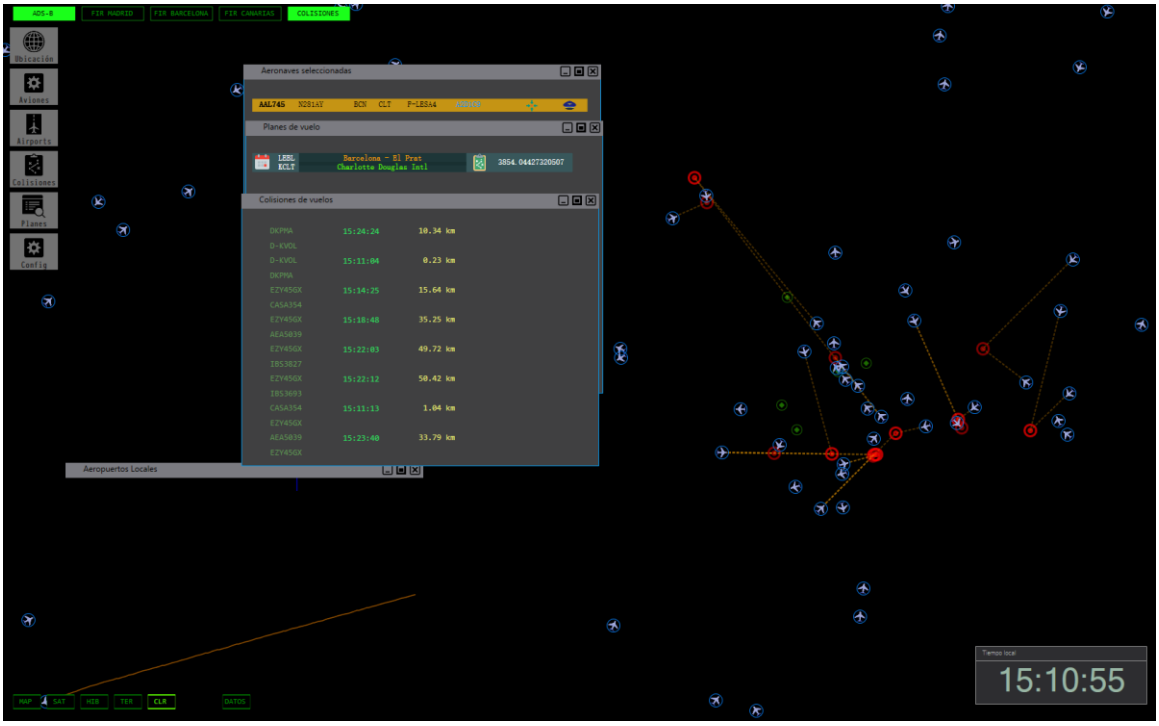
SICOAV II


ADS-B
FIR MADRID
FIR BARCELONA
FIR CANARIAS
COLISIONES
Ubicación
Aviones
Airports
Colisiones
Planes
Config
Aeronaves seleccionadas
AAL745 N2814Y BCN CLT F-LESA4
Planes de vuelo
LEBL
KCLT
Barcelona - El Prat
Charlotte Douglas Intl
3854 04427320507
Colisiones de vuelos
DKPMA 15:24:24 10.34 km
D-KVOL
D-KVOL 15:11:04 0.23 km
DKPMA
EZY45GX 15:14:25 15.64 km
CASA354
EZY45GX 15:18:48 35.25 km
AEA5039
EZY45GX 15:22:03 49.72 km
IBS3827
EZY45GX 15:22:12 50.42 km
IBS3693
CASA354 15:11:13 1.04 km
EZY45GX
AEA5039 15:23:40 33.79 km
EZY45GX
Aeropuertos Locales
MAP
SAT
HIB
TER
CLR
DATOS
Tiempo local
15:10:55

1 Preafio.

1.1 Propósito del documento.

La posición de los aviones es accesible desde algunos servidores que dan servicio de posicionamiento con el sistema de vigilancia dependiente autónoma (ADS-B) como una cooperación en la que el avión determina su posición por GPS y la comunica periódicamente para realizar tareas de seguimiento.

Este sistema será obligatorio en 2020 para algunos aviones, en el caso de Europa y seguramente que para otros espacios aéreos será de igualmente obligatorio.

La información de la posición de los aviones está disponible en la red. Existen una serie de servidores que dan esta información y otras que son importantes para poder hacer el seguimiento de una aeronave. Algunos de estos servidores son gratuitos y otros es necesario pagar o tienen limitado el número de consultas.

Para el desarrollo de la aplicación usaremos un servidor para conocer la posición de los aviones, el histórico y los planes de vuelo.

Lo que se pretende realizar una aplicación en **.NET** con la que mostrar la posición de los aviones que disponen del sistema **ADS-B** que servirá como plataforma de investigación y desarrollo de algoritmo que sean de interés para la gestión del tráfico aéreo.

1.2 Uso del Documento.

Para conocer los distintos servidores de información relativo a planes de vuelo y posicionamiento de aviones encontraremos algunos ejemplos con los que poder realizar este seguimiento.

El documento es útil para conocer las fuentes de las tramas **ADS-B** y con ellas poder hacer las operaciones necesarias de presentación.

Al basar este documento en las informaciones que nos aportan los servidores ADS-B podemos empezar a trabajar para el desarrollo de los elementos software para la presentación en un **GIS** de la posición de los aviones; así, para la creación e algoritmos que sean útiles en la creación de aplicaciones para gestión de tráfico aéreo.

1.3 Visión del documento.

El documento hace una presentación inicial de los servicios; a los que llamaremos **APIs WEB**, en los que nos basaremos para conocer la posición de los aviones. Estos sistemas no solo muestran la posición, en algunos casos, nos dan información histórica de un avión y en otros nos dan información de aeropuertos, líneas aéreas, regiones y meteorológicos.

En la búsqueda rápida de **APIs WEB** tenemos una colección con las que realizar las operaciones necesarias para saber la posición y en segunda medida la ruta o histórico otras nos pueden aportar otras

1.4 Bases del documento.

Basaremos este desarrollo en los sistemas que nos dicen dónde están los aviones; esto pretende ser una guía de los posibles modelos de datos y la creación de servicios con los que realizar todas aquellas funcionalidades, modelos, vistas que nos permitan crear las tareas software necesarias.

1.5 Referencias de la arquitectura.

Como vamos a usar .NET y queremos que sea una aplicación con la que poder hacer un mantenimiento más o menos sostenible. Es importante hacer una arquitectura basada en el concepto de comandos, así como los de **MVC**. De esta manera podemos hacer una separación de los datos, los servicios y las vistas. Con las que podamos hacer desarrollos a futuro que no tengan un impacto inmediato con los demás elementos de la aplicación.

2 Introducción.

2.1 Propósito.

Crear una aplicación para el seguimiento de aviones dentro de un espacio aéreo con la que realizar investigaciones de algoritmos que sean de interés para las aplicaciones que necesite gestionar tráfico aéreo.

Hacer un estudio de las APIs WEB con las que vamos a trabajar para obtener la información que necesitemos; de esta manera, la aplicación será lo suficientemente buena como para que nos dé una buena visión de futuro con las que podamos crear productos asociados a las investigaciones que desarrollaremos.

2.2 Alcance.

En un primer desarrollo solo crearemos una aplicación con la que poder ver los aviones, los históricos, la posición de los aeropuertos y los planes de vuelo disponibles para cada una de las eronaves.

Esta aplicación pretende ser una ayuda para la visualización de los vuelos comerciales aportados por las API gratitas que hay en la RED. No es una representación real puesto

que solo tenemos las tramas ADS-B que son las que los servidores o APIs WEB dan como la posición GPS de los aviones aquellos aviones que no den su posición GPS o no estén dentro de la cobertura del ADS-B no aparecerán en el sistema.

La aplicación usa un GIS en el que presentar los distintos elementos en el espacio pos sus coordenadas. Existen una gran variedad de elementos fijos que nos ayudaran a tener una visión más clara de la posición en la que nos encontramos.

El mapa que nos proporcionan los servidores cartográficos pueden interferir en la visión de los aviones en el mapa. Es por ello que se debería poder filtrar esa imagen degradándola para que la marca de los aviones no quede difuminada por el mapa.

La aplicación deberá almacenar los datos de algunas de las consultas para evitar repetir consultas puesto que está limitado el número de peticiones en especial al servicio de planes de vuelo. Para este caso, se creará una base de datos donde guardaremos los planes de vuelo que hasta ahora hemos encontrado.

2.2.1 Carga inicial del sistema.

La aplicación está basada en el acceso a las APIs WEB de distintos servidores; es por ello, que necesitamos hacer una primera comprobación de la conectividad con estos servidores.

2.2.1.1 Archivo de configuración.

El archivo de configuración es un elemento fundamental para delimitar el ámbito de actuación de las operaciones de seguimiento. En este archivo se encuentran datos relativos a la posición inicial del GIS.

Cuando el sistema arranque tendrá que localizar el archivo de configuración en el que están almacenados los datos necesarios para comenzar a trabajar con la aplicación. En un primer momento tendrá datos relativos:

POSICION INICIAL: Es la posición de control que está definida por las coordenadas iniciales de la presentación del GIS.

2.2.1.2 Aeropuertos.

Cuando cargamos la aplicación tendremos que localizar los aeropuertos que vamos a gestionar en la aplicación. Estas listas de aeropuertos se añadirán en el momento de hacer la instalación del sistema. Es decir, que podemos tener una serie de aeropuertos iniciales o permitir al usuario añadir otros.

Cuando lancemos la aplicación por primera vez, tendremos que recorrer la lista de aeropuertos y hacer llamadas masivas a la API GET AIRPORT (The Flight Plan Database API , s.f.)

Con cada carga actualizaremos la base de datos de aeropuertos que estará formada por un archivo **XML** con el siguiente formato DB_ICAO_DDMMYYY.XML

2.3 Definiciones acrónimos.

2.4 Visión de conjunto.

Se pretende crear una aplicación desarrollada en .NET como entorno de desarrollo y pruebas con las que acelerar la etapa de desarrollo. La aplicación está pensada para que monitorice un espacio aéreo determinado pudiendo realizar el seguimiento de todos los aviones que compartan tramas ADS-B. En esta primera fase estamos monitorizando las posiciones que están dentro de la península ibérica.

La presentación de los aviones de nuestro espacio aéreo es proporcionada por un servicio gratuito y sin limitaciones de peticiones. Cada uno de estos aviones tiene un identificador único con el que poder solicitar información detallada del mismo a los demás servicios; de esta manera, podeos conocer el histórico de un avión con su identificador; podemos saber, el plan de vuelo vinculado al avión.

Lo que conseguiremos es visualizar en tiempo real las posiciones de los aviones para realizar una simple monitorización del avión. Tenemos la posibilidad de visualizar el histórico y su plan de vuelo.

2.4.1 Historico de tracks.

La idea es almacenar el histórico de los aviones que desaparecen de la presentación; es decir, los aviones que salen de espacio y los aviones que llegan a destino dentro de nuestro espacio aéreo. Es importante, realizar esta operación para evitar el consumo de memoria cargado al sistema con aviones que están fuera.

3 Visión del sistema

El sistema realizar una petición de posicionamiento cada cinco segundos. Por cada respuesta creamos los elementos gráficos para presentar el avión en el GIS junto con los datos de altitud, rumbo, nombre y hora de la última actualización.

Cuando seleccionamos un vuelo se actualiza la lista de vuelos que aparecen en la parte superior izquierda de la ventana y desde la que podemos hacer el seguimiento del mismo y nos muestra el histórico del vuelo.

El sistema realiza accesos a las funciones APIs WEB linkladas a cada uno de los servicios. Con esta información realizaremos el llenado de los contenedores y con la información de estos actualizaremos las vistas asociadas al modelo junto con las reglas de negocio de cada una de las vistas.

El sistema necesita comprobar que se tiene accesos a las APIS WEB con las que hacer las consultas.

Para conocer la posición de los aviones se necesita darle a la función APIs WEB, asociada, las coordenadas del sector; es decir máximo superior izquierdo y mínimo inferior derecho con estos parámetros realizaremos consultas cada cinco segundos y nos darán una lista de aviones que están dentro de nuestro sector. Este es un proceso de carga incremental necesario para del sistema como base de su funcionamiento.

El sistema necesita saber los elementos fijos con los que crear las distintas etiquetas GIS con las que presentar datos de cartografía relacionados con el tráfico aéreo. Estos elementos suelen ser en la mayoría de los casos iguales, pero puede que cambien sin un aviso. Es el caso de los aeropuertos que en algún momento puede que cambie su número de pistas o la orientación de las mismas. Este no será un proceso de carga continuo por este motivo se creará una base de datos en las que actualizar los datos de fijos y evitar la sobre carga del sistema sin la necesidad de realizar masivas actualizaciones; evitando, de esta manera la superar algún limite de llamadas a las API_WEB.

3.1 Características el sistema

El sistema irá refrescando la posición de los aviones que irán evolucionando dentro del espacio aéreo. Este proceso actualiza la lista de aviones y crea los elementos gráficos que se irán mostrando en el GIS.

La presentación de las posiciones de los aviones se realiza de forma continua con las llamadas que se hacen al servicio de petición de posiciones en un sector determinado. Esta es la parte de la aplicación con la que conseguimos que nuestra aplicación muestre las posiciones de forma que tenemos una presentación en tiempo real.

3.1.1 Algoritmos.

3.1.1.1 Encoded Polyline Algorithm Format.

En el servicio GET Plan Search de la API api.flightplandatabase.com para la búsqueda de los planes de vuelo. Hay una serie de parámetros de búsqueda disponibles que se combinarán para formar una solicitud de búsqueda.

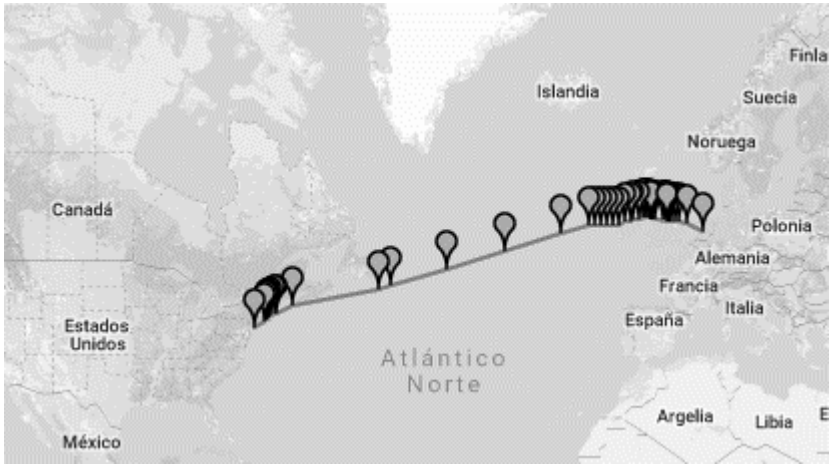
Ejemplo.
https://api.flightplandatabase.com/search/plans?fromICAO=EHAM&toName=Kennedy&limit=1

Salida:

```
[
  {
    "id": 59970,
    "fromICAO": "EHAM",
    "toICAO": "KJFK",
    "fromName": "Schiphol",
    "toName": "John F Kennedy Intl",
    "flightNumber": null,
    "distance": 3417.0572325371704,
    "maxAltitude": 38000,
    "waypoints": 37,
    "likes": 0,
    "downloads": 0,
    "popularity": 0,
    "notes": "",
    "encodedPolyline": "yvh~Hgi`\\qxhHjh~Ci_zAnuw@mtSt_Lcgn@hd]wn{@ldf@kfsC
    "createdAt": "2015-07-06T08:06:41.000Z",
    "updatedAt": "2015-07-06T08:06:41.000Z",
    "tags": [
      "generated"
    ],
    "user": null
  }
]
```


Uno de los campos es el encodedPolyline que es una cadena de texto que utiliza la tabla de código ASCII con los que generar una serie de puntos cartográficos. En el ejemplo de arriba aparece la cadena:

"encodedPolyline":
"slg~Haoa\sipC`khOoha@ptdH(wJxelBotLrieCkcDvts@gaAdpTshBrya@we@lzKorCzdq@g(Cfyt@gcE~heAs~UlfbHchAl_v@ce@jv\sq@hth@o_@trZwrAbsjAs(@rb(@o)@z(cAvgb@ls|CwdR`wrEfqf@bnmAziQxi_Fzxy@pumE?~hbE?~hbE?~hbE?~hbE?~hbE~hbE~po)~reK~b`|@~reK~b`|@~reK~b`|@~s`B~reKv|xJ~tpzA~~mDzjmQzzKbqg@r~n@~gsCfgp@n()@fe)fde@bzGfcJbcKnfNjv(j|b@bhZj(`@b_rDrozI"



1 Interactive Polyline Encoder Utility

Los valores de ENCODE se suman con 63 (el carácter ASCII '?') Antes de convertirlos en codificación de caracteres ASCII para el esquema de codificación base64 familiar: El algoritmo también verificó el grupo de byte significativo anterior, si este bit se establece en 1, el punto aún no está completamente formado y deben seguir datos adicionales.

Todos los puntos codificados en Base64 como enteros con signo, como latitudes y distancias, son valores con signo. El formato de codificación dentro de una polilínea necesita Dada una longitud máxima de +/- 180 grados con una precisión de 5 decimales (180.00000 a -180.00000), esto da como resultado la necesidad de un valor entero binario con signo de 32 bits.

3.1.2 Funciones APIs WEB

3.1.2.1 Posición de los aviones.

A continuación, se describe la API que nos dice los aviones que están volando por un sector definido como un rectángulo de 2 puntos cartográficos.

Llamada.....: http://data-live.flightradar24.com/zones/fcgi/feed.js?

Parametros. : bounds ; adsb; air; array.

Ejemplo: http://data-live.flightradar24.com/zones/fcgi/feed.js?bounds=34.300317,30.003735,128.861654,135.054609&adsb=1&air=1&array=1

Ejemplo de trama de salida:

"1e0f5283","8990C7",31.0418,129.5965,232,41000,407,"3201","T-RJFK1","A333","B-16340",1538400850,"NRT","TPE","BR195",0,0,"EVA195",0

CAMPO_1 = ID_AIRCRAFT -> 1e0f5283.

CAMPO_2 = SSR MODE S - ICAO NUMBER AIRCRAFT -> 8990C7

CAMPO_3 = Latitud -> 31.0418.

CAMPO_4 = Longitud -> 129.5965.

CAMPO_5 = Rumbo -> 232.

CAMPO_6 = Altitud -> 41000

CAMPO_7 = Velocidad -> 407

CAMPO_8 = Transpondedor

CAMPO_9 = Recepción de radar -> T-RJFK1

CAMPO10 = Tipo de avión -> A333.

CAMPO11 = Numero de vuelo -> B-16340

CAMPO12 = Tiempo.

CAMPO13 = Aeropuerto de salida -> NRT. (IATA)

CAMPO14 = Aeropuerto de llegada-> TPE. (IATA)

CAMPO15 = Numero de vuelo IATA -> BR195.

CAMPO16 = ¿

CAMPO17 = ¿

CAMPO18 = Indicativo de llamada (código ICAO) -> EVA195

3.1.3 Elementos gráficos.

3.1.3.1 Avión.

El símbolo de tráfico "básico" se usa para representar el tráfico aéreo. Los símbolos de tráfico pueden ser modificados desde el símbolo básico para proporcionar información de estado especial, como sobre el terreno, seleccionado, designado y alertado.

Los símbolos representados son ejemplos. El ancho de línea, el tamaño físico y el tono de las figuras no son requisitos. Los requisitos se establecen en el texto asociado.

Direccional Básico (ver Figura B-1).

(1) Si la direccionalidad es válida, se debe representar el símbolo de tráfico direccional básico con una forma de punta de flecha orientada por la direccionalidad.

(2) El color debe ser cian o blanco.

(3) El color debe ser del mismo color que el símbolo no direccional básico.

(4) El color no debe ser del mismo color que el símbolo de la propia nave.

(5) Para pantallas que no integran el sistema de aplicaciones de vigilancia de aeronaves (ASA) con TCAS, el símbolo puede estar lleno o no.

(6) Para los sistemas integrados de TCAS / ASA y las implementaciones de ATAS, el símbolo debe estar vacío.

2 Fig B-1

Básico no direccional (ver Figura B-2).

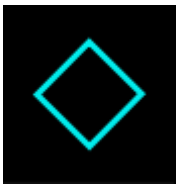
(1) Si la direccionalidad no es válida, el símbolo de tráfico no direccional básico debe ser representado con una forma de diamante.

(2) El color debe ser cian o blanco.

(3) El color debe ser del mismo color que el símbolo direccional básico.

(4) El color no debe ser del mismo color que el símbolo de la propia nave.

(5) Para pantallas que no integran ASA con TCAS, el símbolo puede estar lleno o sin llenar



3 Figura B-2. Símbolo básico no direccional

3.2 Arquitectura del sistema.

3.2.1 Flightplandatabase.

La API de la base de datos del plan de vuelo se proporciona para que los usuarios y desarrolladores creen y vinculen aplicaciones a nuestra ruta y datos de navegación. Esta página documenta cada uno de los puntos finales disponibles, los parámetros esperados y la estructura de las respuestas. La mayoría de los puntos finales de la API están a disposición del público sin autenticación. Sin embargo, algunos puntos finales requieren una clave API, que se puede obtener desde la página de configuración de su cuenta.

3.2.1.1 GET Airport.

Obtiene información sobre un aeropuerto.
GET https://api.flightplandatabase.com/nav/airport/{icao}

ICAO tipo cadena que es un identificador icao del aeropuerto; por ejemplo, para el aeropuerto de Asturias LEAS.

Uso: Con la información que nos da este método podemos realizar una representación gráfica del aeropuerto o una ventana de información. Para el caso se la ventana grafica pasaríamos a la creación de una ventana GIS centrada en las coordenadas y con un recuadro con el tamaño del aeropuerto donde poder representar los que nos dice la API. Esta consulta se almacenará para no tener que repetir la petición a la API.

En ocasiones es posible hacer una carga inicial del sistema haciendo una llamada masiva a la API para cargar el sistema con los datos de los aeropuertos que hemos añadido en una lista. Esto es más adecuado que estar continuamente haciendo representaciones de los datos. Para ello hacemos una carpeta de aeropuertos con un XML que identifique a cada uno de los aeropuertos que tenemos.

3.2.1.2 GET Plan Search

Obtiene información de un plan de vuelo conocidos; para un ejemplo sencillo, el aeropuerto de origen del vuelo y el aeropuerto de llegada.

https://api.flightplandatabase.com/search/plans

Ejemplo:

https://api.flightplandatabase.com/search/plans?fromICAO=EHAM&toName=Kennedy&limit=1

Parameter	Location	Type	Required	Default	Description
q	Querystring	String	No	-	Simple search query. Search departure ICAO & name, destination ICAO & name, username, tags and the flight number
from	Querystring	String	No	-	From search query. Search departure ICAO & name
to	Querystring	String	No	-	To search query. Search departure ICAO & name
fromICAO	Querystring	String	No	-	Matches departure airport ICAO
toICAO	Querystring	String	No	-	Matches destination airport ICAO
fromName	Querystring	String	No	-	Matches departure airport name
toName	Querystring	String	No	-	Matches destination airport name
flightNumber	Querystring	String	No	-	Matches flight number
distanceMin	Querystring	Number (distance)	No	-	Minimum route distance, with units determined by the `X-Units` header
distanceMax	Querystring	Number (distance)	No	-	Maximum route distance, with units determined by the `X-Units` header
tags	Querystring	String	No	-	Tag names to search, comma separated
includeRoute	Querystring	Boolean	No	false	Include route objects for each plan in the response. Setting to true requires the request be authenticated with an API key
page	Querystring	String	No	-	The page of results to fetch
limit	Querystring	Number	No	20	The number of plans to return per page (max 100)
sort	Querystring	String	No	created	The order of the returned plans. See Pagination for more options

4 https://api.flightplandatabase.com/search/plans

```
[
  {
    "id": 59970,
    "fromICAO": "EHAM",
    "toICAO": "KJFK",
    "fromName": "Schiphol",
    "toName": "John F Kennedy Intl",
    "flightNumber": null,
    "distance": 3417.0572325371704,
    "maxAltitude": 38000,
    "waypoints": 37,
    "likes": 0,
    "downloads": 0,
    "popularity": 0,
    "notes": "",
    "encodedPolyline": "yvh~Hgi`\\qxhHjh~Ci_zAnuw@mtSt_Lcgn@hd]wn{{@ldf@kfsDtefDawrEjcpD_l~D|fiDqvwJrtnJ}}yEfzpVoelA~vjH}~iCtapD_pcBjcvMogxCjdbXen
    "createdAt": "2015-07-06T08:06:41.000Z",
    "updatedAt": "2015-07-06T08:06:41.000Z",
    "tags": [
      "generated"
    ],
    "user": null
  }
]
```


3.3 Infraestructura de servicios.

3.3.1 BASES DE DATOS.

Los datos van a ser almacenados usando el sistema de ficheros encriptado IB_ARCHIVO_SECRETO con el que haremos una serie de carpeta a modo de tablas generales.

3.3.1.1 Aeropuertos.

Tendremos un archivo con la nomenclatura ICAO.XML:

ICAO: es el nombre del aeropuerto; por ejemplo, el aeropuerto de Madrid es LEMD.

Nombre modelo. IB_ModeloAirport

3.4 APIS WEB.

Las API con las que vamos a realizar las consultas para poder hacer el seguimiento de los aviones son las que se detallan en los siguientes puntos. Algunas de las APIS son gratuitas, otras lo son con limitaciones y en otras solo nos dan una parte de la información.

Pero la principal, la que nos da la posición de los aviones, es gratuita y nos permite poder hacer

3.4.1 ICAO WEB API

La ICAO ha desarrollado herramientas sofisticadas para recopilar y analizar una amplia gama de métricas de datos de seguridad operacional de la aviación que le permiten identificar riesgos globales existentes y emergentes. La ICAO, junto con sus

192 Estados miembros, trabaja con grupos de la industria para lograr un consenso sobre las normas y los métodos recomendados y las políticas internacionales de aviación civil.

Con la seguridad de la aviación en el núcleo de los objetivos fundamentales de la ICAO, la organización se esfuerza constantemente, en estrecha colaboración con toda la comunidad de la aviación, para mejorar aún más el desempeño exitoso de la seguridad de la aviación, al tiempo que mantiene un alto nivel de capacidad y eficiencia. La OACI supervisa e informa sobre numerosos indicadores de desempeño del sector del transporte aéreo; y audita las capacidades de supervisión de la aviación civil de los Estados en las áreas de seguridad y protección.

3.4.1.1 Información de la posición de los aviones.

Esta es la función API que usaremos continuamente durante la ejecución de la aplicación. Lo que hacemos es llamar al servicio:

http://data-live.flightradar24.com/zones/fcgi/feed.js?bounds=34.300317,30.003735,128.861654,135.054609&adsb=1&air=1&array=1

al realizar esta llamada el sistema no dará todos los track ADS-B que estén dentro del rectángulo definido por las coordenadas cartesianas que le pasamos como argumentos.

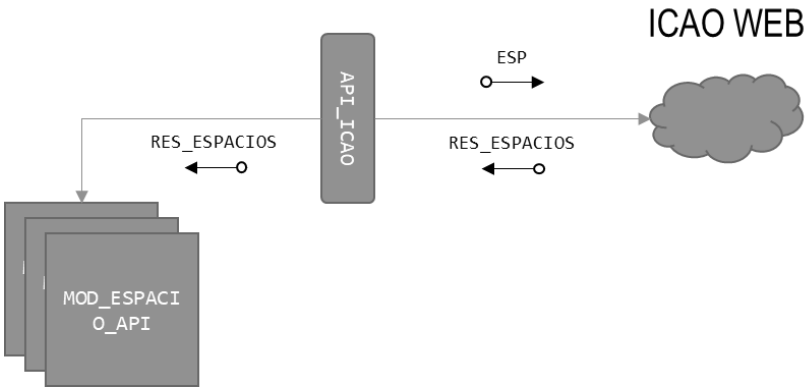
3.4.1.2 Información del histórico de un vuelo.

Con esta función vamos a poder crear una línea con el histórico de un avión; para ello, realizamos la siguiente llamada:

https://data-live.flightradar24.com/clickhandler/?version=1.5&flight=1e22fe3b

3.4.1.3 Información de regiones de vuelo.

Aquí encontraremos una serie de funciones con las que poder dibujar en el GIS regiones aéreas. El comienzo de sus es bastante fácil porque necesitamos saber los ICAO que identifican a las regiones que hay en España. Con estos valores iremos llamando de forma secuencial por cada uno de estos ICAO.



La llamada a la API es la siguiente para conocer las regiones que hay en España:

https://v4p4sz5ijk.execute-api.us-east-1.amazonaws.com/anbdata/airspaces/zones/fir-name-list?api_key=d3e78d90-4de4-11e8-bb63-21ad2a999159&ANRegion=EUR&states=ESP&firs=&format=json

El resultado es algo parecido a esta tabla para el caso de España.

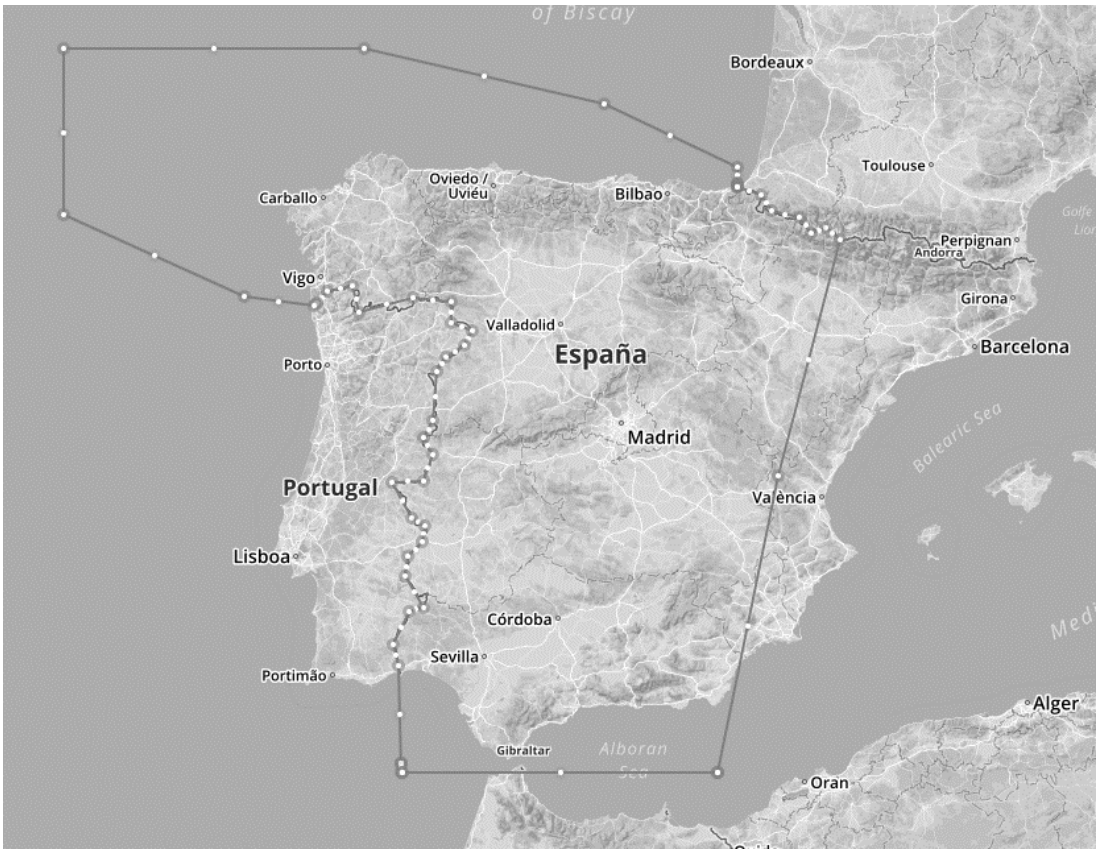
Tabla 1 Espacios aéreos de España

FIRCODE	FIRNAME	REGIÓN	COUNTRYCODE	AREASQKM
LECM	MADRID	EUR	ESP	618262
GCCC	CANARIAS	EUR	ESP	1277821
LECB	BARCELONA	EUR	ESP	271043.202

De la información que nos da esta consulta el FIRCODE es el que necesitamos para saber el polígono que define nuestro sector aéreo o región aérea. Para ello, necesitamos realizar otra llamada a otra de las funciones de la API

https://v4p4sz5ijk.execute-api.us-east-1.amazonaws.com/anbdata/airspaces/zones/fir-list?api_key=d3e78d90-4de4-11e8-bb63-21ad2a999159&firs=LECM&format=json

CENTLAT	REGION	CENTLONG	FTRNAME	ICAOCODE	STATECODE
40.98960632	EUR	-5.5904606	FIR MADRID	LECM	ESP



5 https://www.keene.edu/campus/maps/tool/

ALGORITMO PARA DIVIDIR UN PILIGONO EN PARTES.

3.4.2 Open Sky NetWork

TRACK. Con esta función podemos saber los tracks de un vuelo conocido el icao24.

Ejemplo: https://opensky-network.org/api/tracks/all?icao24=3c4b26&time=0

4 Descripción del componente.

4.1 GIS Mapa.

Como aplicación basada en un GIS este es nuestro componente principal que nos ayudará con la representación de los distintos elementos gráficos con los que realizaremos consultas a las APIS WEB.

Los elementos gráficos están definidos como marcas de gis; estas marcas de GIS, son las que mostraremos objetos gráficos y su posición en el espacio. Para ello, crearemos, una serie de objetos de marcas que tendrán definida una capa en WPF denominada **SHAPES** en la que estará el dibujo de la marca. Este dibujo se presentará centrado por sus

coordenadas; de manera que los gráficos serán de tamaños reducidos tan y como sería una marca en un mapa.

4.1.1 Marca Vuelo.

4.2 Cómo solicitar datos mediante la clase WebRequest

En el procedimiento siguiente se describen los pasos usados para solicitar un recurso de un servidor, por ejemplo, una página web o un archivo. El recurso debe ser identificado por un identificador URI.

Para solicitar datos de un servidor de host

Cree una instancia WebRequest llamando a Create con el URI del recurso.

```
WebRequest request = WebRequest.Create("http://www.contoso.com/");
```

.NET Framework proporciona clases específicas de protocolo derivadas de WebRequest y WebResponse para identificadores URI que empiezan por "http:", "https:", "ftp:" y "file:". Para obtener acceso a recursos con otros protocolos, debe implementar clases específicas de protocolo que se deriven de WebRequest y WebResponse. Para obtener más información, vea Programming Pluggable Protocols (Programar protocolos acoplables).

Establezca los valores de propiedad que sean necesarios en WebRequest. Por ejemplo, para habilitar la autenticación, establezca la propiedad Credentials en una instancia de la clase NetworkCredential.

En la mayoría de los casos, la clase WebRequest es suficiente para recibir datos. En cambio, si necesita establecer propiedades específicas de protocolo, convierta WebRequest al tipo específico de protocolo. Por ejemplo, para obtener acceso a las propiedades de HttpWebRequest específicas de HTTP, convierta WebRequest en una referencia HttpWebRequest. En el siguiente ejemplo de código se muestra cómo se establece la propiedad UserAgent específica de HTTP.

Para enviar la solicitud al servidor, llame a GetResponse. El esquema de URI solicitado determina el tipo real del objeto devuelto de WebResponse.

Cuando haya acabado de usar un objeto WebResponse, debe cerrarlo llamando al método Close. De manera alternativa, si ha obtenido la secuencia de respuesta del objeto de respuesta, puede cerrar la secuencia llamando al método Stream.Close. Si no cierra la respuesta o la secuencia, la aplicación se quedará sin conexiones con el servidor y no podrá procesar más solicitudes.

Puede obtener acceso a las propiedades de WebResponse o convertir WebResponse en una instancia específica de protocolo para leer propiedades específicas de protocolo. Por ejemplo, para obtener acceso a las propiedades de HttpWebResponse específicas de HTTP, convierta WebResponse en una referencia HttpWebResponse. En el ejemplo de código siguiente se muestra cómo mostrar la información de estado enviada con una respuesta.

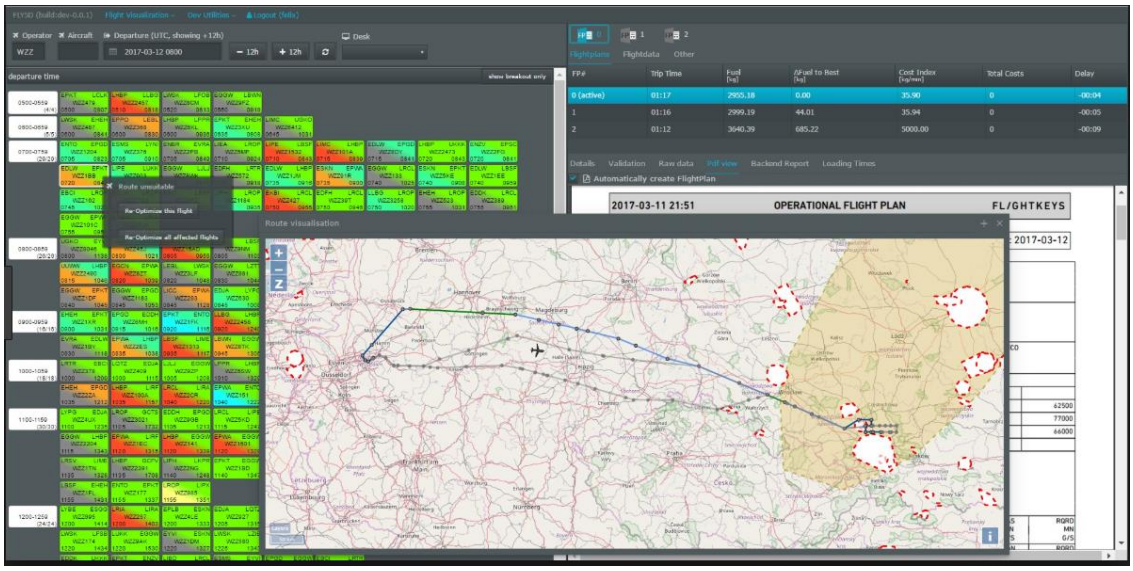
5 Ejemplos de HMI

METEOROLOGICO.

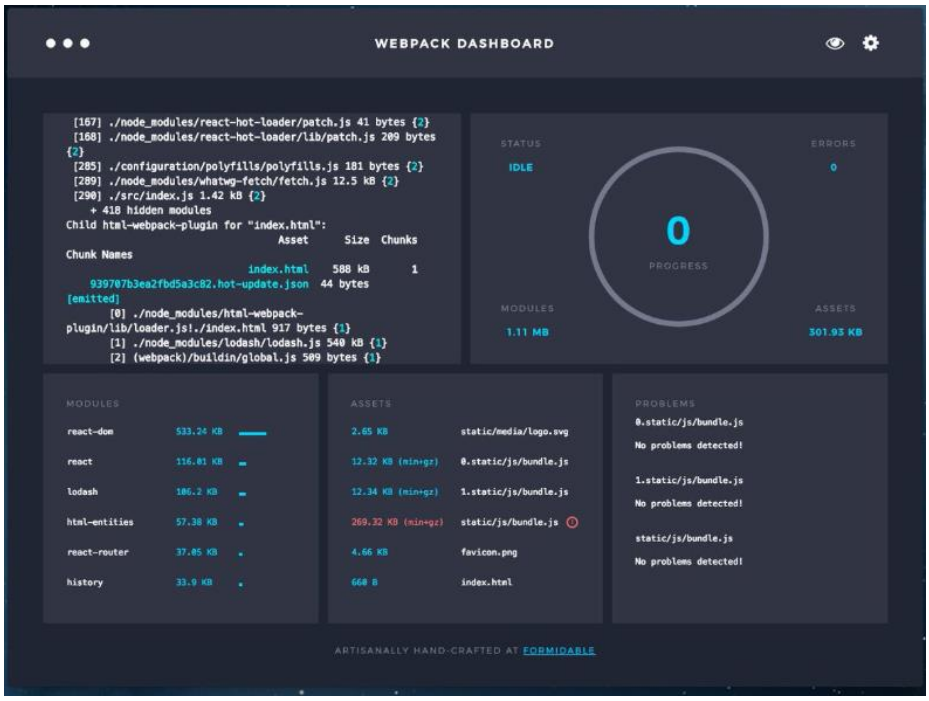
http://tgftp.nws.noaa.gov/data/observations/metar/decoded/LESA.TXT

COMPAÑIAS AEREAS.

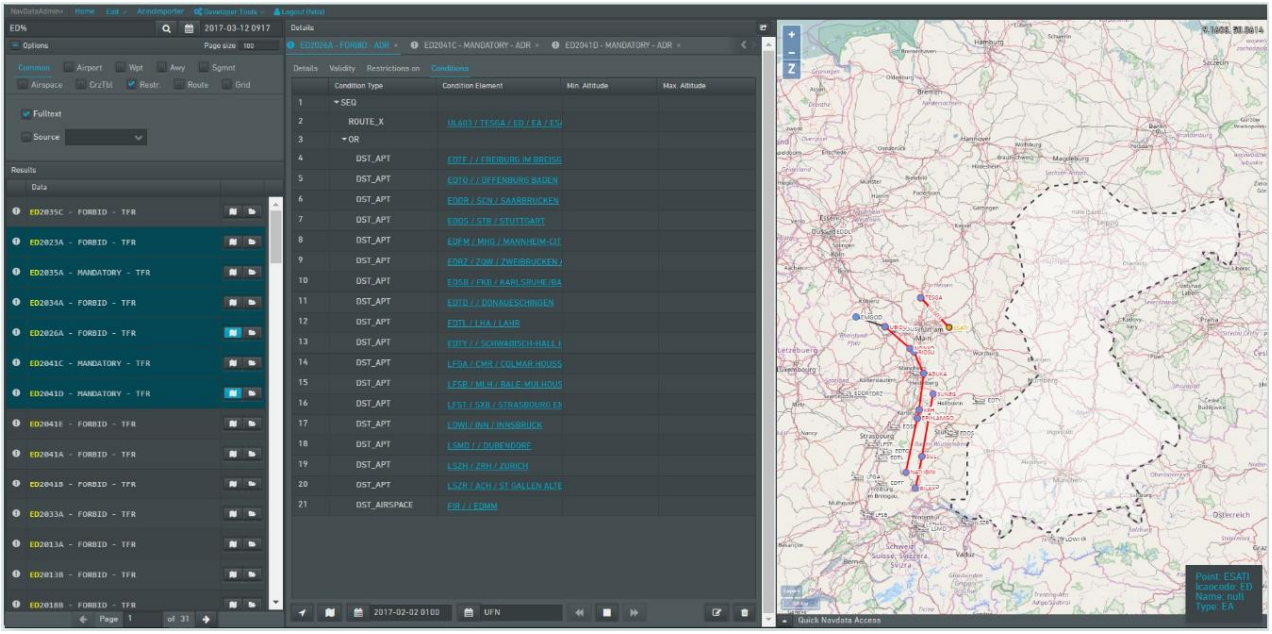
http://www.flightradar24.com/_json/airlines.php



6 http://www.flightkeys.com/



7https://twitter.com/changelog/status/898184163563589632



8http://www.flightkeys.com/

6 Referencias

The Flight Plan Database API . (s.f.). Obtenido de https://flightplandatabase.com/dev/api