

# Flight Hub - Arquitectura Basada en Dominios

## Índice

- 1. [Visión General](#)
- 2. [Arquitectura Actual vs Nueva](#)
- 3. [Componentes Core](#)
- 4. [Identificación de Vuelos](#)
- 5. [Modelo de Base de Datos por Dominios](#)
- 6. [Flujo de Datos](#)
- 7. [Precedencias](#)
- 8. [Stack Tecnológico](#)

## Visión General

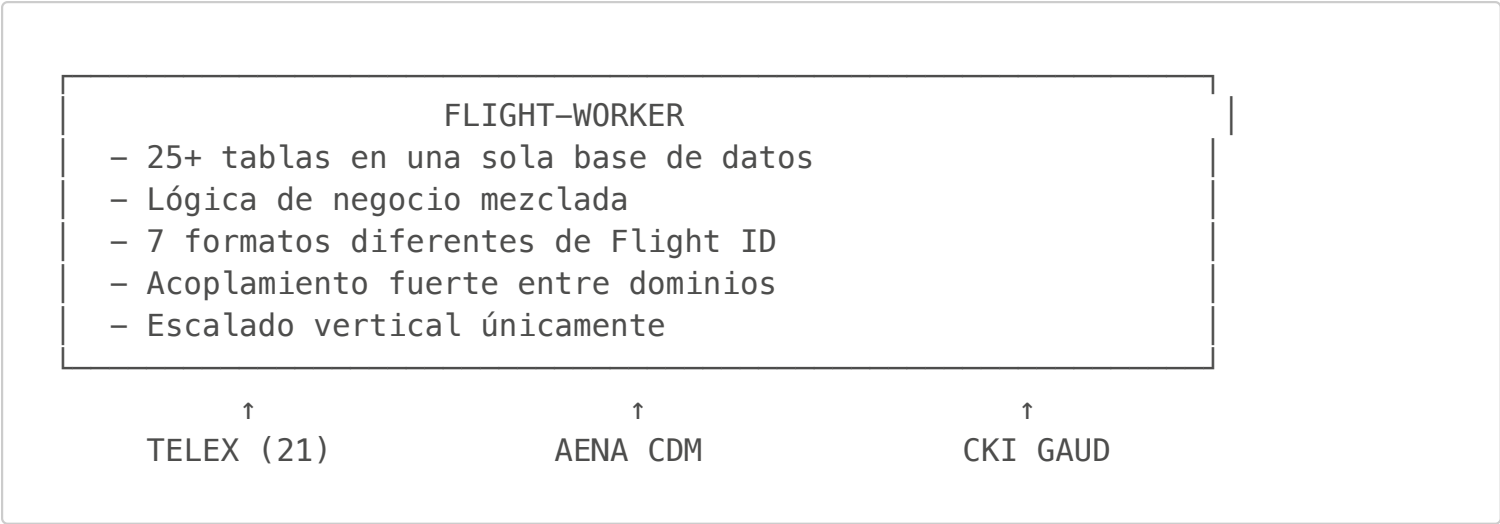
La nueva arquitectura transforma el sistema monolítico actual en una arquitectura basada en **dominios de negocio** independientes, cada uno con su propia base de datos, siguiendo los principios de microservicios y Domain-Driven Design (DDD).

### Principios Fundamentales

- 1. **Un dominio, una base de datos:** Cada microservicio solo escribe en su propia base de datos
- 2. **Comunicación por eventos:** Las integraciones entre dominios se hacen SOLO mediante eventos
- 3. **FUID como identificador único:** Reemplaza los múltiples formatos de Flight ID actuales
- 4. **Matching flexible:** Identificación de vuelos usando múltiples criterios (IATA/ICAO, diferentes campos)
- 5. **Precedencias por campo:** Cada campo tiene sus propias reglas de precedencia por fuente

## Arquitectura Actual vs Nueva

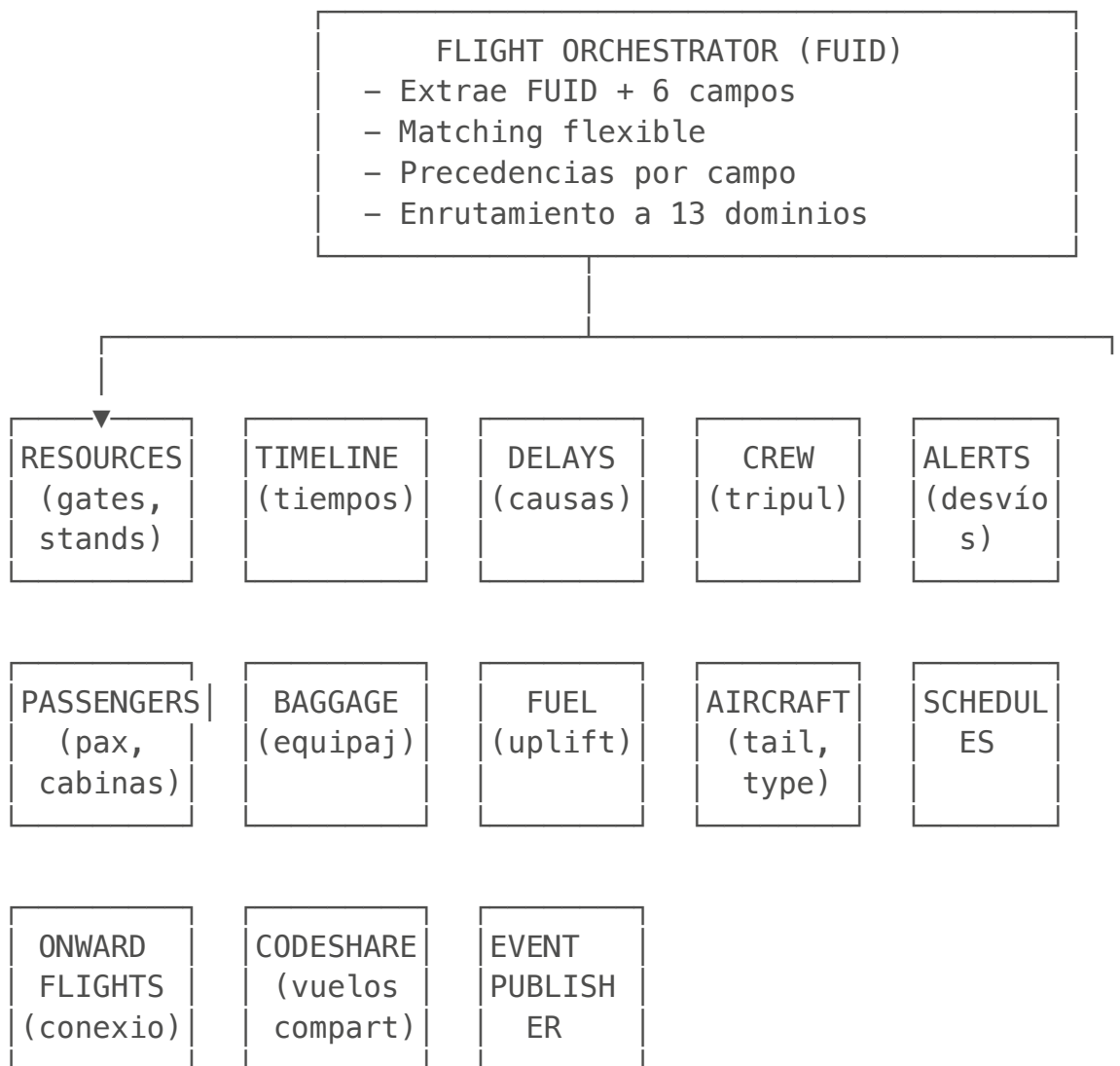
### Arquitectura Actual (Monolítica)



### Problemas:

- ❌ Imposible escalar dominios independientemente
- ❌ Deploys riesgosos (todo o nada)
- ❌ Tabla `flight_departure_info` con 100+ campos mezclando 4 dominios
- ❌ 7 formatos de Flight ID incompatibles entre servicios
- ❌ Precedencias hardcodeadas y difíciles de mantener

## Nueva Arquitectura (Basada en Dominios)



Cada dominio:

- ✓ Base de datos propia (PostgreSQL)
- ✓ Guarda FUID + 6 campos en cada tabla
- ✓ Aplica precedencias por campo
- ✓ Publica eventos a Event Publisher

## Beneficios:

- ✅ Escalado independiente por dominio
- ✅ Deploys independientes
- ✅ FUID único en todo el sistema
- ✅ Matching flexible (IATA/ICAO, múltiples campos)
- ✅ Precedencias configurables por campo

-  Ownership claro de datos

## Componentes Core

### 1. Servicios IN y Parsers

La arquitectura sigue un patrón **IN** → **Parser** → **Orchestrator** donde cada fuente de datos tiene:

1. Un servicio **IN** que recibe mensajes raw
2. Un **Parser** que parsea y guarda en su BD
3. Publica a la **cola del Orchestrator**

#### 1.1 Telex-IN → Telex Parser

**Servicio IN:** **telex-in** recibe mensajes raw desde SITA

**Parser:** **telex-parser** procesa 21+ tipos de mensajes telex

```
TELEX-IN: Recibir mensaje raw desde SITA
```

FUNCIÓN receiveTelexMessage(rawTelex)

```
// 1. Guardar mensaje raw en telex_in_messages
messageId ← GUARDAR_EN_DB(telex_in_messages, {
  rawMessage: rawTelex,
  source: 'SITA',
  receivedAt: AHORA()
})

// 2. Publicar CloudEvent a parser queue (formato text/plain)
PUBLICAR_A_COLA(
  cola: 'telex_parser_queue.fifo',
  messageGroupId: 'telex-parser', // Procesar en orden
  mensaje: {
    specversion: "1.0",
    type: "com.iberia.flighthub.telex_in",
    source: "FlightHub.TelexIn",
    subject: "TELEX RAW",
    id: messageId,
    time: ISO_TIMESTAMP(AHORA()),
    datacontenttype: "text/plain",
    data: rawTelex // ← Raw telex text
  }
)
```

FIN FUNCIÓN

## TELEX-PARSER: Procesar mensaje CloudEvent

**FUNCIÓN** parse(cloudEvent)

```
rawTelex ← cloudEvent.data // Extraer raw text

// 1. Detectar tipo y subtipo usando regex patterns
detección ← DETECTAR_TIPO_TELEX(rawTelex)
// Ejemplo: { type: 'MVT', subtype: 'AD' }

// 2. Parsear mensaje según tipo/subtipo
parsedData ← PARSEAR_POR_TIPO(rawTelex, detección.type,
detección.subtype)

// 3. Guardar en telex_parser (tabla del parser con datos parseados)
telexId ← GUARDAR_EN_DB(telex_parser, {
  messageId: cloudEvent.id,
  rawMessage: rawTelex,
  type: detección.type,
  subtype: detección.subtype,
  parsedData: parsedData,
  source: 'SITA'
})

// 4. Publicar CloudEvent a orchestrator queue
PUBLICAR_A_COLA(
  cola: 'orchestrator_queue.fifo',
  messageGroupId: 'telex-parser', // ← Procesamiento secuencial del
parser
  mensaje: {
    specversion: "1.0",
    type: "com.iberia.flighthub.telex." + detección.type,
    source: "FlightHub.TelexParser",
    subject: detección.type + " " + detección.subtype,
    id: telexId,
    time: ISO_TIMESTAMP(AHORA()),
    datacontenttype: "application/json",
    data: parsedData // ← Solo datos parseados (NO FUID, NO AnchorKey)
  }
)
```

**FIN FUNCIÓN**

**Tipos soportados** (21+ tipos):

- MVT, ASM, LDM, CPM, PSM, WAB, SAM, SRM, SLC, ARR, DEP, M11-M14, MER, AME

## 1.2 Nimbus-In → Nimbus Parser

**Servicio IN:** `nimbus-in` recibe mensajes de Nimbus (fuel data)

**Parser:** nimbus-parser procesa y guarda en fh\_nimbus

### NIMBUS-IN: Recibir mensaje desde Nimbus (XML/JSON)

**FUNCIÓN** receiveNimbusMessage(rawMessage, contentType)

```
// 1. Guardar mensaje raw en nimbus_in_messages
messageId ← GUARDAR_EN_DB(nimbus_in_messages, {
  rawMessage: rawMessage,
  source: 'NIMBUS',
  receivedAt: AHORA()
})

// 2. Publicar CloudEvent a parser queue (formato XML o JSON)
PUBLICAR_A_COLA(
  cola: 'nimbus_parser_queue.fifo',
  messageGroupId: 'nimbus-parser',
  mensaje: {
    specversion: "1.0",
    type: "com.iberia.flighthub.nimbus_in",
    source: "FlightHub.NimbusIn",
    subject: "FUEL DATA",
    id: messageId,
    time: ISO_TIMESTAMP(AHORA()),
    datacontenttype: contentType, // "application/xml" o
"application/json"
    data: rawMessage
  }
)
```

FIN FUNCIÓN

## NIMBUS-PARSER: Procesar mensaje CloudEvent

## FUNCIÓN `parse(cloudEvent)`

```

rawMessage ← cloudEvent.data

// 1. Parsear según content type
SI cloudEvent.datacontenttype == "application/xml" ENTONCES
    parsedData ← PARSEAR_FUEL_XML(rawMessage)
SINO
    parsedData ← PARSEAR_FUEL_JSON(rawMessage)
FIN SI

// 2. Guardar en nimbus_parser (tabla del parser con datos parseados)
nimbusId ← GUARDAR_EN_DB(nimbus_parser, {
    messageId: cloudEvent.id,
    rawMessage: rawMessage,
    parsedData: parsedData,
    source: 'NIMBUS'
})

// 3. Publicar CloudEvent a orchestrator queue
PUBLICAR_A_COLA(
    cola: 'orchestrator_queue.fifo',
    messageGroupId: 'nimbus-parser', // ← Procesamiento secuencial del
parser
    mensaje: {
        specversion: "1.0",
        type: "com.iberia.flighthub.fuel.nimbus",
        source: "FlightHub.NimbusParser",
        subject: "FUEL UPDATE",
        id: nimbusId,
        time: ISO_TIMESTAMP(AHORA()),
        datacontenttype: "application/json",
        data: parsedData // ← Solo datos parseados (NO FUID, NO AnchorKey)
    }
)

FIN FUNCIÓN

```

### Arquitectura y Tablas por Servicio:

SITA → telex-in (guarda en telex\_in\_messages) → telex-parser (parsea y guarda en telex\_parser) → orchestrator\_queue.fifo

NIMBUS → nimbus-in (guarda en nimbus\_in\_messages) → nimbus-parser (parsea y guarda en nimbus\_parser) → orchestrator\_queue.fifo

AENA → aena-in (guarda en aena\_in\_messages) → aena-parser (parsea y guarda en aena\_parser) → orchestrator\_queue.fifo

CKI → cki-in (guarda en cki\_in\_messages) → cki-parser (parsea y guarda en cki\_parser) → orchestrator\_queue.fifo

```
SSIM → ssim-in (guarda en ssim_in_messages) → ssim-parser (parsea y guarda en ssim_parser) → orchestrator_queue.fifo
```

### Patrón de nombrado de tablas:

- `{servicio}_in_messages` - Servicios IN guardan mensajes raw
- `{servicio}_parser` - Parsers guardan datos parseados
- Orchestrator añade FUID y enruta a dominios
- Workers guardan en tablas con prefijo `fh_{dominio}_`

## 2. FUID: Identificador Único de Vuelo

**FUID (Flight Unique Identifier)** es el identificador interno permanente del vuelo:

### Características del FUID

- **Generado por:** Orchestrator (solo para ASM NEW y SSIM NEW)
- **Formato:** ULID (26 caracteres) - Ejemplo: `01HQZ8X9Y1K2M3N4P5Q6R7S8T9`
- **Propósito:** Vincular todos los datos del mismo vuelo, incluso cuando cambian los identificadores externos (aeropuerto, departure\_number, etc.)
- **Inmutable:** Una vez creado, nunca cambia
- **Alcance:** Orchestrator → Domain Workers → Event Publisher
- **NO se publica** a EventBridge (solo los 6 campos externos)

### Flujo Completo del FUID

#### Parser:

- Parsea mensaje
- NO genera ningún identificador
- Publica a `orchestrator_queue.fifo`
- Incluye solo datos parseados (NO FUID)
- FIFO garantizado por la cola SQS

#### Orchestrator:

- Recibe mensaje parseado del Parser
- Extrae los 6 campos del mensaje
- Busca/crea FUID usando los 6 campos
- Guarda FUID en `flights` y `message_log`
- Publica a `domain queues`
- Incluye FUID + 6 campos en `cloudEvent.data`
- FIFO garantizado por las colas SQS FIFO

#### Domain Workers:

- Reciben FUID + 6 campos
- Guardan en BD con FUID + 6 campos (para vincular datos)
- Publican a Event Publisher con FUID + 6 campos
- FIFO garantizado por las colas SQS FIFO

#### Event Publisher:

- Recibe FUID + 6 campos
- Publica a EventBridge solo los 6 campos (NO FUID)

### Ejemplo de vinculación con FUID:

Vuelo IB347 MAD-BCN:

- Primer intento: departure\_number = 1, FUID = 01HQZ8X9...
- Return to base por problema técnico
- Segundo intento: departure\_number = 2, MISMO FUID = 01HQZ8X9...

→ El FUID vincula ambos intentos como el mismo vuelo operacional

## 3. Flight Orchestrator

**Responsabilidad:** Recibir mensajes parseados desde los Parsers, extraer FUID + 6 campos con entrada específica por tipo de dato, gestionar precedencias, y enrutar a dominios.

### Base de Datos:

#### Tabla **flights** (Identificadores)

```
CREATE TABLE flights (  
  -- Identificador único permanente  
  fuid VARCHAR(26) PRIMARY KEY,  
  
  -- Los 6 campos de identificación (para matching)  
  operation_date DATE NOT NULL,  
  flight_designator VARCHAR(10) NOT NULL,  
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',  
  airline_designator VARCHAR(3) NOT NULL,  
  departure_airport VARCHAR(3) NOT NULL,  
  departure_number INTEGER NOT NULL DEFAULT 1,  
  
  -- Códigos ICAO adicionales para matching flexible  
  airline_designator_icao VARCHAR(4),  
  flight_designator_atc VARCHAR(10),  
  departure_airport_icao VARCHAR(4),  
  arrival_airport VARCHAR(3),  
  arrival_airport_icao VARCHAR(4),  
  
  -- Control de estado  
  active BOOLEAN NOT NULL DEFAULT true,  
  principal BOOLEAN NOT NULL DEFAULT true,  
  fuid_new_flight VARCHAR(26),  
  fuid_flight_principal VARCHAR(26),  
  
  -- Auditoría  
  created_at TIMESTAMP NOT NULL DEFAULT NOW(),
```



```

created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
updated_by VARCHAR NOT NULL,

-- Índice compuesto único para los 6 campos
UNIQUE INDEX idx_6_campos (
    operation_date,
    flight_designator,
    operational_suffix,
    airline_designator,
    departure_airport,
    departure_number
),

-- Índices para búsquedas
INDEX idx_operation_date (operation_date),
INDEX idx_airline (airline_designator),
INDEX idx_flight_designator (flight_designator),
INDEX idx_departure_airport (departure_airport),
INDEX idx_active (active),
INDEX idx_principal (principal)
);

```

### Tabla `message_log` (Auditoría Completa)

```

CREATE TABLE message_log (
    id UUID PRIMARY KEY,
    fuid VARCHAR(26) REFERENCES flights(fuid),

    source VARCHAR(50) NOT NULL,           -- "TELEX", "AENA", "CKI", etc.
    message_type VARCHAR(50) NOT NULL,     -- "MVT", "ASM", "CDM", etc.
    message_subtype VARCHAR(50),           -- "NEW", "UPDATE", "CANCEL"

    raw_message JSONB NOT NULL,            -- Mensaje original completo
    parsed_message JSONB,                  -- Mensaje parseado

    processing_status VARCHAR(20),         -- PENDING, PROCESSED, FAILED
    processed_at TIMESTAMP,
    error_message TEXT,

    received_at TIMESTAMP DEFAULT NOW(),

    INDEX idx_fuid (fuid),
    INDEX idx_source_type (source, message_type),
    INDEX idx_status (processing_status)
);

```

### Extracción de FUID + 6 Campos:

## ORCHESTRATOR: Extractores específicos por tipo de mensaje

```
// Extractor para mensajes TELEX (21+ tipos)
FUNCIÓN extractTelexIdentifiers(parsedData, messageType)

// El telex-parser ya parseó el mensaje, solo extraemos identificadores
RETORNAR {
  operationDate: parsedData.flightDate 0 parsedData.operationDate,
  flightDesignator: parsedData.flightNumber,
  operationalSuffix: parsedData.suffix 0 "",
  airlineDesignator: parsedData.airline,
  departureAirport: parsedData.departureAirport,
  departureNumber: parsedData.departureNumber 0 1
}

FIN FUNCIÓN

// Extractor para mensajes NIMBUS
FUNCIÓN extractNimbusIdentifiers(parsedData)

// El nimbus-parser ya parseó los datos de fuel
RETORNAR {
  operationDate: parsedData.operationDate,
  flightDesignator: parsedData.flightNumber,
  operationalSuffix: parsedData.suffix 0 "",
  airlineDesignator: parsedData.airline,
  departureAirport: parsedData.departureAirport,
  departureNumber: 1
}

FIN FUNCIÓN

// Extractor para mensajes AENA CDM
FUNCIÓN extractAENAIdentifiers(parsedData)

// El aena-parser ya parseó el mensaje CDM
RETORNAR {
  operationDate: parsedData.operationDate,
  flightDesignator: parsedData.flightId,
  operationalSuffix: parsedData.operationalSuffix 0 "",
  airlineDesignator: parsedData.airlineIATA,
  departureAirport: parsedData.adept,
  departureNumber: 1
}
```

FIN FUNCIÓN

```
ORCHESTRATOR SERVICE: Procesamiento principal
```

FUNCIÓN processMessage(cloudEvent)

```
// 1. Seleccionar extractor según el tipo de mensaje
SI cloudEvent.type CONTIENE "telex" ENTONCES
    identifiers ← extractTelexIdentifiers(cloudEvent.data,
cloudEvent.subject)
SINO SI cloudEvent.type CONTIENE "nimbus" ENTONCES
    identifiers ← extractNimbusIdentifiers(cloudEvent.data)
SINO SI cloudEvent.type CONTIENE "aena" ENTONCES
    identifiers ← extractAENAIdentifiers(cloudEvent.data)
// ... otros extractores
FIN SI

// 2. Buscar o crear FUID usando los 6 campos
fuid ← findOrCreateFUID(identifiers, cloudEvent)

// 3. Guardar identificadores en orchestrator.flights
GUARDAR_IDENTIFICADORES(fuid, identifiers)

// 4. Guardar mensaje completo en orchestrator.message_log (auditoría)
GUARDAR_EN_DB(message_log, {
    fuid: fuid,
    source: cloudEvent.source,
    messageType: cloudEvent.type,
    rawMessage: cloudEvent,
    processedAt: AHORA()
})

// 5. Enrutar a dominios con FUID + 6 campos
routeToDomains(fuid, identifiers, cloudEvent)
```

FIN FUNCIÓN

### Matching Flexible:

```
MATCHING FLEXIBLE: Buscar candidatos de vuelo
```

FUNCIÓN findFlightCandidates(identifiers)

```
// Buscar vuelos que coincidan con los 6 campos (flexible matching)
candidatos ← CONSULTAR_DB(flights, DONDE {

    // 1. Aerolínea (IATA 0 ICAO)
    (airline_designator = identifiers.airlineDesignator
    0 airline_designator_icao = identifiers.airlineDesignator)

Y
// 2. Flight number (normal 0 ATC)
(flight_designator = identifiers.flightDesignator
0 flight_designator_atc = identifiers.flightDesignator)

Y
// 3. Aeropuertos (departure IATA 0 ICAO)
(departure_airport = identifiers.departureAirport
0 departure_airport_icao = identifiers.departureAirport)

Y
// 4. Fechas (UTC 0 local)
(operation_date = identifiers.operationDate
0 local_operation_date = identifiers.operationDate)

Y
// 5. Operational suffix
operational_suffix = identifiers.operationalSuffix

Y
// 6. Departure number
departure_number = identifiers.departureNumber
})

RETORNAR candidatos

FIN FUNCIÓN
```

### Creación de FUID:

VALIDACIÓN: Verificar si puede crear FUID

FUNCIÓN canCreateFUID(cloudEvent)

```
// SOLO ASM NEW y SSIM NEW pueden crear FUID
SI cloudEvent.source CONTIENE "TelexParser" ENTONCES
```

```

// Verificar si es ASM NEW
RETORNAR cloudEvent.subject CONTIENE "ASM" Y cloudEvent.subject
CONTIENE "NEW"
FIN SI

SI cloudEvent.source CONTIENE "SSIMParser" ENTONCES
// Verificar si es SSIM NEW
RETORNAR cloudEvent.type CONTIENE "NEW"
FIN SI

RETORNAR FALSO // Otros tipos no pueden crear FUID

FIN FUNCIÓN

```

```

CREAR VUELO: Generar nuevo FUID y guardar identificadores

```

```

FUNCIÓN createFlight(identifiers, cloudEvent)

// 1. Generar FUID único (ULID)
fuid ← GENERAR_ULID()
// Ejemplo: "01HQZ8X9Y1K2M3N4P5Q6R7S8T9"

// 2. Extraer datos del mensaje parseado
parsedData ← cloudEvent.data

// 3. Guardar en orchestrator.flights
GUARDAR_EN_DB(flights, {
  fuid: fuid,

  // Los 6 campos de identificación
  airline_designator: identifiers.airlineDesignator,
  airline_designator_icao: BUSCAR_ICAO(identifiers.airlineDesignator),
  flight_designator: identifiers.flightDesignator,
  operational_suffix: identifiers.operationalSuffix,
  operation_date: identifiers.operationDate,
  departure_airport: identifiers.departureAirport,
  departure_airport_icao:
    BUSCAR_ICAO_AIRPORT(identifiers.departureAirport),
  departure_number: identifiers.departureNumber,

  // Datos adicionales
  arrival_airport: EXTRAER(parsedData, 'arrivalAirport'),
  scheduled_departure_time: EXTRAER(parsedData, 'scheduledDeparture'),
  scheduled_arrival_time: EXTRAER(parsedData, 'scheduledArrival'),

  // Metadata
  created_by: cloudEvent.type,

```

```
    created_at: AHORA(),  
    updated_at: AHORA()  
  })
```

RETORNAR fuid

FIN FUNCIÓN

### Enrutamiento a Dominios:

El Orchestrator enruta mensajes parseados a los dominios correspondientes. Un mensaje puede ir a **múltiples dominios** simultáneamente.

**Concepto:** El Orchestrator extrae datos específicos de cada dominio y publica eventos CloudEvents 1.0 a las colas FIFO de cada dominio. El orden está garantizado por las colas SQS FIFO.

**Ver ejemplo completo** en la sección [Flujo de Datos](#) que muestra cómo un mensaje MVT se enruta a 3 dominios (Timeline, Passengers, Fuel) con formato CloudEvents completo.

### 4. Event Publisher (Publicación Externa)

**Responsabilidad:** Agregar cambios de dominios y publicar eventos a sistemas externos con el formato de identificación apropiado.

**Importante:** El FUID es para uso **interno** del sistema. Para publicar a sistemas externos, se usan los **6 campos de identificación** que son estándar en la industria aeronáutica. **El FUID NO se publica a EventBridge.**

### Campos de Identificación Externa

```
interface ExternalFlightIdentifier {  
  operationDate: Date; // Fecha de operación UTC  
  flightDesignator: string; // Número de vuelo: "347"  
  operationalSuffix: string; // Sufijo operacional: "A", "B", ""  
  airlineDesignator: string; // Aerolínea IATA: "IB"  
  departureAirport: string; // Aeropuerto de salida IATA: "MAD"  
  departureNumber: number; // Número de intento de despegue: 1, 2, 3...  
}
```

### Explicación de Campos:

1. **operationDate:** Fecha UTC de operación del vuelo
2. **flightDesignator:** Número de vuelo (sin prefijo de aerolínea)
3. **operationalSuffix:** Sufijo usado en casos de vuelos duplicados programados ("A", "B", o vacío)
4. **airlineDesignator:** Código IATA de la aerolínea (2-3 caracteres)
5. **departureAirport:** Código IATA del aeropuerto de salida
6. **departureNumber:** Número incremental para vuelos turnaround (cuando el mismo vuelo despegue múltiples veces)

## Relación FUID ↔ External Identifier:

```
interface FlightMapping {  
  // Identificador interno (único, inmutable)  
  fuid: string; // "01HQZ8X9Y1K2M3N4P5Q6R7S8T9"  
  
  // Identificadores externos (pueden cambiar con turnarounds)  
  externalId: ExternalFlightIdentifier;  
  
  // Referencia al anterior (para turnarounds)  
  previousDepartureNumber?: number;  
}
```

## Tabla en Flight Orchestrator:

```
CREATE TABLE flights (  
  -- Identificador único permanente  
  fuid VARCHAR(26) PRIMARY KEY,  
  
  -- Los 6 campos de identificación (para matching)  
  operation_date DATE NOT NULL,  
  flight_designator VARCHAR(10) NOT NULL,  
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',  
  airline_designator VARCHAR(3) NOT NULL,  
  departure_airport VARCHAR(3) NOT NULL,  
  departure_number INTEGER NOT NULL DEFAULT 1,  
  
  -- Códigos ICAO adicionales para matching flexible  
  airline_designator_icao VARCHAR(4),  
  flight_designator_atc VARCHAR(10),  
  departure_airport_icao VARCHAR(4),  
  arrival_airport VARCHAR(3),  
  arrival_airport_icao VARCHAR(4),  
  
  -- Control de estado  
  active BOOLEAN NOT NULL DEFAULT true,  
  principal BOOLEAN NOT NULL DEFAULT true,  
  fuid_new_flight VARCHAR(26),  
  fuid_flight_principal VARCHAR(26),  
  
  -- Auditoría  
  created_at TIMESTAMP NOT NULL DEFAULT NOW(),  
  created_by VARCHAR NOT NULL,  
  updated_at TIMESTAMP NOT NULL DEFAULT NOW(),  
  updated_by VARCHAR NOT NULL,  
  
  -- Índice compuesto único para los 6 campos  
  UNIQUE INDEX idx_6_campos (  
    operation_date,  
    flight_designator,
```

```

        operational_suffix,
        airline_designator,
        departure_airport,
        departure_number
    ),

    -- Índices para búsquedas
    INDEX idx_operation_date (operation_date),
    INDEX idx_airline (airline_designator),
    INDEX idx_flight_designator (flight_designator),
    INDEX idx_departure_airport (departure_airport),
    INDEX idx_active (active),
    INDEX idx_principal (principal)
);

```

## Event Publisher: Uso de los 6 Campos (Sin FUID)

EVENT PUBLISHER: Recibir evento de dominio y publicar a EventBridge

**FUNCIÓN** publishDomainEvent(cloudEvent)

```

// 1. Extraer los 6 campos que ya vienen en el evento
// NO se usa FUID en publicación externa
externalId ← {
    operationDate: cloudEvent.data.operationDate,
    flightDesignator: cloudEvent.data.flightDesignator,
    operationalSuffix: cloudEvent.data.operationalSuffix 0 "",
    airlineDesignator: cloudEvent.data.airlineDesignator,
    departureAirport: cloudEvent.data.departureAirport,
    departureNumber: cloudEvent.data.departureNumber
}

// 2. Extraer el campo "changes" (trigger de publicación)
// El Worker de dominio envía qué campos cambiaron
changes ← cloudEvent.data.changes
// Ejemplo: ["totalPassengers", "checkedInPassengers"]

// 3. Construir payload para EventBridge (sin FUID)
payload ← buildPayload(cloudEvent, externalId, changes)

// 4. Publicar CloudEvent a EventBridge
PUBLICAR_A_EVENTBRIDGE({
    EventBusName: "flight-events-bus",
    Source: "com.iberia.flighthub",
    DetailType: cloudEvent.type,
    Detail: JSON_STRINGIFY(payload),

```



```

Resources: [
  "flight:" + externalId.airlineDesignator + ":" +
externalId.flightDesignator,
  "airport:" + externalId.departureAirport,
  "date:" + externalId.operationDate
]
})

```

FIN FUNCIÓN

BUILD PAYLOAD: Construir payload adaptado por dominio

FUNCIÓN buildPayload(cloudEvent, externalId, changes)

```

// Payload base (común a todos los dominios)
basePayload ← {
  // Identificación externa con los 6 campos (NO incluye FUID)
  flightIdentifier: externalId,

  // Timestamp del cambio
  timestamp: cloudEvent.time,

  // Fuente del cambio
  source: cloudEvent.source,

  // Tipo de evento
  eventType: cloudEvent.type,

  // Campos que cambiaron (trigger)
  changes: changes
}

// Agregar datos específicos del dominio
SI cloudEvent.type CONTIENE "passengers" ENTONCES
  RETORNAR {
    ...basePayload,
    passengers: {
      total: cloudEvent.data.total_passengers,
      checkedIn: cloudEvent.data.checked_in_passengers,
      boarded: cloudEvent.data.boarded_passengers,
      cabins: cloudEvent.data.cabin_configurations
    }
  }
}

SINO SI cloudEvent.type CONTIENE "fuel" ENTONCES
  RETORNAR {
    ...basePayload,

```

```

        fuel: {
            uplift: cloudEvent.data.fuel_uplift,
            planned: cloudEvent.data.fuel_planned,
            remaining: cloudEvent.data.fuel_remaining
        }
    }

    SINO SI cloudEvent.type CONTIENE "timeline" ENTONCES
        RETORNAR {
            ...basePayload,
            times: {
                scheduledDeparture: cloudEvent.data.scheduled_departure_time,
                estimatedDeparture: cloudEvent.data.estimated_departure_time,
                actualDeparture: cloudEvent.data.actual_departure_time,
                gate: cloudEvent.data.departure_gate
            }
        }

    // ... otros dominios siguen el mismo patrón

    FIN SI

    FIN FUNCIÓN

```

### Ejemplo Completo: Domain Worker → Event Publisher

```

PASSENGERS SERVICE: Procesar evento de dominio

```

```

FUNCIÓN processPassengersEvent(cloudEvent)

```

```

    // 1. Extraer datos del CloudEvent
    fuid ← cloudEvent.data.fuid
    identifiers ← EXTRAER_6_CAMPOS(cloudEvent.data)
    newData ← cloudEvent.data

```

```

    // 2. Leer datos actuales de la BD
    currentData ← LEER_DB(fh_pax_summary, fuid)

```

```

    // 3. Aplicar precedencias y detectar cambios
    changes ← []
    dataToUpdate ← {}

```

```

    SI PUEDE_ACTUALIZAR("total_passengers", currentData.source,
cloudEvent.source) ENTONCES

```

```

        SI newData.total_passengers != currentData.total_passengers ENTONCES
            dataToUpdate.total_passengers ← newData.total_passengers

```

```

    AGREGAR(changes, "total_passengers")
  FIN SI
FIN SI

SI PUEDE_ACTUALIZAR("checked_in_passengers", currentData.source,
cloudEvent.source) ENTONCES
  SI newData.checked_in_passengers != currentData.checked_in_passengers
ENTONCES
    dataToUpdate.checked_in_passengers ← newData.checked_in_passengers
    AGREGAR(changes, "checked_in_passengers")
  FIN SI
FIN SI

// 4. SI hubo cambios, actualizar BD
SI changes NO ESTÁ VACÍO ENTONCES
  ACTUALIZAR_DB(fh_pax_summary, fuid, {
    ...dataToUpdate,
    source: cloudEvent.source,
    updated_at: AHORA()
  })

// 5. Publicar evento al Event Publisher CON campo "changes"
PUBLICAR_A_COLA(
  cola: 'event_publisher_queue.fifo',
  mensaje: {
    specversion: "1.0",
    type: "com.iberia.flighthub.passengers.updated",
    source: "FlightHub.PassengersService",
    subject: "PAX UPDATE",
    id: GENERAR_ULID(),
    time: ISO_TIMESTAMP(AHORA()),
    datacontenttype: "application/json",
    data: {
      // Los 6 campos (NO incluir FUID aquí para Event Publisher)
      operationDate: identifiers.operationDate,
      flightDesignator: identifiers.flightDesignator,
      operationalSuffix: identifiers.operationalSuffix,
      airlineDesignator: identifiers.airlineDesignator,
      departureAirport: identifiers.departureAirport,
      departureNumber: identifiers.departureNumber,

      // Datos del dominio
      total_passengers: dataToUpdate.total_passengers,
      checked_in_passengers: dataToUpdate.checked_in_passengers,
      boarded_passengers: currentData.boarded_passengers,

      // Campo "changes" (trigger para Event Publisher)
      changes: changes, // ["total_passengers",
"checked_in_passengers"]

      // Metadata
      source: cloudEvent.source,
      timestamp: ISO_TIMESTAMP(AHORA())
    }
  }
)

```

```
    }  
  }  
)  
FIN SI  
  
FIN FUNCIÓN
```

### Evento publicado a EventBridge (sin FUID):

```
{  
  "flightIdentifier": {  
    "operationDate": "2025-01-14",  
    "flightDesignator": "347",  
    "operationalSuffix": "",  
    "airlineDesignator": "IB",  
    "departureAirport": "MAD",  
    "departureNumber": 1  
  },  
  "eventType": "com.iberia.flighthub.passengers.updated",  
  "timestamp": "2025-01-14T08:30:00Z",  
  "source": "FlightHub.PassengersService",  
  "changes": ["total_passengers", "checked_in_passengers"],  
  "passengers": {  
    "total": 180,  
    "checkedIn": 150,  
    "boarded": 0  
  }  
}
```

**NOTA:** El FUID NO se incluye en el evento externo (EventBridge). Solo se usan los 6 campos de identificación estándar.







### Manejo de Turnarounds (departureNumber)

Cuando un vuelo despegue múltiples veces en el mismo día:

```
// Primer despegue  
{  
  fuid: "01HQZ8X9Y1K2M3N4P5Q6R7S8T9",  
  operationDate: "2025-01-14",  
  flightDesignator: "347",  
  airlineDesignator: "IB",  
  departureAirport: "MAD",  
  departureNumber: 1, // Primer intento  
  previousDepartureNumber: null  
}  
  
// Diversion/Return – segundo despegue
```

```
{  
  fuid: "01HQZ8X9Y1K2M3N4P5Q6R7S8T9", // Mismo FUID  
  operationDate: "2025-01-14",  
  flightDesignator: "347",  
  airlineDesignator: "IB",  
  departureAirport: "MAD",  
  departureNumber: 2, // Segundo intento  
  previousDepartureNumber: 1 // Referencia al anterior  
}
```

### Beneficios del Modelo Dual:

1.  **FUID interno:** Simplicidad, inmutabilidad, performance (solo uso interno)
2.  **6 campos externos:** Compatibilidad con estándares aeronáuticos
3.  **Trazabilidad:** `departureNumber` mantiene relación entre intentos
4.  **Flexibilidad:** Publisher adapta payloads según consumidor
5.  **Separación de concerns:** Dominios trabajan con FUID + 6 campos, externos solo 6 campos
6.  **Sin FUID en EventBridge:** Los sistemas externos no necesitan conocer el FUID interno

---

## Identificación de Vuelos









### FUID (Identificador Único Permanente)

**FUID** es el identificador interno permanente del vuelo en FlightHub.

**Formato:** ULID (26 caracteres)

**Ejemplo:** 01HQZ8X9Y1K2M3N4P5Q6R7S8T9

### Características:

-  Único en todo el sistema
-  Ordenable cronológicamente
-  Compatible con UUIDs
-  Inmutable (no cambia durante vida del vuelo)
-  Creado SOLO por el Flight Orchestrator (usando ASM NEW o SSIM NEW)
-  **Propósito:** Vincular todos los datos del mismo vuelo, incluso cuando cambian los identificadores externos
-  **NO se publica a EventBridge** (solo para uso interno)
-  **NO se usa para FIFO** (las colas SQS FIFO ya garantizan orden)

### 6 Campos de Identificación Externa

Para publicación externa (EventBridge) y matching de vuelos:

1. **operation\_date** - Fecha UTC de operación
2. **flight\_designator** - Número de vuelo (ej: "347")
3. **operational\_suffix** - Sufijo operacional ("A", "B", o "")
4. **airline\_designator** - Código IATA de aerolínea (ej: "IB")

- 5. **departure\_airport** - Código IATA del aeropuerto (ej: "MAD")
- 6. **departure\_number** - Número de intento de despegue (1, 2, 3...)

Comparación con Sistema Actual

Servicio	Formato Actual	FUID Nueva
telex-worker	IB034720250114MAD	01HQZ8X9Y1K2M3N4P5Q6R7S8T9
aena-worker	IB 0347 20250114MAD	01HQZ8X9Y1K2M3N4P5Q6R7S8T9
etlb-worker	IB 0347 20250114MAD	01HQZ8X9Y1K2M3N4P5Q6R7S8T9
cki-worker	UUID variable	01HQZ8X9Y1K2M3N4P5Q6R7S8T9
ssim-worker	IB0347-20250114-MAD	01HQZ8X9Y1K2M3N4P5Q6R7S8T9

**Beneficio:** Un solo identificador (FUID) en toda la nueva arquitectura.

Modelo de Base de Datos por Dominios

Basado en el schema actual (CloudEvents v1), se descomponen las tablas actuales en **13 dominios granulares** con prefijos consistentes.

Dominios de la Nueva Arquitectura

- 1. 🎯 **Flight Orchestrator** - Identificación y enrutamiento
- 2. 📍 **Resources** - Recursos aeroportuarios
- 3. ⌚ **Timeline** - Tiempos operacionales
- 4. ⚠️ **Delays** - Retrasos
- 5. 👨 **Crew** - Tripulación
- 6. 🚨 **Alerts** - Alarmas y desvíos
- 7. 👤 **Passengers** - Pasajeros y cabinas
- 8. 🧳 **Baggage** - Equipaje y carga
- 9. ⛽ **Fuel** - Combustible
- 10. ✈️ **Aircraft** - Aeronaves
- 11. 📅 **Schedules** - Programación
- 12. 🔄 **Onward Flights** - Vuelos de continuación
- 13. 🤝 **Codeshare** - Vuelos compartidos

🎯 DOMINIO: Flight Orchestrator

**Responsabilidad:** Identificación de vuelos, FUID, enrutamiento

Tablas:

- **flights** (identificadores únicamente)
- **message\_log** (auditoría completa)
- **precedence\_rules** (configuración de precedencias)

**Responsabilidad:** Recursos aeroportuarios (gates, stands, runways, belts)

**Prefijo:** fh\_resource\_

### Tabla fh\_resource\_airport

```
CREATE TABLE fh_resource_airport (  
  fuid VARCHAR(26) PRIMARY KEY REFERENCES flights(fuid),  
  
  -- 6 Campos de Identificación  
  operation_date DATE NOT NULL,  
  flight_designator VARCHAR(10) NOT NULL,  
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',  
  airline_designator VARCHAR(3) NOT NULL,  
  departure_airport VARCHAR(3) NOT NULL,  
  departure_number INTEGER NOT NULL DEFAULT 1,  
  
  -- Recursos de Salida  
  departure_terminal_zone VARCHAR NULL,  
  departure_terminal VARCHAR NULL,  
  departure_stand VARCHAR NULL,  
  departure_runway VARCHAR NULL,  
  departure_checkin_counter_first VARCHAR NULL,  
  departure_checkin_counter_last VARCHAR NULL,  
  departure_checkin_counter_type VARCHAR NULL,  
  departure_boarding_zone VARCHAR NULL,  
  departure_boarding_gate VARCHAR NULL,  
  departure_boarding_gate_2 VARCHAR NULL,  
  departure_bag_belt VARCHAR NULL,  
  departure_bag_belt_status VARCHAR NULL,  
  
  -- Recursos de Llegada  
  arrival_terminal_zone VARCHAR NULL,  
  arrival_terminal VARCHAR NULL,  
  arrival_stand VARCHAR NULL,  
  arrival_runway VARCHAR NULL,  
  arrival_hall VARCHAR NULL,  
  arrival_gate VARCHAR NULL,  
  arrival_bag_belt VARCHAR NULL,  
  arrival_bag_belt_2 VARCHAR NULL,  
  arrival_bag_belt_status VARCHAR NULL,  
  arrival_bag_claim_unit_code VARCHAR NULL,  
  arrival_bag_claim_unit_status VARCHAR NULL,  
  
  -- Auditoría  
  created_at TIMESTAMP NOT NULL,  
  created_by VARCHAR NOT NULL,  
  updated_at TIMESTAMP NOT NULL,  
  updated_by VARCHAR NOT NULL,
```

```
INDEX idx_departure_gate (departure_boarding_gate),  
INDEX idx_departure_stand (departure_stand),  
INDEX idx_arrival_gate (arrival_gate),  
INDEX idx_arrival_stand (arrival_stand)  
);
```



## DOMINIO: Timeline

**Responsabilidad:** Todos los tiempos operacionales del vuelo

**Prefijo:** fh\_timeline\_

### Tabla fh\_timeline\_departure

```
CREATE TABLE fh_timeline_departure (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  -- 6 Campos de Identificación  
  operation_date DATE NOT NULL,  
  flight_designator VARCHAR(10) NOT NULL,  
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',  
  airline_designator VARCHAR(3) NOT NULL,  
  departure_airport VARCHAR(3) NOT NULL,  
  departure_number INTEGER NOT NULL DEFAULT 1,  
  
  -- Tiempos programados  
  departure_time_scheduled TIMESTAMP NOT NULL,  
  
  -- Tiempos estimados  
  departure_time_estimated TIMESTAMP NOT NULL,  
  off_blocks_time_estimated TIMESTAMP NOT NULL,  
  taxi_out_time_estimated NUMERIC NULL,  
  takeoff_time_estimated TIMESTAMP NOT NULL,  
  
  -- Tiempos reales  
  departure_time_actual TIMESTAMP NULL,  
  off_blocks_time_actual TIMESTAMP NULL,  
  taxi_out_time_actual NUMERIC NULL,  
  takeoff_time_actual TIMESTAMP NULL,  
  
  -- Auditoría  
  created_at TIMESTAMP NOT NULL,  
  created_by VARCHAR NOT NULL,  
  updated_at TIMESTAMP NOT NULL,  
  updated_by VARCHAR NOT NULL,  
  
  UNIQUE(fuid),  
  INDEX idx_scheduled (departure_time_scheduled),
```



```
INDEX idx_actual (departure_time_actual)
);
```

### Tabla fh\_timeline\_arrival

```
CREATE TABLE fh_timeline_arrival (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
  flight_designator VARCHAR(10) NOT NULL,
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
  airline_designator VARCHAR(3) NOT NULL,
  departure_airport VARCHAR(3) NOT NULL,
  departure_number INTEGER NOT NULL DEFAULT 1,

  -- Tiempos programados
  arrival_time_scheduled TIMESTAMP NOT NULL,

  -- Tiempos estimados
  arrival_time_estimated TIMESTAMP NOT NULL,
  landing_time_estimated TIMESTAMP NOT NULL,
  on_blocks_time_estimated TIMESTAMP NOT NULL,
  taxi_in_time_estimated NUMERIC NULL,

  -- Tiempos reales
  arrival_time_actual TIMESTAMP NULL,
  landing_time_actual TIMESTAMP NULL,
  on_blocks_time_actual TIMESTAMP NULL,
  taxi_in_time_actual NUMERIC NULL,

  -- Auditoría
  created_at TIMESTAMP NOT NULL,
  created_by VARCHAR NOT NULL,
  updated_at TIMESTAMP NOT NULL,
  updated_by VARCHAR NOT NULL,

  UNIQUE(fuid),
  INDEX idx_scheduled (arrival_time_scheduled),
  INDEX idx_actual (arrival_time_actual)
);
```

### Tabla fh\_timeline\_cdm

```
CREATE TABLE fh_timeline_cdm (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),
```

```

-- 6 Campos de Identificación
operation_date DATE NOT NULL,
flight_designator VARCHAR(10) NOT NULL,
operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
airline_designator VARCHAR(3) NOT NULL,
departure_airport VARCHAR(3) NOT NULL,
departure_number INTEGER NOT NULL DEFAULT 1,

-- Off-Blocks Times (CDM)
off_blocks_time_scheduled_cdm TIMESTAMP NULL,
off_blocks_time_estimated_cdm TIMESTAMP NULL,
off_blocks_time_target_cdm TIMESTAMP NULL,
off_blocks_time_actual_cdm TIMESTAMP NULL,

-- Startup Times (CDM)
startup_request_time_actual_cdm TIMESTAMP NULL,
startup_approval_time_target_cdm TIMESTAMP NULL,
startup_approval_time_actual_cdm TIMESTAMP NULL,

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

UNIQUE(fuid),
INDEX idx_tobt (off_blocks_time_target_cdm),
INDEX idx_tsat (startup_approval_time_target_cdm)
);

```

## Tabla fh\_timeline\_checkin

```

CREATE TABLE fh_timeline_checkin (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
  flight_designator VARCHAR(10) NOT NULL,
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
  airline_designator VARCHAR(3) NOT NULL,
  departure_airport VARCHAR(3) NOT NULL,
  departure_number INTEGER NOT NULL DEFAULT 1,

  -- Check-in CI (inicial)
  checkin_ci_time_actual TIMESTAMP NULL,

  -- Check-in CL (intermedio)
  checkin_cl_time_actual TIMESTAMP NULL,

```

```

-- Check-in CC (final)
checkin_cc_time_actual TIMESTAMP NULL,

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

UNIQUE(fuid)
);

```

## ⚠ DOMINIO: Delays

**Responsabilidad:** Códigos de retraso y tiempos de delay

**Prefijo:** fh\_delay\_

**Tabla fh\_delay**

```

CREATE TABLE fh_delay (
  fuid VARCHAR(26) PRIMARY KEY REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
  flight_designator VARCHAR(10) NOT NULL,
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
  airline_designator VARCHAR(3) NOT NULL,
  departure_airport VARCHAR(3) NOT NULL,
  departure_number INTEGER NOT NULL DEFAULT 1,

  -- Retraso 1
  delay_type_1 VARCHAR NOT NULL,
  delay_number_1 VARCHAR NOT NULL,
  delay_code_1 VARCHAR NOT NULL,
  delay_group_code_1 VARCHAR NOT NULL,
  delay_mins_1 NUMERIC NOT NULL,
  delay_comments_1 VARCHAR NOT NULL,

  -- Retraso 2
  delay_type_2 VARCHAR NULL,
  delay_code_2 VARCHAR NULL,
  delay_mins_2 NUMERIC NULL,

  -- Retraso 3
  delay_type_3 VARCHAR NULL,
  delay_code_3 VARCHAR NULL,
  delay_mins_3 NUMERIC NULL,

  -- Retraso 4

```

```

delay_type_4 VARCHAR NULL,
delay_code_4 VARCHAR NULL,
delay_mins_4 NUMERIC NULL,

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

INDEX idx_delay_code_1 (delay_code_1),
INDEX idx_delay_mins (delay_mins_1, delay_mins_2, delay_mins_3,
delay_mins_4)
);

```



DOMINIO: Crew

**Responsabilidad:** Asignación de tripulación técnica y de cabina

**Prefijo:** fh\_crew\_

**Tabla fh\_crew\_assignment**

```

CREATE TABLE fh_crew_assignment (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

-- 6 Campos de Identificación
operation_date DATE NOT NULL,
flight_designator VARCHAR(10) NOT NULL,
operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
airline_designator VARCHAR(3) NOT NULL,
departure_airport VARCHAR(3) NOT NULL,
departure_number INTEGER NOT NULL DEFAULT 1,

-- Información de tripulación técnica (Cockpit)
cockpit_crew_count INTEGER NULL,
cockpit_employer VARCHAR NULL,

-- Información de tripulación de cabina
cabin_crew_count INTEGER NULL,
cabin_employer VARCHAR NULL,

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

```

```
INDEX idx_fuid (fuid)
);
```



## DOMINIO: Alerts

**Responsabilidad:** Alarmas operacionales y desvíos

**Prefijo:** fh\_alert\_

### Tabla fh\_alert\_alarm

```
CREATE TABLE fh_alert_alarm (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
  flight_designator VARCHAR(10) NOT NULL,
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
  airline_designator VARCHAR(3) NOT NULL,
  departure_airport VARCHAR(3) NOT NULL,
  departure_number INTEGER NOT NULL DEFAULT 1,

  -- Información de la alarma
  alarm_code VARCHAR(20) NULL,
  alarm_text TEXT NULL,
  alarm_severity VARCHAR(20) NULL,

  -- Auditoría
  created_at TIMESTAMP NOT NULL,
  created_by VARCHAR NOT NULL,
  updated_at TIMESTAMP NOT NULL,
  updated_by VARCHAR NOT NULL,

  INDEX idx_fuid (fuid),
  INDEX idx_alarm_code (alarm_code),
  INDEX idx_severity (alarm_severity)
);
```

### Tabla fh\_alert\_diversion

```
CREATE TABLE fh_alert_diversion (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
```

```

flight_designator VARCHAR(10) NOT NULL,
operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
airline_designator VARCHAR(3) NOT NULL,
departure_airport VARCHAR(3) NOT NULL,
departure_number INTEGER NOT NULL DEFAULT 1,

-- Información del desvío
diversion_type VARCHAR(20) NULL,
diversion_airport VARCHAR(3) NULL,
diversion_code VARCHAR(20) NULL,

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

INDEX idx_fuid (fuid),
INDEX idx_diversion_airport (diversion_airport)
);

```

## DOMINIO: Passengers

**Responsabilidad:** Pasajeros, check-in, cabinas, contadores

**Prefijo:** fh\_pax\_

### Tabla fh\_pax\_summary

```

CREATE TABLE fh_pax_summary (
  fuid VARCHAR(26) PRIMARY KEY REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
  flight_designator VARCHAR(10) NOT NULL,
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
  airline_designator VARCHAR(3) NOT NULL,
  departure_airport VARCHAR(3) NOT NULL,
  departure_number INTEGER NOT NULL DEFAULT 1,

  -- Totales generales
  config_total NUMERIC NOT NULL DEFAULT 0,
  capacity_total NUMERIC NOT NULL DEFAULT 0,
  availability_total NUMERIC NOT NULL DEFAULT 0,
  load_factor_total NUMERIC NOT NULL DEFAULT 0,
  booked_pax_total NUMERIC NOT NULL DEFAULT 0,
  checked_pax_total NUMERIC NOT NULL DEFAULT 0,
  boarded_pax_total NUMERIC NOT NULL DEFAULT 0,
  forecast_pax_total NUMERIC NOT NULL DEFAULT 0,

```

```

-- Por conexión
inbound_booked_pax_total NUMERIC NOT NULL DEFAULT 0,
outbound_booked_pax_total NUMERIC NOT NULL DEFAULT 0,

-- Por tipo de pasajero
adult_pax_total NUMERIC NOT NULL DEFAULT 0,
male_pax_total NUMERIC NOT NULL DEFAULT 0,
female_pax_total NUMERIC NOT NULL DEFAULT 0,
no_gender_pax_total NUMERIC NOT NULL DEFAULT 0,
child_pax_total NUMERIC NOT NULL DEFAULT 0,
infant_pax_total NUMERIC NOT NULL DEFAULT 0,

-- Pasajeros especiales
pad_pax_total NUMERIC NOT NULL DEFAULT 0,
dhc_pax_total NUMERIC NOT NULL DEFAULT 0,

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

-- Índices
INDEX idx_booked (booked_pax_total),
INDEX idx_checked (checked_pax_total),
INDEX idx_boarded (boarded_pax_total)
);

```

## Tabla fh\_pax\_cabin

```

CREATE TABLE fh_pax_cabin (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
  flight_designator VARCHAR(10) NOT NULL,
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
  airline_designator VARCHAR(3) NOT NULL,
  departure_airport VARCHAR(3) NOT NULL,
  departure_number INTEGER NOT NULL DEFAULT 1,

  -- Identificación de cabina
  cabin_code VARCHAR(10) NOT NULL,      -- JC, W, Y
  cabin_name VARCHAR(50) NULL,          -- Business, Premium, Turista

  -- Configuración y capacidad
  config_cabin NUMERIC NOT NULL DEFAULT 0,
  capacity_cabin NUMERIC NOT NULL DEFAULT 0,
  availability_cabin NUMERIC NOT NULL DEFAULT 0,
  load_factor_cabin NUMERIC NOT NULL DEFAULT 0,

```

```

-- Contadores de pasajeros
booked_pax_cabin NUMERIC NOT NULL DEFAULT 0,
checked_pax_cabin NUMERIC NOT NULL DEFAULT 0,
boarded_pax_cabin NUMERIC NOT NULL DEFAULT 0,
forecast_pax_cabin NUMERIC NOT NULL DEFAULT 0,

-- Por tipo de pasajero
adult_pax_cabin NUMERIC NOT NULL DEFAULT 0,
child_pax_cabin NUMERIC NOT NULL DEFAULT 0,
infant_pax_cabin NUMERIC NOT NULL DEFAULT 0,

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

UNIQUE(fuid, cabin_code),
INDEX idx_cabin_code (cabin_code)
);

```

### Tabla fh\_pax\_special

```

CREATE TABLE fh_pax_special (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
  flight_designator VARCHAR(10) NOT NULL,
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
  airline_designator VARCHAR(3) NOT NULL,
  departure_airport VARCHAR(3) NOT NULL,
  departure_number INTEGER NOT NULL DEFAULT 1,

  -- Categoría del pasajero especial
  category_code VARCHAR(10) NOT NULL,      -- WCHC, WCHR, WCHS, UM, etc.
  category_name VARCHAR(100) NULL,
  quantity INTEGER NOT NULL,

  -- Auditoría
  created_at TIMESTAMP NOT NULL,
  created_by VARCHAR NOT NULL,
  updated_at TIMESTAMP NOT NULL,
  updated_by VARCHAR NOT NULL,

  INDEX idx_fuid (fuid),
  INDEX idx_category (category_code)
);

```



## DOMINIO: Baggage

**Responsabilidad:** Equipaje, ULDs, peso, carga

**Prefijo:** fh\_bag\_

### Tabla fh\_bag\_summary

```
CREATE TABLE fh_bag_summary (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  -- 6 Campos de Identificación  
  operation_date DATE NOT NULL,  
  flight_designator VARCHAR(10) NOT NULL,  
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',  
  airline_designator VARCHAR(3) NOT NULL,  
  departure_airport VARCHAR(3) NOT NULL,  
  departure_number INTEGER NOT NULL DEFAULT 1,  
  
  -- Información de equipaje  
  total_bags INTEGER NULL,  
  total_bag_weight_kg INTEGER NULL,  
  
  -- Auditoría  
  created_at TIMESTAMP NOT NULL,  
  created_by VARCHAR NOT NULL,  
  updated_at TIMESTAMP NOT NULL,  
  updated_by VARCHAR NOT NULL,  
  
  INDEX idx_fuid (fuid)  
);
```

### Tabla fh\_bag\_cargo

```
CREATE TABLE fh_bag_cargo (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  -- 6 Campos de Identificación  
  operation_date DATE NOT NULL,  
  flight_designator VARCHAR(10) NOT NULL,  
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',  
  airline_designator VARCHAR(3) NOT NULL,  
  departure_airport VARCHAR(3) NOT NULL,  
  departure_number INTEGER NOT NULL DEFAULT 1,
```

```

-- Pesos de carga (kg)
cargo_weight_kg INTEGER NULL,
additional_cargo_weight_kg INTEGER NULL,
total_cargo_weight_kg INTEGER NULL,

-- Pesos de correo (kg)
mail_weight_kg INTEGER NULL,
additional_mail_weight_kg INTEGER NULL,
total_mail_weight_kg INTEGER NULL,

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

INDEX idx_fuid (fuid)
);

```

### Tabla **fh\_bag\_cargo\_item**

```

CREATE TABLE fh_bag_cargo_item (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  -- 6 Campos de Identificación
  operation_date DATE NOT NULL,
  flight_designator VARCHAR(10) NOT NULL,
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
  airline_designator VARCHAR(3) NOT NULL,
  departure_airport VARCHAR(3) NOT NULL,
  departure_number INTEGER NOT NULL DEFAULT 1,

  -- Información del ítem
  item_name VARCHAR(100) NOT NULL,    -- AVI, DGR, HUM, etc.
  item_value NUMERIC NULL,
  item_unit VARCHAR(20) NULL,

  -- Auditoría
  created_at TIMESTAMP NOT NULL,
  created_by VARCHAR NOT NULL,
  updated_at TIMESTAMP NOT NULL,
  updated_by VARCHAR NOT NULL,

  INDEX idx_fuid (fuid),
  INDEX idx_item_name (item_name)
);

```

**Responsabilidad:** Combustible, repostaje

**Prefijo:** fh\_fuel\_

### Tabla fh\_fuel\_summary

```
CREATE TABLE fh_fuel_summary (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  -- 6 Campos de Identificación  
  operation_date DATE NOT NULL,  
  flight_designator VARCHAR(10) NOT NULL,  
  operational_suffix VARCHAR(3) NOT NULL DEFAULT '',  
  airline_designator VARCHAR(3) NOT NULL,  
  departure_airport VARCHAR(3) NOT NULL,  
  departure_number INTEGER NOT NULL DEFAULT 1,  
  
  -- Pesos de combustible (kg)  
  total_fuel_weight_kg INTEGER NULL,  
  taxi_fuel_weight_kg INTEGER NULL,  
  trip_fuel_weight_kg INTEGER NULL,  
  
  -- Auditoría  
  created_at TIMESTAMP NOT NULL,  
  created_by VARCHAR NOT NULL,  
  updated_at TIMESTAMP NOT NULL,  
  updated_by VARCHAR NOT NULL,  
  
  INDEX idx_fuid (fuid)  
);
```

### Tabla fh\_fuel\_accept\_aircraft

```
CREATE TABLE fuel_accept_aircraft (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  arrival_fuel_kg INTEGER,  
  remaining_fuel_kg INTEGER,  
  fuel_tipping_kg INTEGER,  
  depart_fuel_kg INTEGER,  
  calculated_planned_uplift_kg INTEGER,  
  required_block_fuel_kg INTEGER,  
  
  source VARCHAR(50),  
  received_at TIMESTAMP,
```

```
updated_at TIMESTAMP DEFAULT NOW(),  
  
    UNIQUE(fuid)  
);
```

### Tabla **fuel\_close\_flight**

```
CREATE TABLE fuel_close_flight (  
    id UUID PRIMARY KEY,  
    fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
    arrival_fuel_kg INTEGER,  
    remaining_fuel_kg INTEGER,  
    fuel_tipping_kg INTEGER,  
    depart_fuel_kg INTEGER,  
    calculated_planned_uplift_kg INTEGER,  
    required_block_fuel_kg INTEGER,  
  
    source VARCHAR(50),  
    received_at TIMESTAMP,  
    updated_at TIMESTAMP DEFAULT NOW(),  
  
    UNIQUE(fuid)  
);
```

### Tabla **fueling\_events**

```
CREATE TABLE fueling_events (  
    id UUID PRIMARY KEY,  
    fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
    event_type VARCHAR(50),          -- "ACCEPT_AIRCRAFT",  
    "CLOSE_FLIGHT"  
  
    measurement_system VARCHAR(20),    -- "metric", "imperial", "us"  
    supplier VARCHAR(100),  
    vendor VARCHAR(100),  
    invoice_number VARCHAR(50),  
  
    fuel_type VARCHAR(20),              -- "JET A1"  
    density NUMERIC(10, 3),  
    density_unit VARCHAR(20),  
  
    actual_uplift NUMERIC(10, 2),  
    actual_uplift_unit VARCHAR(20),  
  
    new_fuel_kg INTEGER,  
    reason TEXT,  
    reduce_note TEXT,
```

```
source VARCHAR(50),
created_at TIMESTAMP DEFAULT NOW(),

INDEX idx_fuid (fuid)
);
```



## DOMINIO: Operations

**Responsabilidad:** Tiempos, gates, stands, delays, tripulación

### Tabla `departure_times`

```
CREATE TABLE departure_times (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  -- Scheduled
  scheduled_departure_time TIMESTAMP,
  scheduled_departure_time_aena TIMESTAMP,

  -- Estimated
  estimated_departure_time TIMESTAMP,
  estimated_off_blocks_time TIMESTAMP,

  -- Target
  target_off_blocks_time TIMESTAMP,
  target_startup_time TIMESTAMP,
  target_takeoff_time TIMESTAMP,

  -- Actual
  actual_off_blocks_time TIMESTAMP,
  actual_off_blocks_time_acars TIMESTAMP,
  actual_takeoff_time TIMESTAMP,
  actual_takeoff_time_acars TIMESTAMP,
  actual_startup_time TIMESTAMP,

  -- Confirmed
  confirmed_estimated_departure_time_code VARCHAR(1), -- "E", "C"
  calculated_takeoff_time TIMESTAMP,
  new_information_departure_time TIMESTAMP,

  -- Request times
  startup_request_time TIMESTAMP,
  aircraft_ready_time TIMESTAMP,

  -- Door times
  door_close_time TIMESTAMP,
  passenger_door_close_time TIMESTAMP,
  cargo_door_close_time TIMESTAMP,
```

```

-- Return flight
flight_return_land_time TIMESTAMP,
flight_return_on_blocks_time TIMESTAMP,

-- Taxi
taxi_minutes INTEGER,

-- Metadata
target_off_blocks_time_status VARCHAR(20),
target_off_blocks_time_changes TEXT,
target_startup_code VARCHAR(10),

source VARCHAR(50),
source_priority INTEGER,
updated_at TIMESTAMP DEFAULT NOW(),

UNIQUE(fuid)
);

```

## Tabla **arrival\_times**

```

CREATE TABLE arrival_times (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

-- Scheduled
scheduled_arrival_time TIMESTAMP,
scheduled_arrival_time_aena TIMESTAMP,

-- Estimated
estimated_arrival_time TIMESTAMP,
estimated_on_blocks_time TIMESTAMP,
estimated_land_time TIMESTAMP,

-- Actual
actual_on_blocks_time TIMESTAMP,
actual_on_blocks_time_acars TIMESTAMP,
actual_land_time TIMESTAMP,
land_time TIMESTAMP,
land_time_acars TIMESTAMP,

-- Milestones
ten_miles_out_time TIMESTAMP,
arrival_runway_time TIMESTAMP,
arrival_stand_time TIMESTAMP,
arrival_hall_time TIMESTAMP,

source VARCHAR(50),
source_priority INTEGER,
updated_at TIMESTAMP DEFAULT NOW(),

```

```
    UNIQUE(fuid)
);
```

### Tabla **gate\_assignments**

```
CREATE TABLE gate_assignments (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  gate_code VARCHAR(10) NOT NULL,
  gate_type VARCHAR(20),           -- "BOARDING", "ARRIVAL"

  terminal VARCHAR(10),
  terminal_programmed VARCHAR(10),
  terminal_zone VARCHAR(10),

  start_time TIMESTAMP,
  end_time TIMESTAMP,

  -- Estado
  status VARCHAR(20),             -- "ASSIGNED", "ACTIVE", "CLOSED"
  is_current BOOLEAN DEFAULT TRUE,

  -- CKI variant
  is_from_cki BOOLEAN DEFAULT FALSE,

  source VARCHAR(50),
  assigned_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW(),

  INDEX idx_fuid (fuid),
  INDEX idx_gate (gate_code),
  INDEX idx_status (status, is_current)
);
```

### Tabla **stand\_assignments**

```
CREATE TABLE stand_assignments (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  stand_code VARCHAR(10) NOT NULL,
  stand_type VARCHAR(20),         -- "DEPARTURE", "ARRIVAL"

  start_time TIMESTAMP,
  end_time TIMESTAMP,
```

```

status VARCHAR(20),
is_current BOOLEAN DEFAULT TRUE,

-- CKI variant
is_from_cki BOOLEAN DEFAULT FALSE,

source VARCHAR(50),
assigned_at TIMESTAMP DEFAULT NOW(),
updated_at TIMESTAMP DEFAULT NOW(),

INDEX idx_fuid (fuid),
INDEX idx_stand (stand_code)
);

```

### Tabla **baggage\_belt\_assignments**

```

CREATE TABLE baggage_belt_assignments (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  belt_code VARCHAR(10) NOT NULL,
  belt_type VARCHAR(20),           -- "DEPARTURE", "ARRIVAL",
"CLAIM"
  belt_number INTEGER,           -- 1, 2 (si hay secondary)

  start_time TIMESTAMP,
  end_time TIMESTAMP,

  status VARCHAR(20),           -- "ASSIGNED", "ACTIVE", "CLOSED"
  is_current BOOLEAN DEFAULT TRUE,

  source VARCHAR(50),
  assigned_at TIMESTAMP DEFAULT NOW(),

  INDEX idx_fuid (fuid)
);

```

### Tabla **delays**

```

CREATE TABLE delays (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  delay_sequence INTEGER,       -- 1, 2, 3
  delay_code VARCHAR(10),      -- IATA delay codes
  delay_minutes INTEGER,

  extra_delay_code VARCHAR(10),

```



```

extra_delay_minutes INTEGER,

supplementary_information TEXT,

source VARCHAR(50),
created_at TIMESTAMP DEFAULT NOW(),

INDEX idx_fuid (fuid),
INDEX idx_delay_code (delay_code)
);

```

## Tabla crew\_assignments

```

CREATE TABLE crew_assignments (
  id UUID PRIMARY KEY,
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),

  event_type VARCHAR(50),          -- "ACCEPT_AIRCRAFT",
  "CLOSE_FLIGHT", "MANUAL"

  -- Captain
  captain_id VARCHAR(50),
  captain_name VARCHAR(100),
  captain_landing_takeoff VARCHAR(20), -- "LANDING", "TAKEOFF", "BOTH"
  commander_a INTEGER,

  -- First Officer
  first_officer_id VARCHAR(50),
  first_officer_landing_takeoff VARCHAR(20),
  second_in_command_a INTEGER,

  -- Cabin crew
  flight_attendant_a INTEGER,
  tcps_a INTEGER,

  -- Links
  technical_crew_link_flight_id VARCHAR(26),
  auxiliar_crew_link_flight_id VARCHAR(26),

  source VARCHAR(50),
  received_at TIMESTAMP,
  updated_at TIMESTAMP DEFAULT NOW(),

  INDEX idx_fuid (fuid),
  INDEX idx_captain (captain_id)
);

```

## Tabla checkin\_counters

```
CREATE TABLE checkin_counters (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  counter_from VARCHAR(10),  
  counter_to VARCHAR(10),  
  counter_type VARCHAR(20),  
  
  start_time TIMESTAMP,  
  end_time TIMESTAMP,  
  
  source VARCHAR(50),  
  assigned_at TIMESTAMP DEFAULT NOW(),  
  
  INDEX idx_fuid (fuid)  
);
```

## Table **alarms**

```
CREATE TABLE alarms (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  alarm_code VARCHAR(20),  
  alarm_text TEXT,  
  
  source VARCHAR(50),  
  created_at TIMESTAMP DEFAULT NOW(),  
  
  INDEX idx_fuid (fuid)  
);
```

## Table **diversions**

```
CREATE TABLE diversions (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  diversion_airport VARCHAR(3),  
  diversion_code VARCHAR(20),  
  
  source VARCHAR(50),  
  created_at TIMESTAMP DEFAULT NOW(),  
  
  INDEX idx_fuid (fuid)  
);
```



## DOMINIO: Schedules

**Responsabilidad:** SSIM, itinerarios, frecuencias

### Tabla **schedule\_records**

```
CREATE TABLE schedule_records (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26), -- NULL para schedules futuros  
  
  airline_designator VARCHAR(3) NOT NULL,  
  flight_designator VARCHAR(10) NOT NULL,  
  operational_suffix VARCHAR(3),  
  
  period_of_operation_from DATE NOT NULL,  
  period_of_operation_to DATE NOT NULL,  
  days_of_operation VARCHAR(7), -- "1234567"  
  
  departure_station VARCHAR(3) NOT NULL,  
  passenger_std TIME NOT NULL,  
  arrival_station VARCHAR(3) NOT NULL,  
  passenger_sta TIME NOT NULL,  
  
  aircraft_type VARCHAR(10),  
  aircraft_configuration_version VARCHAR(10),  
  service_type VARCHAR(1), -- "J"  
  
  source VARCHAR(50),  
  created_at TIMESTAMP DEFAULT NOW(),  
  
  INDEX idx_airline_flight (airline_designator, flight_designator),  
  INDEX idx_period (period_of_operation_from, period_of_operation_to)  
);
```



## DOMINIO: Aircraft

**Responsabilidad:** Aeronaves, configuraciones

### Tabla **aircraft\_assignments**

```
CREATE TABLE aircraft_assignments (  
  id UUID PRIMARY KEY,  
  fuid VARCHAR(26) NOT NULL REFERENCES flights(fuid),  
  
  tail_number VARCHAR(10) NOT NULL,  
  aircraft_type VARCHAR(10) NOT NULL,
```

```

aircraft_owner VARCHAR(50),
aircraft_version VARCHAR(20),

assigned_at TIMESTAMP NOT NULL,
is_current BOOLEAN DEFAULT TRUE,

source VARCHAR(50),
updated_at TIMESTAMP DEFAULT NOW(),

INDEX idx_fuid (fuid),
INDEX idx_tail_number (tail_number)
);

```

## Tabla **aircraft\_registry**

```

CREATE TABLE aircraft_registry (
  id UUID PRIMARY KEY,
  tail_number VARCHAR(10) UNIQUE NOT NULL,

  aircraft_type VARCHAR(10) NOT NULL,
  aircraft_subtype VARCHAR(20),
  manufacturer VARCHAR(50),
  serial_number VARCHAR(50),

  airline_owner VARCHAR(3),
  registration_country VARCHAR(2),

  operational_status VARCHAR(20),
  in_service_date DATE,
  out_of_service_date DATE,

  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);

```



## DOMINIO: Onward Flights

**Responsabilidad:** Relación entre un vuelo de llegada (inbound) y sus vuelos de continuación (onward), ya sea para pasajeros en conexión o simplemente para mapear continuidad operacional.

**Prefijo:** fh\_onward\_

## Tabla **fh\_onward\_flight**

```

CREATE TABLE fh_onward_flight (
  -- Identificador único
  id UUID PRIMARY KEY,

```

```

-- Vuelo de llegada (inbound)
inbound_fuid VARCHAR(26) NOT NULL REFERENCES fh_flight(fuid),

-- 6 Campos de Identificación del vuelo inbound
operation_date DATE NOT NULL,
flight_designator VARCHAR(10) NOT NULL,
operational_suffix VARCHAR(3) NOT NULL DEFAULT '',
airline_designator VARCHAR(3) NOT NULL,
departure_airport VARCHAR(3) NOT NULL,
departure_number INTEGER NOT NULL DEFAULT 1,

-- Vuelo siguiente / de continuación (onward)
onward_airline_designator VARCHAR(3) NOT NULL,
onward_flight_designator VARCHAR(10) NOT NULL,
onward_operation_date DATE NOT NULL,
onward_operation_day VARCHAR(2),

-- Metadatos y clasificaciones
connection_type VARCHAR(20),          -- direct, interline,
codeshare, etc.
min_connection_time_minutes INTEGER,   -- MCT opcional, para
planificación
turnaround_time_minutes INTEGER,      -- opcional si se usa también
para tiempos entre vuelos

-- Auditoría
created_at TIMESTAMP NOT NULL,
created_by VARCHAR NOT NULL,
updated_at TIMESTAMP NOT NULL,
updated_by VARCHAR NOT NULL,

INDEX idx_inbound (inbound_fuid),
INDEX idx_onward (onward_airline_designator, onward_flight_designator,
onward_operation_date)
);

```

## Notas:

- **Nombre:** `fh_onward_flight` comunica claramente que son vuelos de continuación, no operaciones de turnaround
- **Relación:** Cada registro vincula un vuelo entrante (`inbound_fuid`) con un vuelo "siguiente" definido por su código y fecha de operación
- **Extensible:** Se pueden añadir tablas complementarias como `fh_onward_connection_log`, `fh_mct_rules` o `fh_onward_status` para gestión avanzada

---

👉 DOMINIO: Codeshare

**Responsabilidad:** Vuelos compartidos

## Tabla `codeshare_flights`

```
CREATE TABLE codeshare_flights (  
  id UUID PRIMARY KEY,  
  
  principal_fuid VARCHAR(26) REFERENCES flights(fuid),  
  secondary_fuid VARCHAR(26) REFERENCES flights(fuid),  
  
  codeshare_indicator VARCHAR(1),      -- "P", "S"  
  
  principal_airline VARCHAR(3),  
  principal_flight VARCHAR(10),  
  
  secondary_airline VARCHAR(3),  
  secondary_flight VARCHAR(10),  
  
  created_at TIMESTAMP DEFAULT NOW(),  
  
  INDEX idx_principal (principal_fuid),  
  INDEX idx_secondary (secondary_fuid)  
);
```

---

 DOMINIO: Infrastructure (Compartido)

**Responsabilidad:** Aeropuertos, gates, stands (master data)

Estas tablas permanecen en el servicio `airports-worker` como datos maestros compartidos.

---

## Flujo de Datos

### Ejemplo Completo: MVT con Múltiples Datos

```
┌  
┌ FLUJO COMPLETO: Mensaje MVT → Orchestrator → Dominios → EventBridge ──┐  
└  
└
```

```
// 1. TELEX-IN recibe MVT raw  
rawMVT ← "MVT  
IB347/14  
MAD0845 ETD0842 ET00850  
PAX150 FUEL15000"
```

```
// 2. TELEX-PARSER parsea el mensaje  
parsedMVT ← {  
  airline: 'IB',
```

```

    flightNumber: '347',
    date: '2025-01-14',
    departure: 'MAD',
    estimatedDeparture: '2025-01-14T08:45:00Z',
    offBlocksTime: '2025-01-14T08:42:00Z',
    takeoffTime: '2025-01-14T08:50:00Z',
    totalPassengers: 150,
    fuelOnboard: 15000
}

// 3. TELEX-PARSER publica a Orchestrator (sin FUID, sin AnchorKey)
cloudEventFromParser ← {
  specversion: "1.0",
  type: "com.iberia.flighthub.telex.MVT",
  source: "FlightHub.TelexParser",
  subject: "MVT AD",
  id: "msg-12345",
  time: "2025-01-14T08:00:00Z",
  datacontenttype: "application/json",
  data: parsedMVT // ← Solo datos parseados
}

// 4. ORCHESTRATOR extrae identificadores de los datos parseados
identifiers ← extractTelexIdentifiers(cloudEventFromParser.data)
// {
//   operationDate: '2025-01-14',
//   flightDesignator: '347',
//   operationalSuffix: '',
//   airlineDesignator: 'IB',
//   departureAirport: 'MAD',
//   departureNumber: 1
// }

// 5. ORCHESTRATOR busca/crea FUID usando los 6 campos
fuid ← findOrCreateFUID(identifiers, cloudEventFromParser)
// fuid = "01HQZ8X9Y1K2M3N4P5Q6R7S8T9"

// 7. ORCHESTRATOR extrae datos por dominio y publica 3 eventos
// Los workers reciben: FUID + 6 campos + datos de dominio (NO AnchorKey)

// → Timeline Domain
PUBLICAR_A_COLA(
  cola: 'timeline_events.fifo',
  mensaje: {
    specversion: "1.0",
    type: "com.iberia.flighthub.timeline.updated",
    source: "FlightHub.Orchestrator",
    subject: "TIMELINE MVT",
    id: GENERAR_ULID(),
    time: ISO_TIMESTAMP(AHORA()),
    datacontenttype: "application/json",
    data: {
      fuid: fuid,
      // ← Solo FUID (NO AnchorKey)

```

```

        operationDate: '2025-01-14',          // ← 6 campos
        flightDesignator: '347',
        operationalSuffix: '',
        airlineDesignator: 'IB',
        departureAirport: 'MAD',
        departureNumber: 1,
        estimatedDeparture: '2025-01-14T08:45:00Z', // ← Datos de dominio
        offBlocksTime: '2025-01-14T08:42:00Z',
        takeoffTime: '2025-01-14T08:50:00Z',
        source: 'TELEX_MVT'
    }
}
)

```

// → Passengers Domain

```

PUBLICAR_A_COLA(
    cola: 'passengers_events.fifo',
    mensaje: {
        specversion: "1.0",
        type: "com.iberia.flighthub.passengers.updated",
        source: "FlightHub.Orchestrator",
        subject: "PAX MVT",
        id: GENERAR_ULID(),
        time: ISO_TIMESTAMP(AHORA()),
        datacontenttype: "application/json",
        data: {
            fuid: fuid,          // ← Solo FUID
            operationDate: '2025-01-14', // ← 6 campos
            flightDesignator: '347',
            operationalSuffix: '',
            airlineDesignator: 'IB',
            departureAirport: 'MAD',
            departureNumber: 1,
            totalPassengers: 150,      // ← Datos de dominio
            source: 'TELEX_MVT'
        }
    }
)

```

// → Fuel Domain

```

PUBLICAR_A_COLA(
    cola: 'fuel_events.fifo',
    mensaje: {
        specversion: "1.0",
        type: "com.iberia.flighthub.fuel.updated",
        source: "FlightHub.Orchestrator",
        subject: "FUEL MVT",
        id: GENERAR_ULID(),
        time: ISO_TIMESTAMP(AHORA()),
        datacontenttype: "application/json",
        data: {
            fuid: fuid,          // ← Solo FUID
            operationDate: '2025-01-14', // ← 6 campos

```



```

        flightDesignator: '347',
        operationalSuffix: '',
        airlineDesignator: 'IB',
        departureAirport: 'MAD',
        departureNumber: 1,
        fuelOnboard: 15000,           // ← Datos de dominio
        source: 'TELEX_MVT'
    }
}
)

```

// 6. Cada DOMAIN SERVICE actualiza su BD con FUID + 6 campos

```

// Timeline Service
ACTUALIZAR_DB(fh_timeline_departure, fuid, {
    fuid: fuid,
    operation_date: '2025-01-14',
    flight_designator: '347',
    operational_suffix: '',
    airline_designator: 'IB',
    departure_airport: 'MAD',
    departure_number: 1,
    departure_time_estimated: '2025-01-14T08:45:00Z',
    off_blocks_time_actual: '2025-01-14T08:42:00Z',
    takeoff_time_actual: '2025-01-14T08:50:00Z',
    created_by: 'TELEX_MVT',
    updated_at: AHORA()
})

```

```

// Passengers Service
ACTUALIZAR_DB(fh_pax_summary, fuid, {
    fuid: fuid,
    operation_date: '2025-01-14',
    flight_designator: '347',
    operational_suffix: '',
    airline_designator: 'IB',
    departure_airport: 'MAD',
    departure_number: 1,
    booked_pax_total: 150,
    created_by: 'TELEX_MVT',
    updated_at: AHORA()
})

```

```

// Fuel Service
ACTUALIZAR_DB(fh_fuel_summary, fuid, {
    fuid: fuid,
    operation_date: '2025-01-14',
    flight_designator: '347',
    operational_suffix: '',
    airline_designator: 'IB',
    departure_airport: 'MAD',
    departure_number: 1,
    total_fuel_weight_kg: 15000,

```

```
    created_by: 'TELEX_MVT',
    updated_at: AHORA()
  })

// 7. Cada servicio publica a EVENT PUBLISHER (con "changes")
// Ver ejemplo completo en sección "Event Publisher" arriba
```

## Precedencias

### Sistema de Precedencias por Campo

Cada campo tiene sus propias reglas de precedencia según la fuente.

```
CREATE TABLE precedence_rules (
  id UUID PRIMARY KEY,
  field_name VARCHAR(100) NOT NULL,           -- "estimatedDeparture",
  "fuelOnboard" domain VARCHAR(50) NOT NULL,  -- "operations", "fuel"
  source_priorities JSONB NOT NULL,           -- {"AENA_CDM": 10, "TELEX_MVT":
  8, "MANUAL": 5}

  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW(),

  UNIQUE(field_name, domain)
);
```

### Ejemplo de datos:

```
{
  "field_name": "estimatedDepartureTime",
  "domain": "operations",
  "source_priorities": {
    "AENA_CDM": 10,
    "TELEX_MVT": 8,
    "MANUAL": 5,
    "SSIM": 3
  }
}

{
  "field_name": "fuelOnboard",
  "domain": "fuel",
  "source_priorities": {
    "ETLB": 10,
    "NIMBUS": 9,
    "TELEX_MVT": 7,
```

```

    "MANUAL": 5
  }
}

{
  "field_name": "totalPassengers",
  "domain": "passengers",
  "source_priorities": {
    "CKI": 10,
    "TELEX_MVT": 8,
    "TELEX_LDM": 9,
    "MANUAL": 5
  }
}

```

## Lógica de Precedencias

PRECEDENCIAS: Verificar si se puede actualizar un campo

FUNCIÓN canUpdateField(fuid, fieldName, newSource, newValue)

```

// 1. Obtener valor actual del campo
current ← LEER_VALOR_ACTUAL(fuid, fieldName)

```

```

SI current NO EXISTE ENTONCES
  RETORNAR VERDADERO // No hay valor previo, siempre actualizar
FIN SI

```

```

// 2. Obtener regla de precedencia desde fh_orchestrator.precedence_rules
rule ← CONSULTAR_DB(precedence_rules, DONDE {
  field_name = fieldName
})

```

```

// Ejemplo de rule.source_priorities:
// {
//   "AENA_CDM": 10,
//   "TELEX_MVT": 8,
//   "MANUAL": 5,
//   "SSIM": 3
// }

```

```

currentPriority ← rule.source_priorities[current.source] 0 0
newPriority ← rule.source_priorities[newSource] 0 0

```

```

// 3. Comparar prioridades
SI newPriority > currentPriority ENTONCES

```

```
    RETORNAR VERDADERO // Nueva fuente tiene más prioridad
FIN SI

SI newPriority == currentPriority ENTONCES
    // Misma prioridad: usar timestamp (más reciente gana)
    RETORNAR AHORA() > current.updated_at
FIN SI

RETORNAR FALSO // Nueva fuente tiene menos prioridad, no actualizar

FIN FUNCIÓN
```

---

## Stack Tecnológico

### Messaging

- **AWS SQS FIFO**: Colas FIFO para orden garantizado automáticamente
- **CloudEvents 1.0**: Formato estándar de eventos
- **MessageDeduplicationId**: Prevención de duplicados

**Nota:** El orden FIFO está garantizado por las colas SQS FIFO. No se usa **messageGroupId** con FUID para agrupación.

### Bases de Datos

- **PostgreSQL**: Dominios transaccionales (Operations, Passengers, Fuel, etc.)
- **DynamoDB**: Datos de alta frecuencia si aplica
- **Redis**: Cache compartido para datos maestros

### Eventos

- **CloudEvents**: Formato estándar de eventos internos
- **AWS EventBridge**: Bus de eventos para publicación externa
- **Event Catalog**: Catálogo central de eventos

### Identificadores

- **ULID**: Para FUID (ordenable, único, compatible con UUID)














### Observabilidad

- **OpenTelemetry**: Traces distribuidos
- **CloudWatch**: Logs y métricas
- **Grafana**: Dashboards














---

## Resumen de Dominios y Tablas

### Tabla Resumen Completa

Dominio	Prefijo	# Tablas	Responsabilidad Principal
 <b>Flight Orchestrator</b>	-	3	Identificación única, FUID, enrutamiento, mensajes
 <b>Resources</b>	fh_resource_	1	Recursos aeroportuarios (gates, stands, runways, belts)
 <b>Timeline</b>	fh_timeline_	4	Tiempos operacionales (departure, arrival, CDM, checkin)
 <b>Delays</b>	fh_delay_	1-2	Retrasos y códigos de delay
 <b>Crew</b>	fh_crew_	1	Tripulación técnica y de cabina
 <b>Alerts</b>	fh_alert_	2	Alarmas operacionales y desvíos
 <b>Passengers</b>	fh_pax_	3	Pasajeros, capacidad, cabinas, check-in
 <b>Baggage</b>	fh_bag_	3	Equipaje, carga, correo
 <b>Fuel</b>	fh_fuel_	4	Combustible y repostaje
 <b>Aircraft</b>	fh_aircraft_	2	Aeronaves y configuraciones
 <b>Schedules</b>	fh_schedule_	1	Programación SSIM y servicio
 <b>Onward Flights</b>	fh_onward_	1	Vuelos de continuación
 <b>Codeshare</b>	fh_codeshare_	1	Vuelos compartidos

Conteo Total de Tablas

 Flight Orchestrator:	3 tablas
 Resources:	1 tabla
 Timeline:	4 tablas
 Delays:	1 tabla (+ 1 opcional normalizada)
 Crew:	1 tabla
 Alerts:	2 tablas
 Passengers:	3 tablas
 Baggage:	3 tablas
 Fuel:	4 tablas
 Aircraft:	2 tablas
 Schedules:	1 tabla
 Onward Flights:	1 tabla
 Codeshare:	1 tabla
<hr/>	
TOTAL:	27 tablas

Comparación: Antes vs Después

ANTES (Sistema Legacy):

- ❌ 20+ tablas `flight_*` mezclando responsabilidades
- ❌ `flight_departure_info` con 100+ campos mezclados (4 dominios)
- ❌ Queries complejas con múltiples JOINS
- ❌ Difícil de escalar y mantener
- ❌ 7 formatos diferentes de Flight ID

### DESPUÉS (Nueva Arquitectura):

- ✅ 27 tablas especializadas en 13 dominios granulares
  - ✅ Cada dominio tiene 1-4 tablas enfocadas
  - ✅ Sin JOINS: 6 campos replicados en cada tabla
  - ✅ Escalado independiente por dominio
  - ✅ FUID único en todo el sistema
  - ✅ Prefijos consistentes (`fh_` + dominio)
- 

## Beneficios de la Nueva Arquitectura

1. ✅ **Escalabilidad independiente:** Cada dominio escala según su carga
2. ✅ **Deploys independientes:** Sin riesgo de romper otros dominios
3. ✅ **Ownership claro:** Cada equipo es dueño de su dominio
4. ✅ **Identificación única:** FUID en todo el sistema
5. ✅ **Matching flexible:** Soporta todos los formatos actuales
6. ✅ **Precedencias configurables:** Sin código hardcoded
7. ✅ **Auditoría completa:** Todos los mensajes en `message_log`
8. ✅ **Granularidad:** 32 tablas especializadas vs mezcla actual
9. ✅ **Resiliencia:** Fallos aislados por dominio
10. ✅ **Evolución:** Fácil agregar nuevos dominios