Aplicación: Vektor AI

Proyecto: FlightOnTime

Documento: Manual Técnico
Área: Aviación Civil / Logística / Transporte Aéreo

Descripción:
Vektor AI es una aplicación que integra un modelo predictivo de Machine Learning para estimar la probabilidad de retraso en vuelos comerciales, exponiendo dicha predicción a través de una API REST en tiempo real.

Equipos responsables:

- Equipo de Data Science
- Equipo de Back-End

Estado del proyecto: MVP en desarrollo
Fecha: 16/01/2025
Versión: 1.0

VEKTOR AI

Tabla de contenido

1. Control de versiones del documento **3**

2. Introducción **4**

3. Descripción General del Proyecto **5**

4. Alcance y Objetivos del Sistema **6**

4.1 Objetivo General 6

4.2 Objetivos Específicos 6

4.3 Alcance del Sistema (MVP) 6

5. Arquitectura General del Sistema **7**

5.1 Componentes Principales 7

5.2 Flujo General de Datos 8

5.3 Consideraciones Arquitectónicas 8

6. Modelo Predictivo de Machine Learning **8**

6.1 Tipo de Problema y Definición del Target 9

6.2 Modelos Evaluados y Selección del Modelo Final 9

6.3 Dataset y Variables de Entrada 9

6.4 Integración de Datos Climáticos 10

6.5 Entrenamiento y Evaluación del Modelo 10

6.6 Serialización e Integración con el Backend 11

6.7 Explicabilidad y Análisis de Impacto 11

6.8 Versionamiento y Estado Actual 11

7. Preprocesamiento y Contrato de Integración **11**

7.1 Principios Generales de Preprocesamiento 12

7.2 Responsabilidades del Back-End 12

7.3 Responsabilidades del Servicio de Machine Learning 12

7.4 Contrato de Entrada 13

7.5 Contrato de Salida 13

7.6 Manejo de Errores 13

8. API REST – Backend **14**

8.1 Endpoint de Predicción 14

8.2 Estructura de la Solicitud 14

8.3 Validación de Entradas 14

8.4 Estructura de la Respuesta 15

8.5 Manejo de Errores y Respuestas 15

8.6 Persistencia de Predicciones 15

9. Persistencia y Almacenamiento **16**

9.1 Objetivo de la Persistencia 16

9.2 Información Almacenada 16

9.3 Tecnología de Almacenamiento 16

9.4 Uso de la Información Persistida 17

9.5 Consideraciones de Diseño 17

10. Despliegue y Ejecución 17

10.1 Requisitos del Sistema 17
10.2 Variables de Entorno 18
10.3 Ejecución Mediante Docker Compose 18
10.4 Ejecución Local 18
10.5 Verificación de Funcionamiento 18
10.6 Despliegue en Oracle Cloud Infrastructure (OCI) 19

11. Seguridad y Consideraciones Técnicas 19

11.1 Seguridad de la API 19
11.2 Manejo de Credenciales y Configuración Sensible 20
11.3 Seguridad del Modelo de Machine Learning 20
11.4 Consideraciones de Rendimiento 20
11.5 Enfoque de Diseño y Evolución del Sistema 20

12. Operación y Uso del Sistema 21

12.1 Flujo General de Uso 21
12.2 Uso del Endpoint de Predicción 21
12.3 Interpretación de Resultados 22
12.4 Visualización y Dashboard 22
12.5 Registro y Auditoría de Predicciones 22

13. Despliegue en Oracle Cloud Infrastructure (OCI) 23

13.1 Objetivo del Despliegue en la Nube 23
13.2 Arquitectura de Despliegue en OCI 23
13.3 Recursos Utilizados en OCI 23
13.4 Preparación del Entorno 24
13.5 Construcción y Ejecución de Contenedores 24
13.6 Configuración de Variables de Entorno en OCI 24
13.7 Exposición y Acceso a la API 25
13.8 Verificación del Despliegue 25
13.9 Consideraciones del Entorno Always Free 25

14. Conclusiones Técnicas 26

VEKTOR AI

1. Control de versiones del documento

Versión	Fecha	Descripción del cambio	Responsable
1.0	16/01/2025	Creación inicial del manual técnico de Vektor AI	Equipo Vektor AI
1.1	16/01/2025	Ajuste conceptual: FlightOnTime como caso de uso	Equipo Vektor AI
1.2	16/01/2025	Inclusión de variables climáticas	Equipo Data Science
1.3	20/01/2025	Inclusión de Deployment en OCI	Equipo Back End

2. Introducción

El presente Manual Técnico documenta el diseño, arquitectura y funcionamiento de Vektor AI, una aplicación de análisis predictivo orientada a la estimación de retrasos en vuelos comerciales, desarrollada mediante técnicas de Machine Learning e integración de datos climáticos en tiempo real.

Vektor AI se aplica al caso de uso FlightOnTime, cuyo objetivo es anticipar si un vuelo llegará puntual o presentará un retraso operativo significativo, definido como un atraso igual o superior a 15 minutos respecto a su horario programado. La solución se enmarca en el sector de Aviación Civil, Logística y Transporte Aéreo, donde la anticipación de retrasos permite reducir impactos operativos, mejorar la planificación y aumentar la transparencia hacia los pasajeros.

La aplicación utiliza un modelo de clasificación supervisada basado en Random Forest, entrenado con datos históricos de vuelos y enriquecido con variables meteorológicas obtenidas desde servicios externos. A partir de información disponible antes del despegue, el sistema genera una predicción binaria (Puntual / Retrasado) junto con una probabilidad asociada, expuesta a través de una API REST para su consumo por sistemas externos.

Vektor AI ha sido diseñada como un Producto Mínimo Viable (MVP), priorizando simplicidad, coherencia técnica y reproducibilidad. El sistema incorpora mecanismos de validación, persistencia de predicciones, análisis agregado y visualización, así como un proceso de despliegue contenerizado que facilita su ejecución en distintos entornos.

Este documento está dirigido a desarrolladores, evaluadores técnicos y equipos de mantenimiento, y tiene como propósito proporcionar una visión clara y alineada del sistema, permitiendo comprender su alcance, funcionamiento, limitaciones actuales y posibilidades de evolución futura.

3. Descripción General del Proyecto

Vektor AI es una solución tecnológica orientada a la predicción temprana de retrasos en vuelos comerciales, diseñada para apoyar la toma de decisiones en el ecosistema aeronáutico. El sistema permite estimar, antes del despegue, la probabilidad de que un vuelo se vea afectado por un retraso operativo significativo, entregando información clara, cuantificable y accionable a distintos actores del sector.

El proyecto aborda una problemática recurrente en la aviación civil: la dificultad de anticipar retrasos con suficiente antelación para mitigar su impacto. Estos retrasos generan efectos en cascada que afectan la experiencia de los pasajeros, la planificación de las aerolíneas y la operación de aeropuertos. Vektor AI responde a esta necesidad mediante un enfoque predictivo basado en datos históricos y condiciones externas relevantes, como el clima.

Desde el punto de vista funcional, el sistema recibe información estructurada de un vuelo, incluyendo datos operacionales (aerolínea, aeropuerto de origen y destino, fecha y hora de partida, distancia estimada) y variables meteorológicas obtenidas en tiempo real desde servicios externos. A partir de estos datos, el modelo de Machine Learning evalúa el riesgo de retraso y devuelve una clasificación binaria junto con una probabilidad asociada.

La arquitectura del proyecto está organizada en componentes desacoplados, lo que permite una clara separación de responsabilidades. El modelo predictivo es desarrollado y entrenado por el equipo de Data Science utilizando técnicas de aprendizaje supervisado, mientras que el equipo de Back-End implementa una API REST que expone las predicciones de forma estandarizada. Esta API actúa como punto de integración entre el modelo y los consumidores externos del servicio.

Vektor AI se concibe como un Producto Mínimo Viable, enfocado en demostrar la viabilidad técnica y el valor del enfoque predictivo. El alcance del proyecto prioriza la clasificación puntual versus retrasado, la trazabilidad de las predicciones y la facilidad de despliegue, sin perder de vista la posibilidad de evolución futura hacia funcionalidades más avanzadas, como análisis explicativo, procesamiento en lote o integración con nuevas fuentes de datos.

En conjunto, el proyecto FlightOnTime materializa una solución completa que conecta datos, modelos predictivos e infraestructura de software, entregando una base sólida para la experimentación, evaluación y futura escalabilidad del sistema en contextos reales de operación aérea.

4. Alcance y Objetivos del Sistema

En la presente sección se define los objetivos y el alcance funcional de Vektor AI, estableciendo de forma explícita qué problemáticas aborda el sistema, qué funcionalidades incluye y cuáles quedan fuera del Producto Mínimo Viable (MVP). Esta delimitación es clave para comprender el contexto técnico del proyecto y evitar interpretaciones erróneas sobre sus capacidades actuales.

4.1 Objetivo General

El objetivo principal de Vektor AI es predecir, antes del despegue, si un vuelo comercial se comportará de forma puntual o presentará un retraso operativo significativo, definido como un atraso igual o superior a 15 minutos respecto a su horario programado.

La predicción se entrega mediante una clasificación binaria acompañada de una probabilidad asociada, permitiendo a los distintos actores del sistema aeronáutico anticipar escenarios de riesgo y tomar decisiones informadas.

4.2 Objetivos Específicos

Para cumplir con su objetivo general, el sistema persigue los siguientes objetivos específicos:

- Desarrollar un modelo de Machine Learning capaz de identificar patrones de retraso a partir de datos históricos de vuelos y variables climáticas.
- Integrar información meteorológica en tiempo real como factor influyente en la predicción.
- Exponer el resultado de la predicción a través de una API REST estandarizada y fácilmente consumible.
- Validar y normalizar los datos de entrada para garantizar la consistencia de las predicciones.
- Persistir las solicitudes y resultados de predicción para fines de auditoría y análisis posterior.
- Proveer estadísticas agregadas que permitan observar el comportamiento general del sistema.

4.3 Alcance del Sistema (MVP)

El alcance del Producto Mínimo Viable de Vektor AI incluye las siguientes funcionalidades:

- Predicción individual de vuelos mediante una clasificación binaria (Puntual / Retrasado).
- Cálculo y entrega de una probabilidad asociada a la predicción.
- Uso de datos operacionales del vuelo y condiciones climáticas en el aeropuerto de origen.
- Exposición de predicciones mediante un endpoint REST.
- Persistencia de las predicciones realizadas.
- Despliegue contenerizado para facilitar su ejecución en distintos entornos.

En su versión actual, Vektor AI se enfoca en la validación técnica y funcional del enfoque predictivo. Algunas capacidades avanzadas, como la estimación exacta de minutos de retraso, el

reentrenamiento automático del modelo o la integración con sistemas operacionales reales, se consideran como posibles extensiones futuras y no forman parte del alcance de esta iteración.

La definición clara de este alcance permite evaluar el proyecto en función de sus objetivos reales, entendiendo a Vektor AI como una solución exploratoria y validable, con potencial de evolución hacia versiones más robustas y especializadas.

5. Arquitectura General del Sistema

La arquitectura de Vektor AI ha sido diseñada siguiendo principios de modularidad, desacoplamiento y simplicidad, con el objetivo de facilitar el desarrollo colaborativo, la mantenibilidad del sistema y su futura escalabilidad. La solución se organiza en componentes bien definidos que interactúan entre sí mediante interfaces claras y contratos de integración estandarizados.

Desde una perspectiva de alto nivel, el sistema se compone de un cliente consumidor, una API de Back-End, un servicio de predicción basado en Machine Learning, servicios externos de datos climáticos y un componente de persistencia para el almacenamiento de resultados.

5.1 Componentes Principales

Cliente / Consumidor

Corresponde a cualquier sistema o interfaz que consuma la API de Vektor AI. Puede tratarse de herramientas de prueba como Postman o cURL, un frontend web o un sistema externo. El cliente es responsable de enviar la información del vuelo en el formato definido y de procesar la respuesta entregada por el sistema, para ello se mantiene un manual de usuario con las herramientas disponibles para el cliente final.

API Back-End

El Back-End está implementado como una API REST desarrollada en Java con Spring Boot. Este componente actúa como el punto central de orquestación del sistema, siendo responsable de:

- Recibir y validar las solicitudes de predicción.
- Gestionar la obtención de datos climáticos desde servicios externos.
- Preparar y normalizar los datos de entrada requeridos por el modelo predictivo.
- Invocar el servicio de predicción de Machine Learning.
- Construir y devolver la respuesta estandarizada al cliente.
- Persistir las solicitudes y resultados de predicción.

Servicio de Predicción (Machine Learning)

El servicio de predicción encapsula el modelo de Machine Learning entrenado por el equipo de Data Science. Este componente recibe los datos procesados desde el Back-End y devuelve una predicción binaria junto con la probabilidad asociada. El modelo se encuentra previamente entrenado y serializado, y es cargado por el sistema para su uso en tiempo de ejecución.

Servicios Externos de Clima

Vektor AI integra información meteorológica en tiempo real a través de una API externa de clima. Estos datos aportan contexto adicional a la predicción, permitiendo capturar el impacto de

variables como temperatura, viento y precipitaciones en el comportamiento de los vuelos, así como la velocidad del viento medida en la estación meteorológica más cercana.

El sistema integra variables climáticas en tiempo real como parte del proceso de enriquecimiento de datos previo a la inferencia, siendo el backend responsable de su obtención y preparación antes de la ejecución del modelo predictivo.

Persistencia de Datos

El sistema incluye un componente de almacenamiento que permite registrar las solicitudes recibidas y las predicciones generadas. Esta persistencia habilita la trazabilidad de las decisiones del modelo, el análisis histórico y la generación de estadísticas agregadas.

5.2 Flujo General de Datos

El flujo de operación del sistema se desarrolla de la siguiente manera:

1. El cliente envía una solicitud de predicción a la API Back-End con la información del vuelo.
2. El Back-End valida los datos de entrada y obtiene la información climática correspondiente al aeropuerto de origen.
3. Se realiza el preprocesamiento necesario para adaptar los datos al formato esperado por el modelo predictivo.
4. El servicio de Machine Learning evalúa los datos de entrada y genera una predicción junto con su probabilidad.
5. El Back-End recibe el resultado, lo persiste junto con la solicitud original y construye la respuesta final.
6. La respuesta es devuelta al cliente en formato JSON.

5.3 Consideraciones Arquitectónicas

La arquitectura propuesta permite:

- Separar claramente las responsabilidades entre los distintos componentes del sistema.
- Facilitar la integración y el reemplazo del modelo predictivo sin impactar al consumidor.
- Incorporar nuevas fuentes de datos o funcionalidades de manera progresiva.
- Ejecutar el sistema de forma local o contenerizada mediante Docker, garantizando consistencia entre entornos.

Esta estructura proporciona una base sólida para la evolución futura de Vektor AI, manteniendo un equilibrio entre simplicidad y capacidad de expansión.

6. Modelo Predictivo de Machine Learning

El presente apartado refiere el modelo de Machine Learning utilizado en Vektor AI para la predicción de retrasos de vuelos, detallando los modelos evaluados, la definición del problema, las variables consideradas, el proceso de entrenamiento y las decisiones técnicas que condujeron a la selección del modelo final actualmente activo en el sistema.

6.1 Tipo de Problema y Definición del Target

El problema abordado corresponde a una tarea de clasificación supervisada binaria, cuyo objetivo es determinar si un vuelo presentará un retraso operativo significativo.

Para efectos del modelo, un vuelo se considera Retrasado cuando su retraso es igual o superior a 15 minutos respecto a su horario programado. Esta definición se alinea con estándares comúnmente utilizados en la industria aeronáutica para evaluar puntualidad operativa.

La variable objetivo se representa de forma binaria:

- 0: Vuelo puntual (retraso menor a 15 minutos)
- 1: Vuelo retrasado (retraso igual o superior a 15 minutos)

6.2 Modelos Evaluados y Selección del Modelo Final

Durante la fase de experimentación, el equipo de Data Science entrenó y evaluó distintos modelos de clasificación, con el objetivo de identificar la mejor combinación entre desempeño predictivo, estabilidad y viabilidad de ejecución dentro de un entorno de recursos limitados.

Los modelos evaluados fueron:

- Logistic Regression
- Random Forest
- XGBoost
- LightGBM

El modelo LightGBM fue considerado inicialmente, sin embargo, no logró ser incorporado al flujo final debido a los altos requerimientos de procesamiento que presentaba, los cuales excedían las capacidades disponibles para el entorno de despliegue del proyecto.

Tras el proceso de evaluación, se seleccionó Random Forest como modelo final del MVP, debido a que presentó el mejor balance entre rendimiento, precisión y robustez frente a la variabilidad de los datos, manteniendo además una integración sencilla con la arquitectura del sistema.

El sistema utiliza una única versión del modelo, correspondiente a la versión 1.0 con integración climática.

6.3 Dataset y Variables de Entrada

El modelo fue entrenado utilizando un dataset histórico de vuelos comerciales, que incluye información temporal, operacional y contextual del vuelo, complementada con variables climáticas.

Las variables de entrada finales del modelo incluyen, entre otras:

- Variables temporales: trimestre, mes, día del mes, día de la semana, fecha del vuelo.
- Información de la aerolínea: código único del operador, número de vuelo, identificador de aeronave.
- Información de origen y destino: identificadores y códigos de aeropuertos.

- Información de desempeño del vuelo: horarios programados de salida y llegada.
- Variables derivadas temporales: transformaciones seno y coseno para capturar estacionalidad, indicador de fin de semana.
- Variables climáticas: temperatura, humedad, presión, visibilidad, velocidad del viento y condición meteorológica.
- Distancia estimada del vuelo, calculada automáticamente mediante la fórmula de Haversine.
- Fecha de partida en formato estándar para consumo por el backend.

Si bien el dataset original contiene un conjunto amplio de variables operacionales y de desempeño del vuelo, no todas fueron utilizadas como variables de entrada del modelo. Durante el proceso de preparación de datos, se descartaron aquellas variables con un alto porcentaje de valores faltantes, aquellas no disponibles antes del despegue y aquellas que podían introducir leakage de información. El modelo utiliza exclusivamente variables disponibles previo a la salida del vuelo y variables climáticas obtenidas en tiempo real.

El conjunto de variables fue seleccionado tras un proceso de exploración y limpieza de datos, priorizando aquellas con mayor aporte predictivo y disponibilidad consistente.

6.4 Integración de Datos Climáticos

El modelo incorpora variables climáticas como parte de su entrenamiento y proceso de inferencia. Los datos meteorológicos provienen de la API de OpenWeather y se obtienen en tiempo real para el aeropuerto de origen del vuelo.

En caso de que la información climática no esté disponible directamente para un aeropuerto específico, el sistema utiliza los datos de la estación meteorológica más cercana, garantizando la continuidad del proceso de predicción.

La inclusión del clima como feature permite capturar el impacto de condiciones externas en la probabilidad de retraso, mejorando la capacidad explicativa del modelo.

6.5 Entrenamiento y Evaluación del Modelo

El entrenamiento del modelo se realizó utilizando una partición de datos de 80% para entrenamiento y 20% para evaluación (prueba).

Debido al desbalance presente entre vuelos puntuales y retrasados, las métricas oficiales del proyecto priorizan la correcta detección de retrasos reales. Las métricas consideradas son:

- ROC AUC como métrica principal
- Recall, Precision y F1-score de la clase Retrasado
- Matriz de confusión como complemento de análisis

Los resultados obtenidos muestran un desempeño aproximado del 57,76% en la detección de vuelos retrasados y del 76,34% en vuelos no retrasados, valores consistentes con un enfoque MVP orientado a la detección temprana de riesgo.

6.6 Serialización e Integración con el Backend

Una vez entrenado, el modelo fue serializado utilizando la librería Joblib bajo el nombre random_forest_clima_v1.joblib.

El consumo del modelo se realiza a través de un servicio FastAPI, el cual expone el modelo para su uso por parte del backend. Esta arquitectura permite desacoplar el entrenamiento del modelo de su ejecución en producción y facilita futuras actualizaciones.

Previo a la predicción, el backend debe garantizar que las variables de entrada respeten el orden de columnas esperado por el modelo, siendo este el único preprocesamiento obligatorio requerido para la inferencia.

6.7 Explicabilidad y Análisis de Impacto

El modelo Random Forest utilizado en Vektor AI entrega información de importancia de variables a través del atributo feature_importances_. Esta información permite identificar, a nivel global, los factores que más influyen en la probabilidad de retraso de un vuelo.

Entre las variables con mayor impacto se encuentran la hora programada de salida, la hora programada de llegada, el identificador de la aeronave, el aeropuerto de origen y el operador aéreo.

La explicabilidad proporcionada es de carácter parcial y debe ser interpretada como un apoyo informativo general. No representa una explicación causal exacta para cada predicción individual, sino una referencia global del comportamiento del modelo.

6.8 Versionamiento y Estado Actual

La versión actualmente activa del modelo es la v1.0 con integración climática. El trabajo futuro del equipo de Data Science se enfoca en mejorar los niveles de precisión y recall del modelo, explorando nuevas variables y ajustes de entrenamiento.

7. Preprocesamiento y Contrato de Integración

La siguiente sección relata el preprocesamiento de datos requerido para la generación de predicciones y define el contrato de integración entre el Back-End y el servicio de Machine Learning. Su objetivo es asegurar la consistencia de los datos utilizados por el modelo y evitar desalineaciones entre los distintos componentes del sistema.

7.1 Principios Generales de Preprocesamiento

El diseño del sistema establece una separación clara de responsabilidades entre el Back-End y el servicio de Machine Learning:

- El servicio de Machine Learning es la única fuente de verdad respecto al preprocesamiento del modelo.
- El Back-End no replica transformaciones propias del modelo, sino que garantiza la validez, consistencia y completitud de los datos de entrada.
- Todas las transformaciones de features, encoding de variables categóricas y generación de variables derivadas se realizan dentro del servicio de Machine Learning.

Este enfoque reduce el riesgo de inconsistencias, facilita el mantenimiento del sistema y permite actualizar el modelo sin impactar la capa de consumo.

7.2 Responsabilidades del Back-End

Previo a solicitar una predicción, el Back-End debe asegurar que los datos enviados al servicio de Machine Learning cumplen con los requisitos mínimos establecidos. En particular, el Back-End es responsable de:

- Validar la presencia de todos los campos obligatorios definidos en el contrato de entrada.
- Verificar el tipo de dato y rango válido de cada campo.
- Normalizar formatos básicos, como fechas, horas y códigos de aeropuertos.
- Gestionar la obtención de datos climáticos desde servicios externos.
- Garantizar la coherencia temporal entre la información del vuelo y las variables climáticas.
- Rechazar solicitudes incompletas o inválidas, devolviendo mensajes de error claros.

El Back-End no realiza encoding, escalado ni generación de variables cíclicas, ya que estas transformaciones forman parte del flujo interno del modelo.

7.3 Responsabilidades del Servicio de Machine Learning

El servicio de Machine Learning recibe los datos validados desde el Back-End y ejecuta el flujo completo de preprocesamiento y predicción. Sus responsabilidades incluyen:

- Aplicar las transformaciones necesarias a las variables de entrada.
- Codificar variables categóricas según los encoders utilizados durante el entrenamiento.
- Generar variables derivadas temporales y climáticas.
- Asegurar el orden correcto de las columnas esperado por el modelo.
- Ejecutar la inferencia utilizando el modelo serializado.
- Devolver la predicción junto con la probabilidad asociada.

Este servicio encapsula toda la lógica dependiente del modelo, permitiendo su evolución sin modificar el contrato externo.

7.4 Contrato de Entrada

El contrato de entrada define el formato estándar que debe cumplir toda solicitud de predicción enviada al sistema. La solicitud se realiza mediante un objeto JSON con la siguiente estructura:

```
{
 "aerolinea": "AZ",
 "origen": "GIG",
 "destino": "GRU",
 "fecha_partida": "2025-11-10T14:30:00",
 "distancia_km": 350
```

```
}
```

Todos los campos son obligatorios y deben cumplir con los formatos y dominios definidos. En caso de incumplimiento, el sistema responde con un error de validación.

7.5 Contrato de Salida

El servicio devuelve una respuesta estandarizada en formato JSON, que contiene el resultado de la predicción:

```
{
  "prevision": "Retrasado",
  "probabilidad": 0.78
}
```

La probabilidad se expresa como un valor decimal entre 0 y 1, representando el nivel de confianza del modelo en la predicción entregada.

7.6 Manejo de Errores

El sistema contempla un manejo de errores estructurado para garantizar una comunicación clara con el consumidor de la API. Entre los escenarios considerados se incluyen:

- Datos de entrada incompletos o con formato inválido.
- Valores fuera de dominio para variables categóricas.
- Fallas en la obtención de datos climáticos.
- Errores internos del servicio de predicción.

En todos los casos, el sistema retorna respuestas con códigos HTTP adecuados y mensajes descriptivos que facilitan la identificación del problema.

8. API REST – Backend

Esta sección describe la interfaz de programación expuesta por el Back-End de Vektor AI, responsable de recibir solicitudes de predicción, coordinar la ejecución del modelo de Machine Learning y devolver los resultados de forma estandarizada.

La API ha sido diseñada siguiendo principios REST, utilizando formatos de intercambio JSON y códigos HTTP estándar para facilitar su consumo por distintos clientes.

8.1 Endpoint de Predicción

El endpoint principal del sistema permite obtener la predicción de puntualidad de un vuelo a partir de información operativa básica.

Endpoint: POST /predict

Descripción: Recibe los datos de un vuelo, valida la información, obtiene las condiciones climáticas correspondientes y devuelve la predicción generada por el modelo de Machine Learning junto con su probabilidad asociada. La probabilidad corresponde a la salida directa del modelo de Machine Learning, mientras que la confianza representa un indicador complementario utilizado para facilitar la interpretación del resultado.

El sistema soporta tanto predicciones individuales como predicciones por lotes mediante carga de archivos CSV, permitiendo el análisis simultáneo de múltiples vuelos.

8.2 Estructura de la Solicitud

La solicitud debe enviarse en formato JSON e incluir todos los campos obligatorios definidos en el contrato de integración:

```
{
 "aerolinea": "AZ",
 "origen": "GIG",
 "destino": "GRU",
 "fecha_partida": "2025-11-10T14:30:00",
 "distancia_km": 350
}
```

El campo fecha_partida debe respetar el formato ISO 8601. Todos los campos son obligatorios y deben cumplir con los dominios esperados.

8.3 Validación de Entradas

El Back-End realiza validaciones previas a la predicción para garantizar la integridad de los datos. Entre las validaciones implementadas se incluyen:

- Presencia de todos los campos requeridos.
- Validación de formatos de fecha y hora.
- Validación de códigos de aerolínea y aeropuertos.
- Validación de valores numéricos, como la distancia del vuelo.
- Rechazo de solicitudes con valores nulos o inconsistentes.

En caso de error, el sistema retorna una respuesta clara indicando el motivo de la falla.

8.4 Estructura de la Respuesta

La respuesta exitosa del endpoint de predicción se entrega en formato JSON con la siguiente estructura:

```
{
 "prevision": "Retrasado",
 "probabilidad": 0.78
}
```

El campo prevision indica la clasificación final del vuelo, mientras que probabilidad representa el nivel de confianza del modelo, expresado como un valor decimal entre 0 y 1.

8.5 Manejo de Errores y Respuestas

La API utiliza códigos HTTP estándar para representar el resultado de cada solicitud, entre ellos:

- 200 OK: predicción generada correctamente.
- 400 Bad Request: error de validación en los datos de entrada.
- 500 Internal Server Error: error interno del sistema o del servicio de predicción.

Las respuestas de error incluyen mensajes descriptivos que permiten identificar rápidamente la causa del problema.

8.6 Persistencia de Predicciones

Cada solicitud procesada por el endpoint de predicción es registrada junto con su resultado. Esta persistencia permite:

- Auditoría de predicciones realizadas.
- Análisis histórico del comportamiento del sistema.
- Generación de estadísticas agregadas.

9. Persistencia y Almacenamiento

Este conjunto puntualiza el mecanismo de persistencia implementado en Vektor AI para el registro de solicitudes de predicción y sus resultados. La persistencia de datos cumple un rol clave en la trazabilidad del sistema, permitiendo auditoría, análisis histórico y generación de métricas agregadas.

9.1 Objetivo de la Persistencia

El sistema almacena información relacionada con cada predicción realizada con los siguientes objetivos:

- Mantener un registro histórico de las solicitudes procesadas.
- Permitir la auditoría de las decisiones del modelo.
- Facilitar el análisis del comportamiento general del sistema.
- Servir como base para la generación de estadísticas y visualizaciones.

La persistencia no tiene como objetivo reemplazar sistemas operacionales ni actuar como fuente oficial de datos aeronáuticos.

9.2 Información Almacenada

Por cada solicitud de predicción, el sistema registra un conjunto mínimo de información relevante, que incluye:

- Identificador de la solicitud.
- Fecha y hora de la solicitud.
- Datos principales del vuelo enviados por el cliente.
- Variables climáticas utilizadas en la predicción.
- Resultado de la predicción (Puntual / Retrasado).
- Probabilidad asociada a la predicción.

Este conjunto de datos permite reconstruir el contexto en el cual se generó cada predicción.

9.3 Tecnología de Almacenamiento

El sistema utiliza una base de datos relacional para el almacenamiento de la información. La tecnología específica puede variar según el entorno de ejecución, priorizando soluciones ligeras y fáciles de desplegar, coherentes con el enfoque de MVP.

El acceso a la base de datos es gestionado exclusivamente por el Back-End, el cual se encarga de la creación, lectura y consulta de los registros almacenados.

9.4 Uso de la Información Persistida

Los datos almacenados pueden ser utilizados para:

- Generar estadísticas agregadas sobre el comportamiento del sistema.

- Analizar tendencias de retraso en determinados períodos u horarios.
- Apoyar procesos de validación y mejora del modelo en etapas posteriores.
- Visualizar métricas en dashboards u otras interfaces de análisis.

El uso de esta información se limita al análisis interno del sistema y no contempla su exposición directa a consumidores externos sin un proceso adicional de validación.

9.5 Consideraciones de Diseño

La persistencia implementada responde a criterios de simplicidad y trazabilidad. No se contempla en esta versión del sistema la gestión de grandes volúmenes de datos ni optimizaciones avanzadas de rendimiento, dado el carácter exploratorio del proyecto.

Estas decisiones permiten mantener un equilibrio entre funcionalidad, claridad técnica y facilidad de mantenimiento, sentando una base adecuada para futuras extensiones del sistema.

10. Despliegue y Ejecución

Esta sección describe los mecanismos de despliegue y ejecución del sistema Vektor AI en distintos entornos. El sistema ha sido diseñado para ejecutarse de forma consistente tanto en entornos locales como contenerizados y en la nube, manteniendo la misma arquitectura y configuración base.

El despliegue se apoya en el uso de contenedores Docker, lo que permite portabilidad, reproducibilidad y facilidad de mantenimiento.

10.1 Requisitos del Sistema

Para la ejecución del sistema se requieren los siguientes componentes:

- Sistema operativo compatible con Docker.
- Docker y Docker Compose instalados.
- Java Development Kit (JDK) versión 17 o superior.
- Python 3.9 o superior.
- Acceso a internet para la obtención de datos climáticos desde servicios externos.

Estos requisitos aplican tanto para ejecución local como para despliegue en la nube.

10.2 Variables de Entorno

El sistema utiliza variables de entorno para configurar parámetros sensibles y dependientes del entorno de ejecución, tales como:

- Credenciales de acceso a la API climática.
- Parámetros de conexión a la base de datos.
- Configuración de puertos y servicios.

El uso de variables de entorno permite una correcta separación entre configuración y código, facilitando el despliegue en distintos entornos sin modificar la base del sistema.

10.3 Ejecución Mediante Docker Compose

El sistema puede ejecutarse utilizando Docker Compose, lo que permite levantar de forma coordinada todos los componentes del MVP:

- API Back-End.
- Servicio de predicción de Machine Learning.
- Base de datos para persistencia de predicciones.

Docker Compose asegura que los servicios se inicien con las dependencias correctas y se comuniquen entre sí de manera controlada.

10.4 Ejecución Local

De manera alternativa, los componentes del sistema pueden ejecutarse de forma local sin contenedores, siempre que se cumplan los requisitos del sistema y se configuren correctamente las dependencias.

En este modo, el Back-End y el servicio de Machine Learning se ejecutan como procesos independientes, comunicándose a través de interfaces HTTP definidas en el contrato de integración.

10.5 Verificación de Funcionamiento

Una vez desplegado el sistema, su correcto funcionamiento puede verificarse mediante:

- Envío de solicitudes de predicción utilizando herramientas como Postman o cURL.
- Validación de respuestas exitosas y manejo de errores.
- Confirmación de la persistencia de predicciones en la base de datos.
- Revisión de logs generados por los servicios.

Estas acciones permiten validar que el sistema se encuentra operativo y listo para su uso dentro del alcance del MVP.

10.6 Despliegue en Oracle Cloud Infrastructure (OCI)

Vektor AI contempla su despliegue en Oracle Cloud Infrastructure (OCI), utilizando recursos del nivel Always Free, con el objetivo de contar con un entorno accesible para pruebas, demostraciones y validación del sistema en la nube.

El despliegue en OCI se basa en el uso de contenedores Docker, reutilizando la misma configuración definida para entornos locales. Los servicios del sistema se ejecutan sobre una instancia de cómputo, donde se despliegan el Back-End, el servicio de predicción de Machine Learning y la base de datos.

Este despliegue permite:

- Exponer la API de predicción mediante una dirección pública.
- Evaluar tiempos de respuesta y consumo de recursos.
- Validar la estabilidad del sistema en un entorno cercano a producción.

- Facilitar demostraciones sin necesidad de ejecución local.

Dado el enfoque de MVP, no se contemplan configuraciones avanzadas como balanceo de carga, alta disponibilidad o escalamiento automático, las cuales podrán evaluarse en etapas futuras del proyecto.

11. Seguridad y Consideraciones Técnicas

Esta sección describe las principales consideraciones de seguridad y aspectos técnicos transversales del sistema Vektor AI, teniendo en cuenta su alcance como MVP y su evolución hacia entornos productivos.

11.1 Seguridad de la API

La API expuesta por el sistema ha sido diseñada siguiendo principios básicos de seguridad para servicios REST:

- Uso de estructuras JSON bien definidas para entrada y salida.
- Validación estricta de los datos de entrada, evitando solicitudes incompletas o mal formateadas.
- Manejo controlado de errores, evitando exponer información sensible a través de mensajes de respuesta.
- Separación clara entre lógica de negocio, acceso a datos y capa de exposición.

En el contexto del MVP, la API no incorpora mecanismos avanzados de autenticación o autorización, los cuales podrían ser añadidos en una etapa posterior en la evolucion hacia un entorno productivo.

11.2 Manejo de Credenciales y Configuración Sensible

Las credenciales y configuraciones sensibles, como claves de acceso a servicios externos de clima o parámetros de base de datos, no se encuentran embebidas en el código fuente.

Estas configuraciones se gestionan mediante variables de entorno, lo que permite:

- Reducir el riesgo de exposición de información sensible.
- Facilitar el despliegue en distintos entornos (local, contenedores, nube).
- Mantener una separación clara entre código y configuración.

11.3 Seguridad del Modelo de Machine Learning

El modelo predictivo de Machine Learning se utiliza exclusivamente como un componente de inferencia dentro del sistema y no expone directamente su estructura interna al usuario final.

Las predicciones entregadas incluyen únicamente:

- La clasificación del estado del vuelo.
- La probabilidad asociada a dicha clasificación.

Cuando se entrega información de explicabilidad, esta se presenta de forma interpretativa y resumida, evitando la exposición de detalles técnicos que puedan comprometer la integridad del modelo o generar confusión en el usuario.

11.4 Consideraciones de Rendimiento

El sistema ha sido diseñado priorizando un balance entre precisión del modelo y eficiencia computacional. En este contexto:

- Se optó por un modelo Random Forest por su buen desempeño y estabilidad.
- Se descartó el uso de modelos más costosos computacionalmente, como LightGBM, debido a las limitaciones de recursos del entorno objetivo.
- El tiempo de respuesta del endpoint de predicción se mantiene dentro de rangos aceptables para un sistema de consulta en tiempo real.

Estas decisiones permiten que el sistema funcione adecuadamente incluso en entornos con recursos limitados.

11.5 Enfoque de Diseño y Evolución del Sistema

El diseño de Vektor AI prioriza la simplicidad, la estabilidad y la eficiencia, asegurando un funcionamiento confiable dentro del alcance definido para el MVP.

Las decisiones técnicas adoptadas permiten:

- Ejecutar el sistema en entornos con recursos controlados.
- Mantener tiempos de respuesta adecuados para predicciones en tiempo real.
- Facilitar el despliegue, mantenimiento y comprensión del sistema.
- Sentar una base sólida para futuras extensiones funcionales.

La arquitectura modular del sistema permite incorporar de forma progresiva nuevas capacidades, tales como mecanismos avanzados de seguridad, estrategias de escalamiento, reentrenamiento automático del modelo y nuevas fuentes de datos, sin necesidad de rediseñar los componentes existentes.

Este enfoque garantiza que el sistema no solo cumple con los objetivos actuales, sino que también se encuentra preparado para evolucionar según las necesidades del negocio y del entorno operativo.

12. Operación y Uso del Sistema

Esta sección describe el uso operativo de Vektor AI desde el punto de vista funcional, detallando cómo interactuar con el sistema y cómo interpretar sus resultados. El objetivo es facilitar su utilización por parte de usuarios técnicos, equipos de integración o personal encargado de la supervisión del sistema.

12.1 Flujo General de Uso

El flujo de operación del sistema se compone de las siguientes etapas:

1. Recepción de la información del vuelo mediante una solicitud HTTP.

2. Validación de los datos de entrada.
3. Enriquecimiento de la información con datos climáticos en tiempo real.
4. Preprocesamiento de variables según las reglas definidas por el modelo.
5. Ejecución del modelo predictivo de Machine Learning.
6. Generación de la predicción y probabilidad asociada.
7. Persistencia del resultado para fines de auditoría y análisis.
8. Retorno de la respuesta al cliente consumidor de la API.

Este flujo garantiza una predicción consistente y reproducible para cada solicitud.

12.2 Uso del Endpoint de Predicción

El sistema expone un endpoint principal para la predicción del estado del vuelo.

- Método: POST
- Endpoint: /predict

La solicitud debe incluir la información básica del vuelo en formato JSON, cumpliendo con el contrato de integración definido entre los equipos de Data Science y Back-End.

La respuesta del sistema contiene:

- El estado previsto del vuelo (Puntual o Retrasado).
- La probabilidad asociada a dicha predicción, expresada como un valor decimal entre 0 y 1.
- Información complementaria utilizada para la interpretación del resultado, cuando corresponde.

12.3 Interpretación de Resultados

La predicción entregada por Vektor AI representa una estimación probabilística basada en patrones históricos y condiciones actuales, especialmente climáticas.

La probabilidad asociada permite:

- Evaluar el nivel de riesgo de retraso.
- Comparar distintos escenarios operativos.
- Apoyar la toma de decisiones con información cuantitativa.

Cuando se dispone de información de explicabilidad, el sistema presenta los factores más influyentes en la predicción de manera interpretativa y orientada al usuario.

12.4 Visualización y Dashboard

El sistema incluye un dashboard que presenta métricas agregadas en tiempo real, tales como:

- Porcentaje de vuelos previstos como retrasados.

- Distribución de predicciones por franja horaria.
- Impacto de variables climáticas en las predicciones.
- Historial reciente de solicitudes al sistema.

Estas visualizaciones permiten supervisar el comportamiento general del sistema y detectar patrones relevantes para la operación.

12.5 Registro y Auditoría de Predicciones

Todas las predicciones generadas por el sistema son almacenadas para su posterior análisis y auditoría. Este registro permite:

- Revisar el comportamiento histórico del modelo.
- Analizar tendencias de retraso.
- Validar el desempeño del sistema en distintos contextos operativos.
- Facilitar procesos de mejora continua y reporting.

La persistencia de la información fortalece la trazabilidad y confiabilidad del sistema.

13. Despliegue en Oracle Cloud Infrastructure (OCI)

Esta sección describe el despliegue del sistema Vektor AI en Oracle Cloud Infrastructure (OCI), considerando la arquitectura definida y el uso de contenedores Docker como base del proceso. El objetivo de este despliegue es contar con un entorno en la nube que permita ejecutar el sistema de forma remota para pruebas, demostraciones y validación operativa.

13.1 Objetivo del Despliegue en la Nube

El despliegue en OCI permite:

- Ejecutar el sistema en un entorno accesible desde internet.
- Validar el comportamiento del MVP fuera del entorno local.
- Exponer la API de predicción para consumo externo controlado.
- Evaluar rendimiento, estabilidad y consumo de recursos.
- Facilitar demostraciones funcionales del sistema completo.

Este despliegue se enmarca dentro de un enfoque MVP y utiliza recursos compatibles con el nivel Always Free de OCI.

13.2 Arquitectura de Despliegue en OCI

La arquitectura desplegada en OCI replica la arquitectura lógica definida para el sistema, manteniendo la separación por componentes y el enfoque de microservicios.

Los principales servicios desplegados son:

- Frontend (Nginx)
- Backend (Spring Boot)

- Servicio de Machine Learning (FastAPI)
- Base de datos PostgreSQL

Todos los servicios se ejecutan como contenedores Docker y se comunican entre sí mediante una red Docker interna, manteniendo el mismo flujo de datos definido en la arquitectura general del sistema

13.3 Recursos Utilizados en OCI

El despliegue se realiza sobre una instancia de cómputo en OCI, configurada para ejecutar Docker y Docker Compose.

Los recursos utilizados incluyen:

- Instancia Compute (Linux).
- Red virtual (VCN) con acceso público.
- Reglas de seguridad para habilitar el acceso a los puertos necesarios.
- Almacenamiento local para contenedores y volúmenes de datos.

Este enfoque permite un despliegue simple y controlado, adecuado para el alcance del proyecto.

13.4 Preparación del Entorno

Antes del despliegue, se realizan las siguientes acciones:

- Instalación de Docker y Docker Compose en la instancia OCI.
- Configuración de variables de entorno necesarias, incluyendo claves de servicios externos.
- Copia del código fuente y archivos de configuración al servidor.
- Verificación de conectividad de red y acceso a internet.

Estas acciones aseguran que el entorno esté preparado para ejecutar el sistema sin modificaciones adicionales al código.

13.5 Construcción y Ejecución de Contenedores

El sistema se despliega utilizando Docker Compose, reutilizando la misma configuración definida para entornos locales.

Docker Compose permite:

- Construir las imágenes de cada servicio.
- Inicializar los contenedores en el orden correcto.
- Configurar la red interna entre servicios.
- Verificar el estado de salud de cada componente.

Este mecanismo asegura consistencia entre los entornos local y cloud, reduciendo riesgos de configuración.

13.6 Configuración de Variables de Entorno en OCI

Las variables de entorno se configuran directamente en el entorno de ejecución o mediante archivos de configuración utilizados por Docker Compose.

Entre las principales variables se incluyen:

- Claves de acceso a la API climática.
- Configuración de la base de datos.
- Parámetros de red y puertos expuestos.

Este enfoque mantiene separada la configuración del código y facilita cambios entre distintos entornos.

13.7 Exposición y Acceso a la API

Una vez desplegado el sistema, el Backend expone la API REST a través de la dirección pública asignada por OCI.

Los endpoints disponibles y su comportamiento se mantienen idénticos a los definidos en el contrato de integración, garantizando compatibilidad con clientes existentes

El acceso a la API permite:

- Realizar predicciones individuales.
- Ejecutar predicciones por lotes.
- Consultar historial y estadísticas.
- Verificar el estado de salud del sistema.

13.8 Verificación del Despliegue

El correcto funcionamiento del despliegue en OCI se valida mediante:

- Acceso al endpoint de salud del Backend.
- Ejecución de solicitudes de predicción desde clientes externos.
- Revisión de logs de los contenedores.
- Validación de persistencia de predicciones en la base de datos.

Estas verificaciones confirman que el sistema se encuentra operativo en el entorno cloud.

13.9 Consideraciones del Entorno Always Free

El despliegue en OCI se realiza considerando las restricciones propias del nivel Always Free, priorizando un uso eficiente de los recursos disponibles.

Las decisiones de diseño adoptadas permiten:

- Ejecutar el sistema de forma estable.
- Mantener tiempos de respuesta adecuados.

- Operar el MVP sin costos asociados.

La arquitectura modular del sistema permite escalar o adaptar el despliegue a configuraciones más avanzadas en etapas futuras, sin necesidad de rediseñar los componentes principales.

14. Conclusiones Técnicas

El proyecto Vektor AI cumple con el objetivo de desarrollar una solución predictiva funcional para la estimación de retrasos en vuelos, integrando técnicas de Machine Learning, datos operacionales y variables climáticas en tiempo real dentro de una arquitectura moderna y modular.

A lo largo del desarrollo se logró diseñar e implementar un sistema completo, capaz de recibir información de un vuelo, enriquecerla con datos externos, ejecutar un modelo predictivo y entregar una respuesta clara y accionable a través de una API REST estandarizada. Este flujo demuestra la viabilidad técnica de aplicar modelos de clasificación supervisada al problema de la puntualidad aérea.

Desde el punto de vista técnico, el proyecto destaca por la correcta separación de responsabilidades entre los componentes de Data Science y Back-End, el uso de contratos de integración bien definidos y la adopción de contenedores Docker para asegurar portabilidad y consistencia entre entornos. La integración con Oracle Cloud Infrastructure (OCI) refuerza la capacidad del sistema para operar en la nube y ser consumido de forma remota.

El modelo predictivo seleccionado presenta un equilibrio adecuado entre desempeño, estabilidad y eficiencia computacional, permitiendo su ejecución en entornos con recursos controlados sin comprometer la calidad de las predicciones. La incorporación de variables climáticas aporta un valor adicional al modelo y mejora su capacidad explicativa.

Vektor AI se presenta como un Producto Mínimo Viable (MVP) sólido, funcional y extensible, que valida el enfoque tecnológico propuesto y sienta las bases para futuras mejoras. Su diseño modular permite la incorporación progresiva de nuevas capacidades sin necesidad de rediseñar la arquitectura existente.

En conjunto, el proyecto demuestra la aplicación efectiva de buenas prácticas de ingeniería de software y ciencia de datos, entregando una solución coherente, escalable y alineada con necesidades reales del sector aeronáutico.