# Manual de Usuario

## API del Modelo Predictivo de Deserción Estudiantil

Universidad Distrital Francisco José de Caldas — OATI

David Gerardo Díaz Gómez  Cristian Arturo Parra González

July 6, 2026

| | |
|---|---|
| Dirigido a | Personal de Bienestar Institucional y OATI |
| Requiere saber programar | No (salvo la sección 7, opcional) |
| Servicio que documenta | `servicio_api/` del repositorio del proyecto |

## Contenido

**1 Qué es este servicio** **3**

**2 Requisitos previos** **3**

**3 Puesta en marcha** **4**

3.1 Verificar que está corriendo . . . . . 4

3.2 Apagar el servicio . . . . . 5

**4 Cómo pedir una predicción** **5**

4.1 Con el navegador (sin escribir código) . . . . . 5

4.2 Desde otro sistema . . . . . 5

4.3 ¿Cuál endpoint de predicción usar? . . . . . 6

4.4 Datos de entrada . . . . . 6

4.5 Qué devuelve el servicio . . . . . 8

4.6 Ejemplos completos . . . . . 8

4.7 Ejemplo: estudiante recién matriculado . . . . . 10

**5 Reentrenar el modelo** **11**

**6 Solución de problemas** **12**

**7 Uso avanzado: correr sin Docker** **12**

# 1 Qué es este servicio

El modelo predictivo de deserción de la UDFJC estima, para cada estudiante, la probabilidad de que abandone sus estudios, la clasifica en un nivel de riesgo y explica las causas concretas de esa clasificación. Este manual describe cómo usarlo en su forma de **servicio**: una API que corre dentro de un contenedor Docker y a la que cualquier sistema —el navegador, Postman o una aplicación institucional— puede pedirle predicciones mediante mensajes HTTP.

Antes, obtener una predicción exigía ejecutar a mano un script de Python sobre un archivo Excel, desde la terminal de un computador específico. Ahora el mismo modelo, con las mismas métricas, queda disponible como un servicio permanente: se le envía la información de uno o varios estudiantes y responde al instante, sin tocar Excel ni ejecutar Python manualmente.

Tabla 1: Qué cambió con el servicio.

| | Antes | Ahora |
|---|---|---|
| Cómo se pide una predicción | Script de Python a mano | `POST` a `/api/v1/prediccion` |
| Quién puede usarlo | Solo quien tenga el entorno | Cualquier sistema, por HTTP |
| Entrada | Un archivo Excel completo | Un JSON con uno o varios estudiantes |
| Dónde corre | Python local | Contenedor Docker portable |
| Cómo saber si está vivo | No había forma automática | `GET /api/health` |
| Documentación de uso | Leer el código | Interactiva en `/api/swagger` |

# 2 Requisitos previos

Para poner el servicio en marcha se necesitan tres cosas:

1. **Docker Desktop** instalado y abierto, con el motor en ejecución (ícono en verde o mensaje "Engine running"). Sin esto, ninguno de los comandos de este manual funciona.
2. **Los modelos entrenados**: los archivos `fase0.pkl` y `fase1.pkl` en la carpeta `modelos\` de la raíz del proyecto. Si no existen, la sección 5 explica cómo generarlos con un doble clic.

3. **El archivo de configuración** `.env` dentro de `servicio_api\`. No viene incluido a propósito (es local de cada máquina); se crea una única vez copiando la plantilla:

```
cd servicio_api
copy .env.example .env
```

Con los valores por defecto alcanza para el uso local. Solo hay que editar `.env` si se quiere cambiar el puerto (`API_PORT`) o pasar a modo producción (`RUN_MODE=prod`, que deshabilita Swagger y restringe CORS).

# 3 Puesta en marcha

Con los requisitos listos, desde PowerShell en la carpeta `servicio_api\`:

```
docker compose up -d --build
```

El significado de cada parte: `up` levanta el servicio; `-d` lo deja corriendo en segundo plano y devuelve la terminal; `-build` reconstruye la imagen si hubo cambios de código (si no cambió nada puede omitirse y arranca más rápido). La primera vez tarda unos minutos porque descarga las dependencias; las siguientes veces, segundos. Al terminar, la terminal muestra un mensaje del tipo `Container servicio_api-api-desercion-1 Started`.

## 3.1 Verificar que está corriendo

```
docker compose ps
```

Debe listar el contenedor con `STATUS` en `Up ... (healthy)`. Si dice `(health: starting)`, es normal: la primera verificación de salud tarda unos diez segundos; repita el comando un momento después.

La comprobación definitiva es preguntarle directamente al servicio:

```
curl http://localhost:8080/api/health
```

Respuesta esperada:

```
{"Status": "Ok", "Code": "200",
 "ModeloFase0Cargado": true, "ModeloFase1Cargado": true}
```

Si responde `Degraded` con código 503, el servicio está arriba pero no encontró los modelos: verifique que `modelos\fase0.pkl` y `fase1.pkl` existan (sección 5) y que la ruta del volumen en `docker-compose.yml` sea correcta.

### 3.2 Apagar el servicio

```
docker compose down
```

## 4 Cómo pedir una predicción

### 4.1 Con el navegador (sin escribir código)

Abra en el navegador:

```
http://localhost:8080/api/swagger
```

Ahí está la documentación interactiva del servicio. En el endpoint `POST /v1/prediccion`, el botón *Try it out* permite pegar el JSON de un estudiante (como los ejemplos de la sección 4.6), ejecutar la petición y ver la respuesta completa en pantalla. Es la vía recomendada para probar y para demostraciones.

### 4.2 Desde otro sistema

El servicio expone cuatro rutas:

Tabla 2: Endpoints del servicio.

| Método | Ruta | Qué hace |
|---|---|---|
| `GET` | `/api/health` | Verificación de salud: 200 si los modelos están cargados, 503 si no |
| `GET` | `/api/swagger` | Documentación interactiva (solo en modo `dev`) |
| `POST` | `/api/v1/prediccion` | Predicción completa para estudiantes **con historial académico** (cierre de semestre) |
| `POST` | `/api/v1/prediccion-ingreso` | Predicción temprana para estudiantes **recién matriculados**, sin historial académico |

Hay ejemplos listos para usar en `servicio_api\tests\casos_prueba.md` y una colección importable en Postman en `servicio_api\tests\postman_collection.json`, incluyendo casos de error.

## 4.3 ¿Cuál endpoint de predicción usar?

Hay dos puertas de entrada, según la situación del estudiante:

Tabla 3: Cuándo usar cada endpoint de predicción.

| Situación del estudiante | Endpoint |
|---|---|
| Ya cursó al menos un semestre (tiene notas, inasistencias, créditos reprobados) | `/v1/prediccion` |
| Recién se matriculó, todavía no tiene historial académico | `/v1/prediccion-ingreso` |

**Advertencia importante: no use `/v1/prediccion` con `Promedio_Acumulado: 0` para un estudiante nuevo.** El sistema interpretará ese cero como que el estudiante reprobó todo, no como que aún no tiene notas, y el riesgo saldrá inflado (verificado: un mismo estudiante pasa de 41.1 % a 49.2 % solo por ese relleno). Para estudiantes recién matriculados use siempre `/v1/prediccion-ingreso`, que no pide datos académicos.

## 4.4 Datos de entrada

El cuerpo de la petición es un JSON con la lista `estudiantes`; cada estudiante lleva los campos de la tabla 4. Puede enviarse un solo estudiante o varios en la misma petición.

Tabla 4: Campos de entrada por estudiante.

| Campo | Tipo | Descripción |
|---|---|---|
| `COD_ESTUDIANTE` | texto | Identificador del estudiante (se devuelve tal cual en la respuesta) |
| `Edad_Ingreso` | número | Edad al ingresar a la Universidad |
| `Estrato_Homol` | número | Estrato socioeconómico (0 a 6) |
| `Icfes_Matematicas` | número | Puntaje ICFES de matemáticas |
| `Icfes_Biologia` | número | Puntaje ICFES de biología |
| `Icfes_Sociales` | número | Puntaje ICFES de sociales |

| Campo | Tipo | Descripción |
|---|---|---|
| `Icfes_Espanol` | número | Puntaje ICFES de español |
| `Icfes_IdiomaExt` | número | Puntaje ICFES de idioma extranjero |
| `Tipo_Colegio` | 0/1 | 1 = colegio privado, 0 = oficial |
| `Trabaja_Ingreso` | 0/1 | 1 = trabajaba al momento del ingreso |
| `Beca_Activa` | 0/1 | 1 = tiene beca activa |
| `VICTIMA_CONFLICTO` | 0/1 | 1 = víctima del conflicto armado |
| `RURAL` | 0/1 | 1 = procedencia rural |
| `t_tpublico_min` | número | Minutos de viaje en transporte público a la facultad |
| `dist_recta_km` | número | Distancia en línea recta a la facultad, en km |
| `CantEspaciosReprobados` | número | Espacios académicos reprobados acumulados |
| `Promedio_Acumulado` | número | Promedio académico acumulado (sobre 5.0) |
| `CantFallas` | número | Inasistencias acumuladas |
| `TIENE_CONVENIO` | 0/1 | 1 = pertenece a un convenio (Jóvenes a la U, PIC) |
| `Modo_Transporte_Num` | 0/1 | 1 = transporte público, 0 = otro |
| `Facultad` | texto fijo | Uno de los seis valores institucionales (ver abajo) |
| `Sexo` | texto | `M` o `F` |
| `Zona_Residencia` | texto | Zona o localidad de residencia (ej. `Usme`) |
| `MODO_TRANSPORTE` | texto | Medio de transporte (ej. `BUS`, `BICICLETA`) |

El campo `Facultad` no acepta texto libre: debe ser exactamente uno de estos seis valores acordados con las oficinas, que el servicio traduce internamente al formato del modelo:

```
ARTES_ASAB
CIENCIAS_MATEMATICAS_Y_NATURALES
CIENCIAS_Y_EDUCACION
INGENIERIA
TECNOLOGIA_POLITECNICA_TECNOLOGICA
MEDIO_AMBIENTE_Y_RECURSOS_NATURALES
```

Si falta un campo o llega con un valor inválido, el servicio no predice a ciegas: responde con un error que indica qué campo revisar.

## 4.5 Qué devuelve el servicio

Por cada estudiante enviado, la respuesta incluye la predicción completa del sistema —no solo un número—:

Tabla 5: Campos de la respuesta por estudiante.

| Campo | Contenido |
|---|---|
| `COD_ESTUDIANTE` | El mismo identificador enviado |
| `probabilidad` | Probabilidad de deserción, entre 0 y 1 |
| `categoria_riesgo` | Nivel de riesgo: Muy Bajo, Bajo, Medio, Alto o Crítico |
| `factor_1`, `factor_2`, `factor_3` | Las tres causas que más empujan el riesgo de ese estudiante, con su valor real (ej. "Inasistencias acumuladas (12 inasistencias)"). Si el estudiante tiene menos de tres factores de riesgo, los espacios restantes llegan como "-" |
| `accion_sugerida` | Acción recomendada para Bienestar según el nivel |

La tabla 6 resume cómo leer el nivel de riesgo. Importante: el nivel indica *cuándo* actuar, pero los factores indican *por qué* y *cuánto* — dos estudiantes en el mismo nivel pueden necesitar intervenciones distintas, y esa diferencia está en los factores.

Tabla 6: Niveles de riesgo y acción sugerida.

| Nivel | Probabilidad | Prioridad | Acción sugerida |
|---|---|---|---|
| Muy Bajo | 0–15 % | Mínima | Seguimiento estándar semestral |
| Bajo | 15–35 % | Baja | Monitoreo semestral |
| Medio | 35–55 % | Media | Invitar a tutoría académica y revisar carga |
| Alto | 55–75 % | Alta | Remitir a Bienestar; plan de acompañamiento |
| Crítico | 75–100 % | Urgente | Intervención inmediata CADEP + apoyo financiero |

## 4.6 Ejemplos completos

**Caso 1 — perfil de bajo riesgo** (colegio privado, buen ICFES, sin inasistencias, becado):

```
{
```

```
  "estudiantes": [
    {
      "COD_ESTUDIANTE": "EJEMPLO-BAJO-RIESGO",
      "Edad_Ingreso": 18, "Estrato_Homol": 3,
      "Icfes_Matematicas": 70, "Icfes_Biologia": 68,
      "Icfes_Sociales": 65, "Icfes_Espanol": 62,
      "Icfes_IdiomaExt": 60,
      "Tipo_Colegio": 1, "Trabaja_Ingreso": 0, "Beca_Activa": 1,
      "VICTIMA_CONFLICTO": 0, "RURAL": 0,
      "t_tpublico_min": 55, "dist_recta_km": 3.8,
      "CantEspaciosReprobados": 0, "Promedio_Acumulado": 4.5,
      "CantFallas": 0, "TIENE_CONVENIO": 0,
      "Modo_Transporte_Num": 0,
      "Facultad": "CIENCIAS_MATEMATICAS_Y_NATURALES",
      "Sexo": "M", "Zona_Residencia": "Chapinero",
      "MODO_TRANSPORTE": "BICICLETA"
    }
  ]
}
```

**Caso 2 — perfil de alto riesgo** (víctima del conflicto, ICFES bajo, promedio 2.5, seis inasistencias, 145 minutos de transporte):

```
{
  "estudiantes": [
    {
      "COD_ESTUDIANTE": "EJEMPLO-ALTO-RIESGO",
      "Edad_Ingreso": 25, "Estrato_Homol": 1,
      "Icfes_Matematicas": 35, "Icfes_Biologia": 38,
      "Icfes_Sociales": 42, "Icfes_Espanol": 40,
      "Icfes_IdiomaExt": 33,
      "Tipo_Colegio": 0, "Trabaja_Ingreso": 1, "Beca_Activa": 0,
      "VICTIMA_CONFLICTO": 1, "RURAL": 0,
      "t_tpublico_min": 145, "dist_recta_km": 14.7,
      "CantEspaciosReprobados": 3, "Promedio_Acumulado": 2.5,
      "CantFallas": 6, "TIENE_CONVENIO": 0,
      "Modo_Transporte_Num": 1,
      "Facultad": "ARTES_ASAB",
      "Sexo": "F", "Zona_Residencia": "Usme",
```

```
      "MODO_TRANSPORTE": "BUS"
    }
  ]
}
```

Al enviar ambos casos, el servicio responde con probabilidades y niveles claramente distintos, y con factores coherentes con cada perfil — es una buena prueba de que todo está funcionando correctamente de punta a punta.

## 4.7 Ejemplo: estudiante recién matriculado

Para un estudiante sin historial académico se usa `/v1/prediccion-ingreso` (sección 4.3), que solo pide los 13 campos que existen el día de la matrícula — nótese que no lleva promedio, inasistencias ni créditos reprobados:

```
{
  "estudiantes": [
    {
      "COD_ESTUDIANTE": "EST-INGRESO-001",
      "Edad_Ingreso": 18, "Estrato_Homol": 3,
      "Icfes_Matematicas": 55, "Icfes_Biologia": 50,
      "Icfes_Sociales": 58, "Icfes_Espanol": 52,
      "Icfes_IdiomaExt": 48,
      "Tipo_Colegio": 0, "Trabaja_Ingreso": 0, "Beca_Activa": 0,
      "VICTIMA_CONFLICTO": 0, "RURAL": 0,
      "t_tpublico_min": 70
    }
  ]
}
```

Respuesta:

```
{
  "predicciones": [
    {
      "COD_ESTUDIANTE": "EST-INGRESO-001",
      "probabilidad_ingreso": 0.607,
      "categoria_riesgo_ingreso": "Alto",
```

```
      "accion_sugerida": "Remitir a Bienestar; plan de
          acompanamiento"
    }
  ]
}
```

Dos diferencias frente al endpoint principal: los campos de la respuesta se llaman `probabilidad_ingreso` y `categoria_riesgo_ingreso` (para que no se confundan con una predicción de cierre de semestre), y no llegan los factores `factor_1/2/3`, que solo aplican cuando hay historial académico. Los niveles de riesgo y las acciones sugeridas se leen igual que en la tabla 6. El ejemplo completo, listo para copiar, está también en `servicio_api\tests\casos_prueba.md`, sección 2b.

# 5 Reentrenar el modelo

El servicio **no entrena** el modelo: solo lo sirve. Los archivos `fase0.pkl` y `fase1.pkl` se generan aparte, y hay que regenerarlos cuando cambie el histórico de matriculados (típicamente al cierre de cada periodo académico, cuando Urano entrega datos nuevos).

Para eso existe `reentrenar_modelo.bat` en la raíz del proyecto, pensado para usarse sin saber programar:

1. Coloque en `data\entrada\` los reportes de matriculados exportados de Urano (los que tenga, sin importar cuántos años cubran) y el archivo de inasistencias/convenios. No hace falta renombrarlos: el sistema detecta qué es cada archivo por su contenido.
2. Haga doble clic en `reentrenar_modelo.bat`.
3. Espere unos 8 minutos. El script verifica el entorno y los archivos antes de empezar, y avisa con un mensaje claro si falta algo (por ejemplo, si no se ha corrido `setup_windows.bat` o si la carpeta de entrada está vacía).

Al terminar quedan generados en `modelos\`: `fase0.pkl` y `fase1.pkl` (los modelos), `metadata.json` (información de la corrida) y `metricas_entrenamiento.json` (las métricas del modelo recién entrenado, útiles para verificar que el reentrenamiento no degradó el desempeño).

**Paso final imprescindible:** si la API está corriendo en Docker, el modelo nuevo no se carga solo. Hay que reiniciarla para que lo tome:

```
cd servicio_api
docker compose restart
```

Después del reinicio, `curl http://localhost:8080/api/health` debe volver a responder `Ok` con ambos modelos cargados.

# 6 Solución de problemas

Tabla 7: Problemas frecuentes al levantar o usar el servicio.

| Síntoma | Causa probable | Solución |
|---|---|---|
| `env file ... not found` | Falta el `.env` | `copy .env.example .env` dentro de `servicio_api\` |
| `error during connect ...` | Docker Desktop cerrado o iniciando | Abrirlo y esperar el ícono verde |
| `port is already allocated` | El puerto 8080 está ocupado | `docker compose down` primero, o cambiar `API_PORT=8081` en `.env` |
| `ps` muestra `(unhealthy)` | No encuentra los `.pkl` | Verificar `modelos\fase0.pkl` y `fase1.pkl` (sección 5) y la ruta del volumen |
| `up` se queda "pegado" | Faltó `-d` | Ctrl+C y volver a correr con `-d` |
| Cambié código y no se refleja | Imagen vieja | `docker compose up -d -build` |
| Health responde `Degraded`/503 | Modelos no cargados | Igual que `(unhealthy)`: revisar los `.pkl` y reiniciar |

Para ver qué está pasando por dentro en cualquier momento:

```
docker compose logs -f
```

# 7 Uso avanzado: correr sin Docker

Para una prueba rápida sin levantar el contenedor (requiere el entorno virtual del repositorio principal ya instalado):

```
cd servicio_api
$env:RUTA_MODELOS = "..\modelos"
$env:RUN_MODE = "dev"
```

```
..\.venv\Scripts\python.exe api.py
```

La API queda igualmente en `http://localhost:8080`, con Swagger en `/api/swagger`. Esta vía es solo para pruebas locales; para cualquier uso compartido o permanente debe usarse el contenedor Docker de la sección 3.