrementar la eficiencia, mantenibilidad y robustez del sistema de automatización del parqueadero. Al implementar técnicas de refactorización, realizar pruebas completas y automatizar procesos mediante un pipeline CI/CD, el proyecto ganará en calidad y confiabilidad. Las propuestas reflejan un enfoque de desarrollo centrado en las buenas prácticas de ingeniería de software y en la optimización de recursos, lo que garantiza que el sistema pueda adaptarse y escalarse en el futuro.