APLICACIÓN PARA LA SITUACIÓN GEOGRÁFICA DEL OBSERVADOR MEDIANTE POSICIONAMIENTO ASTRONÓMICO

Germán Núñez Ordiz

Consultor: Xavier Navarro Esteve

Ingeniería Técnica en Informática – 2011/12

Índice

INTRODUCCIÓN:

Típicamente, para situar el buque mediante observación astronómica, se deben realizar numerosos cálculos que llevarán al marino la pérdida de mucho tiempo (hasta una y dos horas), mientras que a un ordenador le supondrán un tiempo de cálculo ínfimo. Por lo que, parece buena idea, el disponer de una aplicación informática que realice estos cálculos automáticamente. No obstante, en la actualidad prácticamente éstas no existen y es por ello, que en este trabajo, se ha querido desarrollar una. Como Licenciado en Náutica que soy he podido aportar los conocimientos de navegación astronómica necesarios para llevar a cabo esta aplicación, así como un enfoque práctico de la forma final y utilidades que debería de presentar una aplicación de este tipo, a fin de conseguir un producto lo más sencillo, práctico e intuitivo posible.

El programa está escrito en el lenguaje orientado a objetos *C#*, haciendo uso de la tecnología de *Microsoft .NET Framework* y se ha denominado como; "SPAC", siglas de; *"Sistema de posicionamiento ayudado por computador"*. Adicionalmente se han realizado unos complementos para Windows Mobile que implementan algunas de las funcionalidades del proyecto principal, con el fin de mostrar algunas de las posibles líneas futuras del proyecto y su implementación en una plataforma más restrictiva.

Durante el trabajo y en apéndices a este se explicarán todos los algoritmos utilizados para posicionar al buque y para generar las coordenadas y las correcciones a las alturas instrumentales tomadas de los distintos astros, así como su fiabilidad y precisión.

En definitiva, se ha querido hacer un programa informático, de fácil uso, que sea lo más práctico posible a la hora de ayudar al marino a hallar la situación del buque, mediante el empleo de las técnicas, ya conocidas, de situación por observación astronómica.

OBJETIVOS Y PLANIFICACIÓN:

1.- Objetivos principales:

En este trabajo final de carrera básicamente se ha realizado una aplicación informática de escritorio desarrollada en *C#* y para *Windows Forms*, que tiene como objetivo principal cubrir los siguientes aspectos:

1. Posicionamiento del observador mediante el método de la recta de altura; más concretamente, posicionamiento por hasta tres rectas de altura (simultáneas o no) y combinaciones de estas con una observación en la Meridiana.

2. Situación por observación en Meridiana.

3. Generación de todos los datos astronómicos necesitados para realizar estos cálculos de forma automática y mediante el uso de algoritmos de muy alta precisión de uso actual en entornos profesionales, estos son;

   3.1. Coordenadas Solares, Lunares y de todos los Planetas del Sistema Solar

   3.2. Coordenadas de todas las estrellas de hasta magnitud visual aparente de 6,5 y las principales entre magnitudes de 6,5 y 8.

   3.3. Cálculo de la predicción de los efectos de nutación y aberración terrestre y precesión de los equinoccios, mediante el uso de algoritmos avanzados.

   3.4. Cálculo de determinados eventos astronómicos.

   3.5. Correcciones a aplicar a las alturas observadas de cualquiera de los astros anteriormente mencionados.

   3.6. Cálculo de datos físicos de todos los astros, tales como; distancia a la Tierra, ángulo de fase, magnitud visual, fracción iluminada, elongación, etc.

4. Construir/mostrar el cielo del lugar; esto es, dada la situación del observador el programa le mostrará/tabulará todos los astros que tenga visibles en ese momento.

5. Añadir opciones de personalización del entorno en el programa para cada usuario del sistema.

En resumen el programa permite además de la mera aplicación para hallar la situación geográfica del marino, otra vertiente, que podíamos denominar como de "observatorio" en la que la idea es ofrecer el mayor número de datos e informaciones relativas a los astros para que el programa pueda usarse como si de un planetario se tratase, ayudándonos así a observar e identificar los astros que nos interesen.

Se ha elegido como nombre para denominar al programa el de; SPAC, siendo este las siglas de; *"Sistema de posicionamiento ayudado por computador"*.

Adicionalmente se implementan unas determinadas funcionalidades de SPAC para la plataforma móvil de Windows Mobile, con el fin de aprovechar las tecnologías que .NET nos brinda en materia de móviles. Las principales funcionalidades, que cubrirá la aplicación para móvil serán;

- Orientación por compás astronómico.
- Efemérides astronómicas de especial interés, como; entrada de estaciones, momento de meridiana Solar o fases lunares.
- La aplicación del móvil, como en la aplicación de PC, permitirá al usuario configurar los parámetros principales del programa, tales como; posición por defecto, errores a considerar en las observaciones, datos físicos de la observación, etc.

2.- Principales motivaciones:

Las principales motivaciones para realizar este trabajo final de carrera, son diversas;

- Por un lado, aprovechar el conocimiento en materia de astronomía y navegación astronómica que puedo aportar como licenciado en náutica.
- Por otro lado, realizar una aplicación hecha a medida para el marino, con el fin de implementar todas las opciones y cálculos que este requiriese a la hora de posicionarse, con el fin de mejorar y aportar seguridad a la navegación.
- Conseguir resultados más óptimos, más rápidos y más fiables de los que seríamos capaces de obtener realizando los cálculos de forma manual.
- Aumentar el número de astros sobre el que poder calcular sus efemérides y por lo tanto sobre el que el marino puede elegir para situarse.
- Aportar un sistema redundante de posicionamiento al de GPS, que sea completamente automatizado y que pueda dar el posicionamiento del marino de forma fiable e inmediata, sin tener que lidiar con largos y tediosos cálculos.
- Realizar un tipo de aplicación original y singular, de la que actualmente no existe ninguna igual en el mercado.
- Aprovechar las tecnologías ofrecidas por la plataforma .NET para crear una aplicación de uso simple e intuitivo a la vez que potente y práctica.

3.- Análisis de riesgos:

De forma inicial se han estudiado los principales riesgos que se pueden presentar durante el desarrollo del proyecto y pueden ser clasificados en tres grandes apartados;

- **Riesgos en la algorítmica:** en el peor caso, podría ocurrir, que alguno de los algoritmos a implementar de cálculo astronómico de las coordenadas de algún astro, no sea asumible para el tiempo fijado para el proyecto. Esto es, por ser extremadamente

difícil de desarrollar ó por no hallarse bien documentada una solución algorítmica válida para implementar aquello que necesitamos.

Para evitar esta situación se ha realizado un estudio previo de los algoritmos de cálculo astronómico presentes actualmente para cada tipo de astro, viendo que en cada uno de ellos "aparentemente" existe un algoritmo asumible para el tiempo fijado del proyecto a implementar.

Se insiste en este punto, ya que, tal vez una de las grandes dificultades de este proyecto sea la implementación informática de dichos algoritmos necesarios para cubrir los objetivos del proyecto, no obstante gracias a las tecnologías ofrecidas por .NET tendremos múltiples recursos para intentar subsanar este problema.

- **Riesgos en las tecnologías elegidas:** hay que considerar un período de aprendizaje de cada una de las tecnologías .NET que haya que aplicar en el presente proyecto. Con el consiguiente riesgo de caer en retrasos debidos a una asimilación lenta o más costosa de aquello que se había previsto en un inicio. Sobre especial atención hay que considerar imprevistos a nivel de aprendizaje y dificultades diversas a la hora de implementar sobre *Windows Mobile*, por tener que trabajar sobre un entorno mucho más restrictivo y cerrado al del PC, como es el de *.net Compact Framework.*

- **Riesgos físicos:** podría pasar que nos encontráramos con impedimentos de hardware a la hora de realizar todas las funciones que el programa pretende abordar, debido al número de operaciones y cálculos complejos que pueden llegar a demandar los algoritmos de cálculo de posicionamiento astronómico y generación de coordenadas de los astros. Ante esta situación nos encontraríamos con que la salida esperada del programa vendría retrasada por un cierto intervalo de tiempo, que dependiendo de su magnitud podría ser asumible o no. Esta situación tendrá especial riesgo en el desarrollo bajo plataforma móvil por esta estar dotada siempre de un procesador y memoria mucho más reducidos que los que puede ofrecer un PC.

4.- Requisitos del proyecto:

El proyecto principal se realizará como una aplicación de tipo; *Windows Forms* escrita en *C#* bajo *Visual Studio 2010* i orientado hacia la versión *4.0* de *.NET*.

En cuanto al proyecto para *Windows Mobile* se realizará también en *C#* pero en *Visual Studio 2008* y como un proyecto orientado hacia *Windows Mobile 6.0 o superior*. El motivo de elegir la versión de *Visual Studio 2008* y no la *2010* para el desarrollo en móvil es simple; el desarrollo para *Windows Mobile* no se encuentra soportado en *Visual Studio 2010* mientras que sí que lo está en *Visual Studio 2008*. El desarrollo para móvil se realizará bajo la última versión estable conocida del *Compact Framework* de *.NET*, es decir, la versión; *3.5 .NET Compact Framework.*

La relación de software de desarrollo necesario a lo largo del proyecto sería la que sigue:

- **Visual Studio 2010.**
- **Microsoft Framework .NET 4.0**

- **Visual Studio 2008 con complementos para Windows Mobile 6.0 o superior.**
- **Windows Mobile 6 Professional SDK Refresh**
- **Windows Mobile 6.5.3 Professional DTK**
- **Windows Mobile 6.5.3 Professional Images**
- **Microsoft Compact Framework .NET 3.5**
- **Microsoft Office 2010 para la realización de la memoria y la presentación.**

Así mismo se dispondrá de un teléfono móvil *HTC HD mini* con *Windows Mobile 6.5* para poder realizar las pruebas y ver el comportamiento de la aplicación en un teléfono real y no solo en el emulador de *Microsoft*.

Como equipo de trabajo y desarrollo se dispondrá de un PC de sobremesa con *Windows Vista Business de 64 bits* y con un procesador *Intel i7 920 a 2,67 GHz* con *6 GBytes* de *RAM*. También se dispondrá de máquinas virtuales *VMWare* con; *Windows Vista 32 bits*, *Windows 7* y *Windows XP,* para poder realizar las correspondientes pruebas de compatibilidad en cada una de estas plataformas, a fin de poder dar soporte para las tres versiones de *Microsoft Windows* más utilizadas actualmente.

5.- Planificación del proyecto:

La planificación temporal del proyecto va marcada en base a las fechas de presentación de cada una de las PAC asociadas. De modo que, podríamos dividir el proyecto en cuatro grandes etapas;

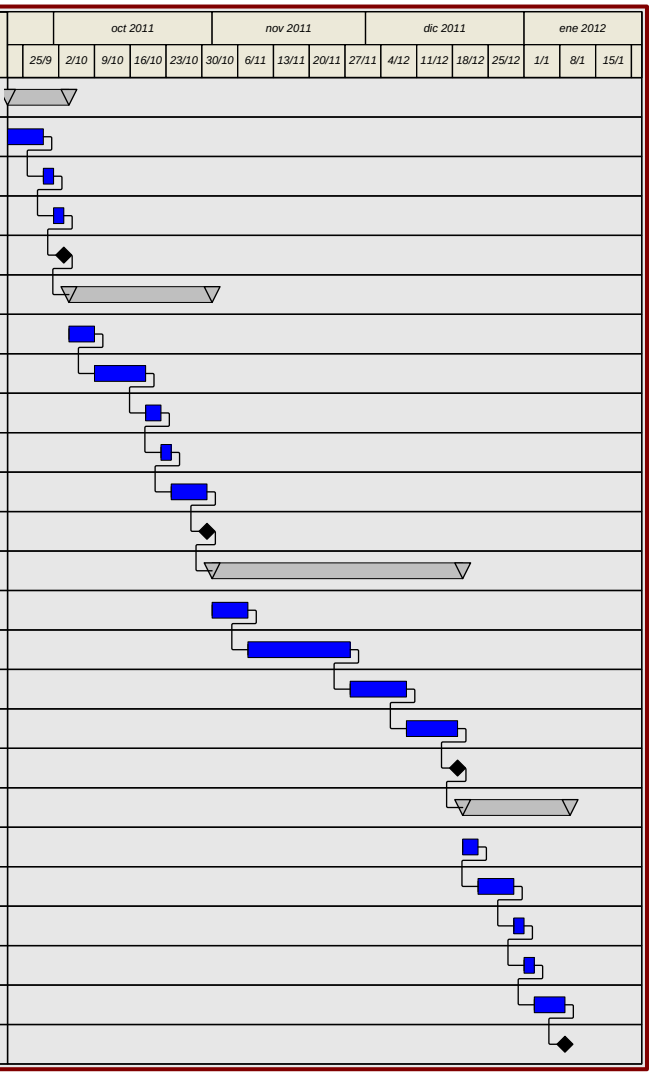
- **Etapa 1: Presentación, descripción y planificación del proyecto:**
  Del 22 de septiembre al 3 de octubre.
  - Presentación y descripción de los objetivos principales a cubrir por el proyecto.
  - Análisis de requisitos y análisis de riesgos.
  - Instalación del entorno de desarrollo necesario para abordar el proyecto.
  - Planificación temporal del proyecto y elaboración del diagrama de Gantt.
  - Elaboración y presentación de la PAC1.
- **Etapa 2: Análisis y diseño del proyecto:**
  Del 4 al 31 de octubre.
  - Estudio de los principales algoritmos astronómicos y de posicionamiento a utilizar y elección de aquellos que necesitaremos implementar en nuestro programa.
  - Elaboración del diagrama de clases del programa.
  - Elaboración de diagramas UML adicionales.
  - Diseño del entorno visual de la aplicación.
  - Elaboración y presentación de la PAC2.
- **Etapa 3: Implementación del proyecto:**
  Del 1 de noviembre al 19 de diciembre.
  - Implementación de los algoritmos astronómicos y de posicionamiento estudiados anteriormente en la etapa de análisis y diseño.
  - Implementación del programa principal bajo *Windows Forms*.

- Implementación de los complementos para móvil bajo *Windows Mobile*.
- Elaboración y presentación de la PAC3.

- **Etapa 4: Documentación y presentación:**
  Del 20 de diciembre al 9 de enero de 2012.
  - Documentar algoritmos astronómicos y de posicionamiento.
  - Documentar opciones y funcionalidades del programa principal.
  - Documentar complementos realizados para móvil.
  - Elaborar la presentación final del TFC.
  - Realizar la entrega final del TFC.

5.1.- Diagrama de Gantt:

Id.	Nombre de tarea	Comienzo	Fin	Duración
1	**ETAPA 1: Presentación, descripción y planificación del proyecto**	22/09/2011	03/10/2011	12d
2	Elaboración de la propuesta del TFC	22/09/2011	28/09/2011	7d
3	Elaboración del plan de trabajo	29/09/2011	30/09/2011	2d
4	Instalación del entorno de desarrollo	01/10/2011	02/10/2011	2d
5	*Entrega PAC1*	03/10/2011	03/10/2011	0d
6	**ETAPA 2: Análisis y diseño del proyecto**	04/10/2011	31/10/2011	28d
7	Estudio del algoritmo de posicionamiento por recta de altura	04/10/2011	08/10/2011	5d
8	Estudio y elección de los algoritmos de generación de coordenadas de los astros	09/10/2011	18/10/2011	10d
9	Diseño del diagrama de clases	19/10/2011	21/10/2011	3d
10	Diseño de otros diagramas UML	22/10/2011	23/10/2011	2d
11	Diseño del entorno visual de la aplicación	24/10/2011	30/10/2011	7d
12	*Entrega PAC2*	31/10/2011	31/10/2011	0d
13	**ETAPA 3: Implementación del proyecto**	01/11/2011	19/12/2011	49d
14	Implementación de algoritmos de posicionamiento astronómicos	01/11/2011	07/11/2011	7d
15	Implementación de los algoritmos de generación de coordenadas de los astros	08/11/2011	27/11/2011	20d
16	Implementación de SPAC bajo Windows Forms	28/11/2011	08/12/2011	11d
17	Implementación de las funcionalidades requeridas de SPAC para plataformas móviles	09/12/2011	18/12/2011	10d
18	*Entrega PAC3*	19/12/2011	19/12/2011	0d
19	**ETAPA 4: Documentación y presentación**	20/12/2011	09/01/2012	21d
20	Documentar algoritmos de posicionamiento	20/12/2011	22/12/2011	3d
21	Documentar algoritmos de generación de coordenadas de los astros	23/12/2011	29/12/2011	7d
22	Documentar opciones y funcionalidades de SPAC	30/12/2011	31/12/2011	2d
23	Documentar módulos de SPAC para móvil	01/01/2012	02/01/2012	2d
24	Preparar presentación del TFC	03/01/2012	08/01/2012	6d
25	*Entrega final del TFC*	09/01/2012	09/01/2012	0d



ANÁLISIS Y DISEÑO:

1.- Análisis de requisitos funcionales:

La motivación de SPAC es realizar un software que ayude al marino a posicionarse únicamente teniendo un sextante y un reloj, es decir, únicamente conociendo la altura de un astro y el momento en el que este fue observado.

Por lo tanto SPAC consistirá en un programa en el que el usuario entrará hasta tres observaciones consecutivas o no de tres astros que pueden ser a su vez, todos ellos, el mismo astro o no.

Cuando se inicie SPAC el usuario tendrá la opción de interactuar con él para conocer datos de interés de los astros que actualmente tiene en su cielo, o por el contrario, utilizarlo para posicionarse. La primera opción permite que SPAC se comporte como si de un planetario se tratase mientras que la segunda permite posicionar al usuario.

Si el usuario desea posicionarse deberá añadir al menos una observación de un astro cualquiera, para esa observación el usuario deberá introducir a SPAC, como mínimo los siguientes datos;

- Astro observado (el nombre que lo identifique).
- Altura tomada con el sextante.
- Hora en que fue tomada dicha altura.
- La situación estimada en la que cree estar, o por el contrario, si esta no se conociese y siempre que el astro observado estuviese en su momento de meridiana deberá indicarle al programa que pese a no conocer su situación estimada sí que sabe que dicha observación de ese astro fue tomada en su momento de culminación (meridiana), a lo que el programa, le pedirá un último dato adicional que será si el astro observado en el momento de la meridiana fue visto cara al norte o cara al sur.

Por otra parte, el programa no permitirá el posicionamiento por más de tres observaciones, por ser conocido por el marino que el posicionamiento con un número mayor a tres observaciones induce unos errores en la situación final calculada bastante elevados y por tanto siendo tres observaciones el caso óptimo para un posicionamiento fiable. De igual forma permitirá posicionarse si:

- El usuario solo introduce una observación y esta es realizada a un astro que en dicho momento está en su meridiana.
- El usuario introduce dos observaciones cualesquiera y una de ellas o ninguna de ellas es tomada en el momento de la meridiana.

El programa hará una serie de cálculos para informar de cómo de fiable es cada observación introducida, de modo que;

- Si hay un incremento de altura mayor a 20 millas indicará la falta de fiabilidad de la observación realizada y recomendará repetirla.

- Si la altura del astro es negativa indicará al usuario que dicha observación no puede añadirse por estar el astro bajo su horizonte, es decir, por ser una observación imposible de realizar en la realidad y por tanto el usuario deberá de observar otro astro para continuar.
- Si hay azimuts paralelos el programa alertará que se está entrando una observación que genera un azimut que es paralelo con alguna otra observación previamente entrada, dado que así es imposible la situación indicará al usuario que debe de repetirla.
- Si la fase lunar impide la observación de la Luna que supuestamente el usuario habría introducido como realizada.

El programa informará al usuario tras añadir cada observación si con las que tiene presentes ya es capaz de posicionarlo o no, pudiendo elegir el usuario de posicionarse con solo las que tiene hasta el momento o añadir alguna otra para dotar la posición final calculada de mayor fiabilidad.

El usuario podrá; guardar, abrir y editar todos sus cálculos astronómicos, considerándose como tal aquel compendio de observaciones realizadas para poder hallar su posición final. Los cálculos astronómicos serán guardados en un archivo con un formato determinado. SPAC también permitirá que el usuario pueda imprimir un cálculo astronómico.

En la parte de observatorio, SPAC, trabajará con una configuración básica sobre la que calcular las efemérides y demás datos astronómicos de interés. Dicha configuración básica será la que se constituya a partir de las preferencias del programa y estas a su vez vendrán determinadas por los siguientes campos:

- Situación geográfica para generar las coordenadas
- Hora a utilizar para generar las coordenadas y zona horaria.
- Elevación sobre el nivel del mar en la que se encuentra el usuario.
- Presión atmosférica.
- Temperatura.
- Error de índice del sextante que utiliza el usuario.
- Formato en que quiere que se le presenten los resultados.
- Cálculo de efemérides para una fecha juliana determinada.
- Cambio del valor de la Delta de T que internamente debe usar el programa.
- Tipo de hora que quiere que SPAC le pregunte de forma predeterminada (UTC, Local o TD).

Se implementarán las funcionalidades siguientes para Windows Mobile 6.0 o superior:

- Obtener los datos del posicionamiento de un astro demandado para un momento dado. Esto es poder buscar entre los astros que maneja SPAC (Sol, Luna, Planetas y Estrellas) y mostrar los datos de dicho astro para el momento deseado.

- Función ¡Oriéntame! el usuario entrará su posición actual y el programa le mostrará dónde está el norte verdadero mediante la implementación de un compás astronómico (el

propio programa ya calcula si para la posición del observador es de día o de noche o si siendo de noche hay luna llena para saber sobre que astro deberá de orientar al usuario).

- Mostrar la fase lunar para una fecha dada.

- Determinar el momento de entrada de cada estación para un año dado.

- Mostrar el cielo del lugar. Esto es enumerar los astros que el usuario puede ver desde su posición actual en cada momento.

2.- Modelo conceptual:

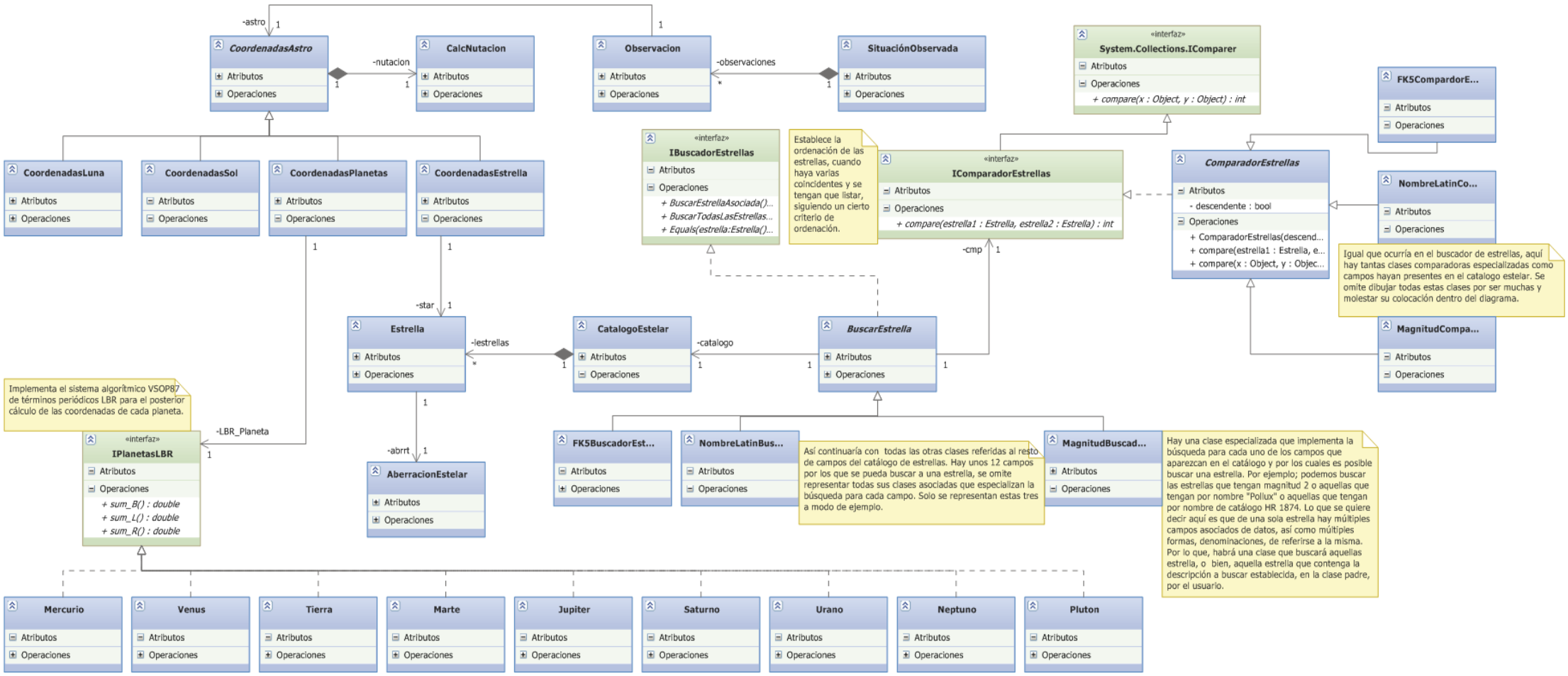
El éxito de este proyecto se vuelca no solo en el análisis informático de requisitos sino también en aquellos conceptos básicos de astronomía necesarios para llevarlo a cabo. Dichos conceptos básicos son los algoritmos de cálculo astronómico tanto de posicionamiento como de generación de coordenadas y efemérides de los astros gracias a los que podremos posicionarnos. Por lo tanto, antes de nada habrá que analizar dichos algoritmos y conceptos clave para poder entender las necesidades conceptuales del proyecto que posteriormente se explicaran con mayor detalle.

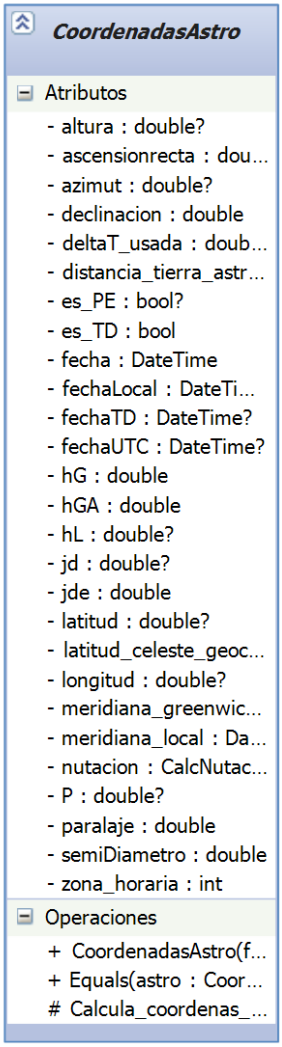
No obstante del anterior análisis de requisitos formales vemos que intervienen en juego, además de muchas otras, las siguientes entidades;

- Astro
- Observación
- Tipo de observación
- Cálculo astronómico
- Situación
- Coordenadas astronómicas
- Efemérides astronómicas
- Tiempo
- Preferencias del usuario

Como principal y único actor vemos al usuario que interacciona con el sistema, dicho actor lo denominaremos de ahora en adelante con el nombre de; **usuario**.

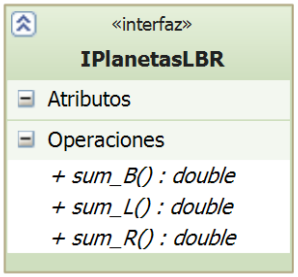
2.1.- Diagrama de Clases:





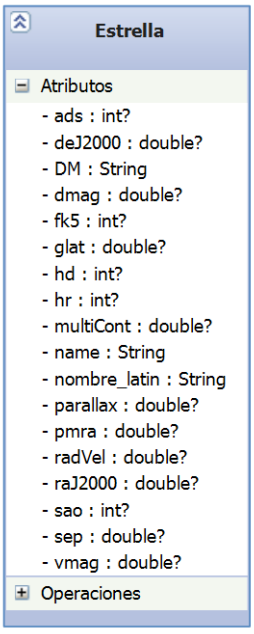
El anterior diagrama se ha realizado como un proyecto de modelado de Visual Studio 2010 y forma parte de la solución principal de Visual Studio del proyecto de SPAC. Como **principales observaciones** tendríamos las siguientes:

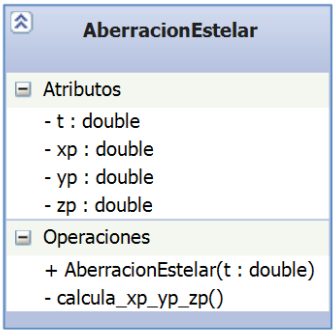
- Hay una clase abstracta *CoordenadasAstro* que es especializada por cada tipo de astro con el que puede trabajar nuestro programa; Luna, Planetas, Estrellas y el Sol. El motivo de esta especialización es debido a que para cada tipo de astro el comportamiento a la hora de calcular sus coordenadas es diferente. Además, pueden aparecen nuevos atributos especializados que no pueden contemplarse de forma general para todos los astros, sino que únicamente para aquellos de ese tipo. Por ejemplo, solo la luna tiene fase lunar.
- En el caso de los planetas, sus coordenadas, son calculadas según la teoría VSOP87, esta requiere de una serie términos periódicos (del orden de los millares) que dependen del planeta que estemos tratando, es decir, son diferentes para cada planeta. El cálculo de los términos periódicos de cada planeta se realiza a través de la interfaz "*IPlanetasLBR*". Para cada Planeta tendremos una clase que implemente la anterior interfaz, en la que se materializan los valores para los atributos de las series LBR y el método especializado que implementa el cálculo de la sucesión de términos LBR para ese planeta.



Así pues las coordenadas de cada planeta quedarán determinadas en la instanciación de la clase CoordenadasPlaneta (que hereda de CoordenadasAstro), a la que deberemos decirle para qué planeta queremos que calcule las coordenadas en aquella instancia, eso se hace mediante un literal que indica el nombre del Planeta para el que hay que calcular sus coordenadas. Véase que esta clase en función del planeta a calcular instanciará la clase Planeta de las que implementen la interfaz *IPlanetasLBR* anteriormente para aquél planeta que le hayamos pedido las coordenadas.

- Las clases *CoordenadasLuna* y *CoordenadasSol* heredan también de la clase abstracta *CoordenadasAstro* de modo que en ellas se implementarán los métodos y atributos especializados para calcular las coordenadas de dichos astros.
- La clase *CoordenadasEstrella* también hereda de *CoordenadasAstro* y especializa el cálculo de coordenadas para cualquier estrella. Entre otros tiene como atributos especializados, uno denominado como *star* de tipo *Estrella*. Dicha clase *Estrella* encapsula los datos que definen y permiten calcular las coordenadas de una determinada estrella.

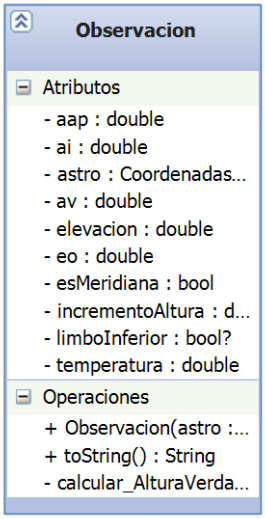




En la clase *CoordenadasEstrella* también encontramos el atributo *abrrt* de la clase *AberraciónEstelar.* Como su nombre indica, esta clase, implementa la funcionalidad y los atributos para el cálculo del efecto de la aberración estelar, que se ha de tener en cuenta

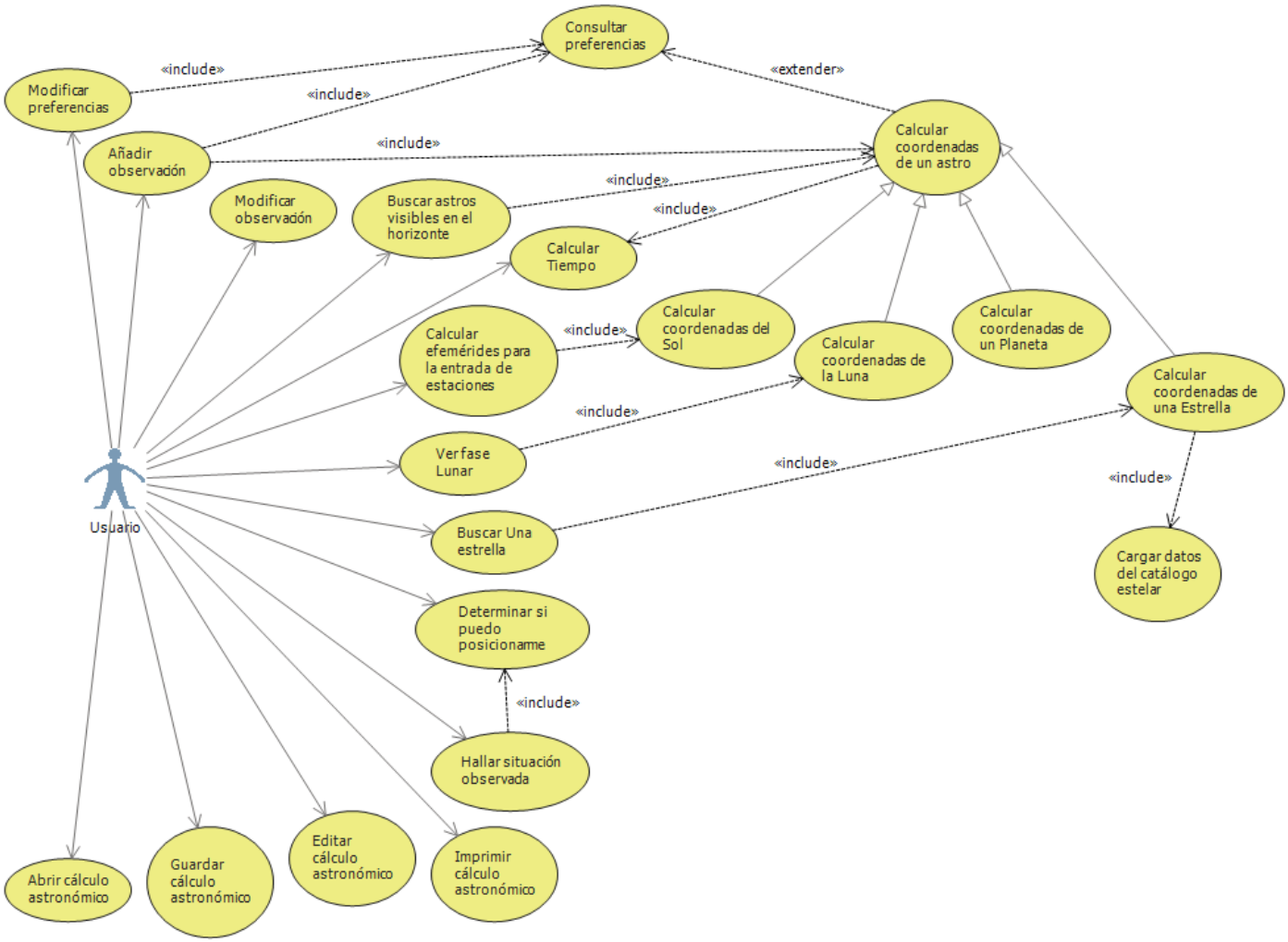
para calcular las coordenadas de una estrella.

- La clase *BuscarEstrella* permite la búsqueda de cualquier estrella de entre las 9110 que el programa maneja. Esta clase implementa tanto la interfaz *IBuscadorEstrellas* como la interfaz *IComparadorEstrellas*. La primera tiene la misión de exponer las funcionalidades para buscar una estrella (target) dentro de las que maneja el programa, o bien, para listar todas aquellas que responden a un determinado criterio de búsqueda (match). La segunda interfaz hereda de la interfaz de *.NET IComparer* y especializa un método expuesto *compare(…):Int*, de modo que, esta interfaz implementa todos los comparadores, para cada uno de los campos con los que es posible definir o buscar una estrella, con el fin de que luego se puedan utilizar para ordenar y presentar las listas según el criterio de ordenación dado por el usuario (instanciación de alguna clase que implemente esta interfaz; *IComparadorEstrellas*).
- *CatalogoEstelar* representa la clase contenedora de todas las estrellas que maneja el programa.
- Cada una de las observaciones realizadas por el usuario serán una instancia de la clase *Observacion*. En ella se encuentran todos los atributos que determinan inequívocamente una observación a un astro determinado. Por lo tanto esta clase tendrá como atributo siempre una instancia de la clase *CoordenadasAstro* con las coordenadas del astro con el que realizó el usuario dicha observación.



- El conjunto de observaciones realizadas por el usuario para posicionarse, así como los métodos que permiten el cálculo de la posición final del observador, se modelan en el la clase contenedora de observaciones; *SituaciónObservada.* Que con el método *hallar_situacion()* calculará en base a las observaciones añadidas por el usuario su situación final geográfica. Por tanto, es en esta clase donde se implementarán propiamente los algoritmos para el cálculo del posicionamiento astronómico final del usuario.

2.2- Diagrama de casos de uso:



Del diagrama de casos de uso podemos destacar la herencia presentada en el caso de uso; "Calcular coordenadas de un astro" dicha herencia indica que el usuario puede, si bien pedir las coordenadas de cualquier astro, también pedir aquellas para un determinado tipo de astro; Sol, Luna, Planetas y Estrellas. Por tanto, deben de contemplarse como alternativas especializadas que el actor Usuario puede ejercer dentro del programa como cualquier otra. También tendremos que cuando el Usuario ejecute los tres casos de uso; "Calcular efemérides para la entrada de estaciones", "Ver fase Lunar" y "Buscar una estrella", se ejecutarán simultáneamente y forzosamente (*include*) el caso de uso especializado de "Calcular Coordenadas de un astro" que tenga asociado aquel caso de uso principal que el actor Usuario ha demando al sistema. Así pues para calcular las efemérides de las estaciones, forzosamente, el sistema habrá de ejecutar también el caso de uso "Calcular coordenadas del Sol", por ser el Sol aquel astro que determina la entrada de las estaciones terrestres.

Los casos de uso relativos a la gestión de archivos de cálculos astronómicos son; Abrir cálculo astronómico, Guardar cálculo astronómico y Editar cálculo astronómico.

Los casos de uso referidos a la gestión de las preferencias del entorno de la aplicación son los de; Consultar preferencias y Modificar Preferencias. A continuación se indican detallados los principales casos de uso del programa:

Modificar preferencias:	
Resumen:	Modifica las preferencias del programa a petición del usuario.
Actor:	Usuario
Casos relacionados:	Consultar preferencias.
Precondición:	Las preferencias están cargadas
Pos condición:	Las preferencias modificadas son actualizadas en el sistema.
Descripción:	1. Se ejecuta el caso de uso Consultar preferencias. 2. Se presenta al usuario el formulario Opciones de SPAC… con las preferencias actuales del sistema. 3. El usuario modifica aquellos campos de las preferencias que quiera modificar y cierra el formulario. 4. Se actualizan las preferencias de programa y se serializan las nuevas preferencias en el directorio de Windows de ese usuario.

Añadir observación:	
Resumen:	Añade una observación por la que poder situarse el usuario en el sistema.
Actor:	Usuario
Casos relacionados:	Consultar preferencias y Calcular coordenadas de un astro.
Precondición:	Hay menos de tres observaciones en el sistema.
Pos condición:	Hay una nueva observación en el sistema.
Descripción:	1. El usuario indica que quiere añadir una observación. 2. El sistema comprueba que hay menos de tres observaciones añadidas en el sistema. En caso afirmativo deja lo deja continuar, sino, finaliza el caso de uso. 3. El sistema presenta el formulario de añadir observación. 4. El usuario añade los datos pertinentes a esa observación. 5. El sistema comprueba que dichos datos sean válidos, pudiendo pasar: a. Si hay un incremento de altura mayor elevado alerta de la poca fiabilidad e la observación introducida y da a elegir al usuario si quiere continuar o no.

b. Si la observación se tomó en la meridiana del astro y hay alguna otra observación previamente introducida que ya se tomara en la meridiana del astro, entonces no se añadirá la observación presente y se dará la opción de rectificar los datos entrados.
c. Si hay azimuts paralelos con alguna otra observación ya entrada no se dejará continuar y se dará la opción de rectificar los datos entrados.
d. Por el contrario se introducirá la observación correctamente y se actualizará la lista de observaciones en *SituacionObservada* por las que poderse posicionar el usuario.

Hallar situación observada:	
Resumen:	Obtiene la latitud y longitud del observador, es decir, lo posiciona geográficamente.
Actor:	Usuario
Casos relacionados:	Determinar si puedo posicionarme.
Precondición:	Hay suficientes observaciones para calcular la posición y la posición del usuario es desconocida.
Pos condición:	La posición del usuario es conocida.
Descripción:	1. El usuario indica que quiere hallar su posición. 2. El sistema ejecuta el caso de uso; Determinar si puedo posicionarme. 3. Si puede posicionarse, el sistema calcula su posición mediante *SituacionObservada*. Si no puede informará al usuario que deberá de introducir más observaciones para poder continuar. 4. El sistema presenta un formulario con la situación final del usuario.

Calcular coordenadas de un astro:	
Resumen:	Obtiene las coordenadas de un astro.
Actor:	Usuario
Casos relacionados:	Calcular tiempo, consultar preferencias, calcular coordenadas Sol, Luna, Planetas y Estrellas.

Precondición: Las coordenadas no son conocidas.

Pos condición: Las principales coordenadas del astro son ahora conocidas.

Descripción:

1. El usuario indica que quiera hallar las coordenadas de un determinado astro.
2. Se calculan las coordenadas del astro según si el usuario indica su situación o no.
    a. Si indica su situación entonces se calcularan también sus coordenadas horizontales.
    b. Si no se indica, entonces se calcularán todas las coordenadas del astro excepto las horizontales.
3. Se presentan las coordenadas al usuario.

Excepciones:

AstroException.TierraNoDistanciaAstro = "No existe distancia al astro cuando el astro es la misma Tierra."

AstroException.TierraNoParalaje = "La Tierra no tiene paralaje"

AstroException.TierraNoSD = "La Tierra no tiene semi-diámetro."

AstroException.TierraNoMeridianaLocal="La Tierra no tiene momento de meridiana local."

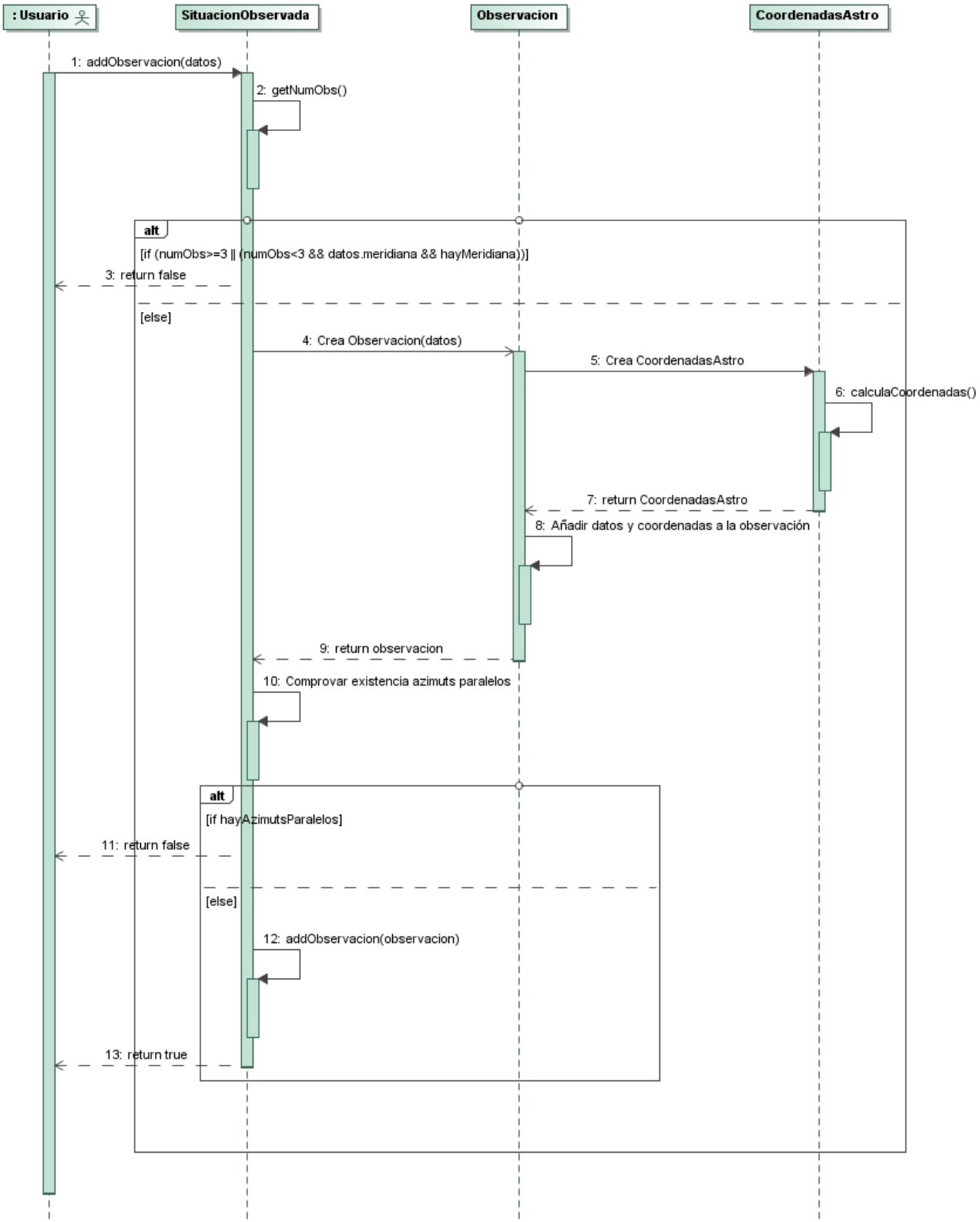
AstroException.TierraNoMeridianaGreenwich = "La Tierra no tiene momento de meridiana en Greenwich."

AstroException.TierraNoAzimut = "La Tierra no tiene azimut"

AstroException.TierraNoAltura="La Tierra no tiene altura." AstroException.TierraNoCoordenadasEquatoriales="La Tierra no tiene coordenadas ecuatoriales para ella misma."

2.3.- Diagrama de secuencias:

Se ha representado el diagrama de secuencias asociado al caso de uso de *Añadir observación*:

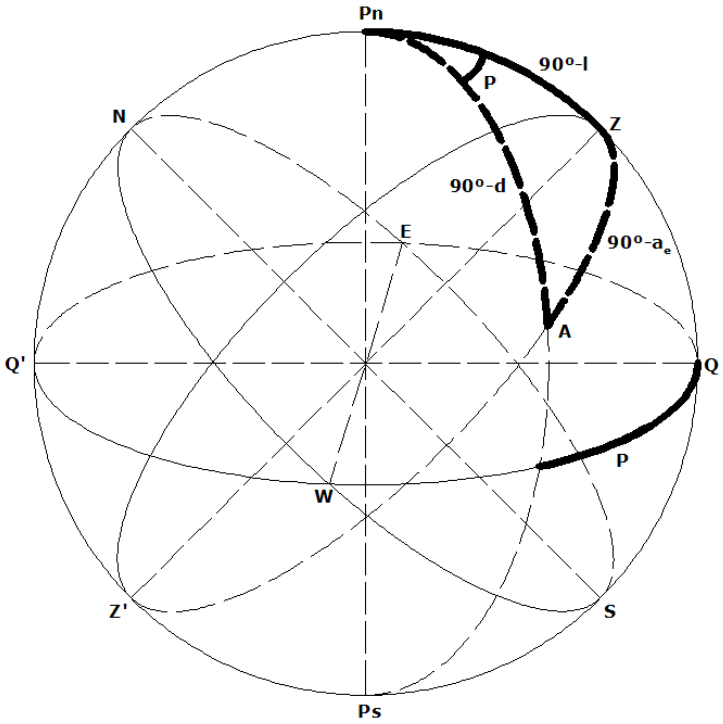


Cuando un usuario desea añadir la observación le pasa al método *addObservación()* de la clase contenedora *SituacionObservada* una *tupla* con los datos que constituyen la observación, es

decir, los datos que ha entrado el usuario en el programa. En ese momento *SituaciónObservada* comprueba el número de observaciones que tenía añadidas previamente el usuario, si dicho número es mayor o igual a tres, o bien, el usuario ya realizó una observación en la meridiana del astro y ahora quiere añadir otra observación tomada en la meridiana del mismo astro o de cualquier otro, entonces no se le permitirá añadir más observaciones y se contestará con *false* como respuesta a que no se ha añadido la observación caracterizada por esos datos. Si por el contrario se está en condición de añadir una nueva observación, se generarán las coordenadas del astro que se observó y se comprobará que no haya ninguna otra observación que tenga un azimut paralelo al que se ha obtenido al calcular los datos del astro para esta última observación. Si lo hubiese, entonces todo el proceso se pararía y dicha observación no llegaría a introducirse en el sistema contestando con un *false* como respuesta a su petición de añadir una nueva observación. Por el contrario, si no hay azimuts paralelos, la observación se añadirá a la lista de observaciones de *SituacionObservada* y se finalizará el proceso correctamente.

3.- Análisis de algoritmos:

Adjunto a este trabajo se incorpora un anexo dónde se detallan con mayor claridad los algoritmos y bases astronómicas necesarias para entender el trabajo realizado. Una parte muy importante del trabajo será el análisis previo de todos los algoritmos astronómicos a utilizar en la siguiente fase de implementación. Es decir, aquí se asentarán las bases de estos algoritmos y se describirá cada uno de ellos.



La resolución de cualquier problema de posicionamiento astronómico pasa por resolver el anterior triángulo esférico (remarcado en negrita). En los siguientes apartados veremos cómo podemos hacer esto para posicionarnos.

3.1.- Algoritmos utilizados para posicionar al observador:

La resolución típica de cualquier problema de posicionamiento astronómico pasa por la resolución gráfica, pero en programación se deben buscar métodos de resolución analíticos a fin de encontrar resultados más precisos y que requieran de menos consumo de recursos del sistema para hallar la solución al problema, de los que podrían necesitarse al emplear gráficos.

Así pues se han estudiado las resoluciones gráficas de todos los casos de los problemas de posicionamiento astronómico, para poder crear una resolución analítica que satisficiera a cada uno de ellos.

A continuación se exponen los casos, sus representaciones gráficas (si tienen) y el correspondiente algoritmo que los resuelve:

3.1.1.- Posicionamiento por un astro:

Observando un solo astro solo podremos posicionarnos con precisión en latitud. Para ello el programa nos permite posicionarnos con un solo astro solo cuando este esté pasando por el meridiano superior del lugar (momento de la meridiana del astro). Se recuerda que el meridiano superior del lugar es aquel que contiene los polos y el zenit. Si dicho astro fuese el sol en este momento sucedería el mediodía.

El paso del astro por el meridiano superior del lugar, implica que el ángulo en el polo *"P"* sea nulo y que la altura del astro (a) sea máxima. Al ser el ángulo en el polo nulo se tendrá lo siguiente:

$$sen(a) = sen(l) * sen(d) + \cos(l) * \cos(d) * \cos(P)$$

Si $P = 0º$, entonces se tiene que:

$$sen(a) = sen(l) * sen(d) + \cos(l) * \cos(d) \Rightarrow sen(a) = \cos(d - l)$$

Donde;

$$z = 90º - a \Rightarrow a = 90º - z$$

Substituyendo:

$$sen(90º - z) = \cos(d - l) \Rightarrow cos(z) = \cos(d - l)$$

De lo que resulta:

$$\boxed{\boldsymbol{l = d - z}}$$

Donde *"**l**"* es la latitud observada, es decir en aquella que se halla el observador.

La distancia cenital *"z"* tendrá valor positivo si se observa el astro hacia el norte (se dice cómo; cara al norte), implica que su azimut es norte verdadero, y tendrá valor negativo y si se observa el astro hacia el sur (se dice cómo; cara al sur) implica que su azimut es sur verdadero.

El programa permite que el usuario indique si el astro ha sido observador cara al norte o cara al sur. No obstante, si se conociese la situación estimada en el momento de la observación y se introdujese, el programa calcularía de forma automática si el astro se observó cara al norte o cara al sur, evitando así que lo hubiese de introducir el usuario.

Para calcular el azimut del astro en el momento de la meridiana con una situación de estima conocida, se sigue el siguiente algoritmo:

a) Si la declinación del astro y la latitud estimada del observador son del mismo nombre se tendrá que:

- El astro se observó cara al norte cuando la declinación sea mayor a la latitud estimada del observador.
- El astro se observó cara al sur cuando la declinación sea menor a la latitud del observador.

b) Sino:

- El astro se observará con azimut igual al nombre de su declinación (d).

Así pues:

Si el astro se observó cara al norte: $\boldsymbol{l = d - z}$

Si el astro se observó cara al sur: $\boldsymbol{l = d + z}$

Dónde: $z = 90º - a$

3.1.2.- Posicionamiento por dos astros:

Con la observación de dos astros podremos obtener una latitud y longitud bastante precisas.

El posicionamiento por dos astros se realiza mediante la técnica de la tangente *Marcq*. La tangente *Marcq* corresponde a la recta de altura que pasa por el determinante que se genera al observar un astro. La recta de altura no es otra cosa que la recta perpendicular al azimut del astro observado y sobre dónde se supone que nos hallamos. La resolución gráfica consiste en trazar desde la situación estimada "*Se"* en el momento de la observación las coordenadas de los dos determinantes (*D1 y D2)* utilizando el azimut "*Z"* como rumbo y el incremento en altura "Δa" como distancia navegada y hacer pasar por cada uno de ellos las rectas de altura correspondientes (*R1 y R2*). La intersección de las dos rectas genera la situación observada (*So*), que es aquella en la que se halla el observador. A continuación se muestra la representación gráfica para encontrar la "*So"*:

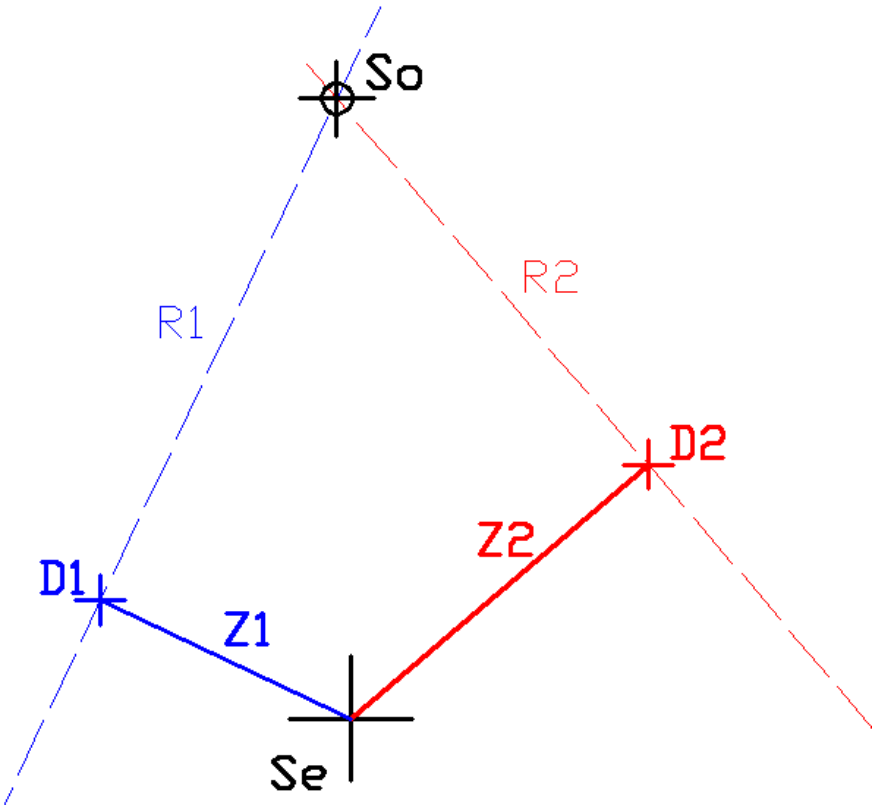


Figura 3.1.

Esto se puede resolver analíticamente encontrando las ecuaciones de las dos rectas de altura (líneas discontinuas) y resolviendo el sistema que generan ambas. Por tanto, la solución serán las coordenadas en el **plano** de la situación observada.

Se destaca la palabra clave "plano" puesto que las coordenadas en el plano no son las coordenadas geográficas. En realidad el anterior dibujo aparentemente lineal no lo es, ya que si bien la separación entre paralelos es constante a lo largo de todo el globo, la separación entre meridianos no, siendo máxima en el ecuador e infinitesimalmente pequeña en los polos. Se puede aproximar diciendo que un arco de paralelo que podemos denominar *"A"* produce una variación de longitud "ΔL" por la siguiente razón geométrica:

$$\Delta L = A * \sec(lm) = \frac{A}{\cos(lm)}; \quad \text{ó} \quad también:$$

$$A = \frac{\Delta L}{\sec(lm)} = \Delta L * \cos(lm)$$

Dónde:

$$lm = latitud\ media = \frac{l_i + l_f}{2}$$

Este arco de paralelo situado a una latitud igual a la media de la latitud de partida y la de llegada, también es llamado en náutica como apartamiento, por ello se simboliza con la letra *A.*

Pues bien como toda recta la podremos expresar mediante la ecuación:

$$y = mx + b$$

Dónde *"m"* es la pendiente de la recta y *"b"* es el intercepto de la recta con el eje *"y",* como se conocerá un punto de la recta y su pendiente, ya podemos definirla. A continuación se demuestra como:

Si tenemos que toda recta viene expresada por: "$y = mx + b$" y conocemos un punto (x_1, y_1) de la recta, también se tendrá la siguiente ecuación:

$$y_1 = mx_1 + b$$

Si restamos la primera ecuación con la segunda ecuación se tendrá que:

$$y - y_1 = m(x - x_1)$$

La anterior ecuación es conocida como la forma punto pendiente de la ecuación de la recta y observamos que en ella el término *"b"* desaparece. Nótese que también puede ser escrita en la forma de:

$$y = mx + (y_1 - mx_1)$$

Lo que indica que el intercepto con el eje *"y"* viene dado por:

$$b = y_1 - mx_1$$

Así pues nuestra recta tendrá la siguiente ecuación:

$$\boldsymbol{y = mx + (y_1 - mx_1)}$$

Dónde la pendiente *"m"* será igual a la tangente del ángulo "$\propto$" formado entre el eje "x" y la recta ó lo que es lo mismo; a la cotangente del rumbo "R" por ser :

$$\propto = 90 - R$$

$$m = tg(\propto) = cotg(R) = \frac{1}{tg(R)} = \frac{\cos(R)}{\sin(R)}$$

Así pues la intersección de ambas rectas se producirá en el punto que resulte del siguiente sistema lineal:

$$\begin{cases} y = m_1 x + (y_1 - mx_1) \\ y = m_2 x + (y_2 - mx_2) \end{cases}$$

Igualando ambas ecuaciones y aislando *"x"* se obtiene la siguiente expresión:

$$\boxed{\boldsymbol{x = \frac{-y_1 + m_1 x_1 + y_2 - m_2 x_2}{m_1 - m_2}}}$$

Para obtener la otra coordenada simplemente se deberá substituir el valor obtenido de "x" en cualquiera de las dos ecuaciones que forman el sistema.

El anterior sistema será compatible determinado siempre y cuando se cumpla la siguiente condición:

$$m_1 \neq m_2$$

De no ser así el sistema será incompatible dando como resultado " $\pm\infty$". En efecto:

$$x = \frac{-y_1 + m_1 x_1 + y_2 - m_2 x_2}{m_1 - m_2} \Rightarrow m_1 = m_2 \Longrightarrow \frac{-y_1 + m_1 x_1 + y_2 - m_2 x_2}{0} = \pm\infty$$

Este resultado era obvio, ya que dos rectas paralelas nunca se cortarán en un punto. Dado que la pendiente de las rectas es la perpendicular a la de sus azimuts, significará que no podremos posiciónanos con dos observaciones en las que haya azimuts iguales u opuestos. Por ello el programa antes de realizar el cálculo comprobará que todos los azimuts de las observaciones sean distintos y de no serlos no nos situará y advertirá del fallo al usuario para que lo corrija.

El valor obtenido de *"y"* será igual a la latitud del observador, pero el valor obtenido de *"x"* será lo que podemos llamar como; "longitud en el plano del observador". Entonces por lo explicado anteriormente y sabiendo que "L_i" y "l_i" son la longitud y latitud estimada iniciales donde se halla el observador, que a su vez son el punto de origen de coordenadas de nuestro sistema cartesiano expresado como (L_i, l_i) la longitud geográfica del observador vendrá dada por la siguiente expresión:

Como $A = \Delta x$, entonces:

$$\Delta L = \Delta x * \sec(lm)$$

$$\Delta x = x - L_i$$

$$\Delta L = L_f - L_i \Longrightarrow L_f = L_i + \Delta L$$

De lo que resulta:

$$\boldsymbol{L_f = L_i + (x - L_i) * \sec(lm) = L_i + \frac{(x - L_i)}{\cos\left(\frac{l_f + l_i}{2}\right)}}$$

Donde:

$$L_i = longitud\ inicial\ estimada\ del\ observador.$$

$$l_i = latitud\ inicial\ estimada\ del\ observador.$$

$$L_f = longitud\ observada\ o\ final\ del\ observador.$$

$$l_f = y = latitud\ observada\ o\ final\ del\ observador.$$

La anterior explicación servirá para hallar la situación observada cuando ambas observaciones de los astros se realicen desde una misma situación estimada. De no ser así simplemente se tomarán como inicio del sistema de coordenadas cartesiana las coordenadas correspondientes a la segunda situación de estima y a esta se le transportará el determinante de la primera observación realizada y a su vez se trazará el determinante correspondiente a la segunda situación estimada, y se trazarán sobre estos las rectas de altura que les correspondan.

El cálculo analítico de la intersección entre las dos rectas de altura se hallará de igual forma a la explicada anteriormente. Es decir, encontrando las ecuaciones de las dos rectas y hallando el punto de su intersección. Una vez encontrado este se pasará a coordenadas geográficas, tal y como se explicó anteriormente, obteniéndose así la situación observada.

3.1.3.- Posicionamiento por tres astros:

El programa permite introducir hasta tres observaciones. El posicionamiento por la observación de tres astros es un ejercicio común en navegación astronómica siendo los astros más observados en estos casos las estrellas y los planetas. No obstante el programa permite que se le introduzca la observación de cualquier astro para posicionarse por este método.

La situación observada *"So"* se calculará mediante el siguiente algoritmo:

1. Dependiendo de si se observan en igual situación de estima o no los tres astros, se hará lo siguiente:
    a. Si las tres observaciones se realizan desde la misma situación estimada; desde esta se calculan los determinantes en el plano de las tres observaciones.
    b. Si alguna de las tres observaciones no se realiza desde igual situación de estima que las otras dos ó todas se realizan desde distintas situaciones estimadas; se trazarán los tres determinantes en el plano desde la situación estimada correspondiente a la última observación.
2. Por cada determinante se hace pasar la recta de altura correspondiente a su observación.
3. Si las tres rectas de altura se cortan en un único punto; este se pasará a coordenadas geográficas obteniéndose así la situación observada.
4. Si las tres rectas de altura no se cortan en un único punto; significa que se cortarán dos a dos en tres puntos distintos y estos formarán entre sí un triángulo.
5. Se utiliza el sistema de las bisectrices de los azimuts para posicionar al observador, debido a que gracias a este se corrigen los errores sistemáticos. Se buscan las bisectrices de los azimuts que le correspondan a cada lado del triángulo, o dicho de otro modo, se buscan las bisectrices de los ángulos generados entre los azimuts correspondientes a las dos rectas de altura que han originado cada uno de los tres vértices del triángulo.
6. Las bisectrices de los azimuts se cortarán en un único punto y las coordenadas de este, pasadas a geográficas, darán la situación observada.

La resolución gráfica del anterior algoritmo tendría la siguiente forma:

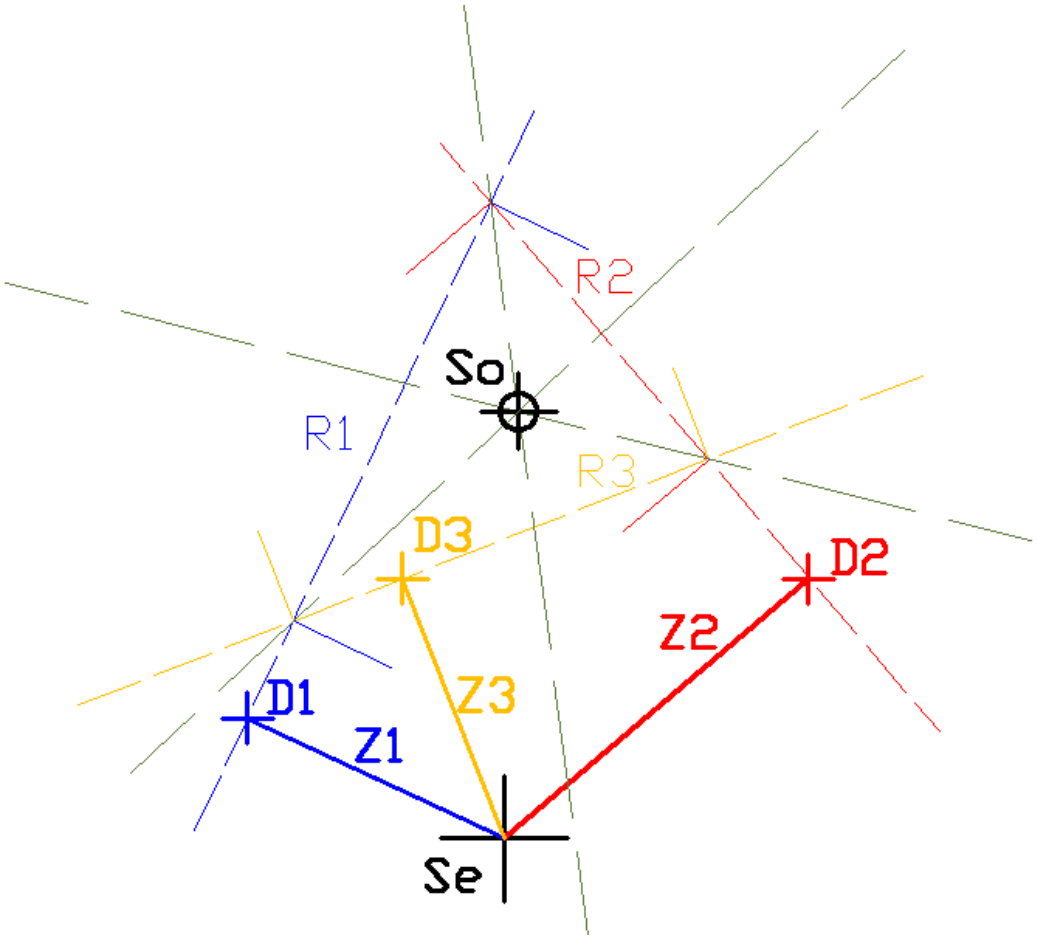


Figura 3.2.

Para hallar la situación observada solamente deberemos hallar las ecuaciones de las tres rectas de altura (R1, R2 y R3) calcular el punto de intersección de cada una con la otra, para obtener las coordenadas de los tres vértices del triángulo. Y una vez hecho esto, se ha de mirar que estos tres vértices en realidad no sean el mismo punto, ya que de ser así esta sería la situación observada en el plano y se acabaría el problema. Si no fuesen el mismo punto, se debería hallar la ecuación de las rectas bisectrices y calcular la intersección entre tan solo dos de las tres bisectrices de los azimuts que se generan (ya que al añadir la tercera se obtendría el mismo resultado o punto de intersección que con las otras dos) y este pasarlo a coordenadas geográficas obteniendo así la situación observada.

A continuación se explica cómo hallar la ecuación de una recta bisectriz de dos azimuts cualquieras (Z_A, Z_B) que pasa por un vértice del triángulo:

Su ecuación tendrá la siguiente forma:

$$\boldsymbol{y = mx + (y_{ab} - mx_{ab})}$$

Y solo deberemos tener en cuenta lo siguiente:

$$m = cotg(\beta) = \frac{\cos(\beta)}{\sin(\beta)}$$

$$\beta = \frac{Z_A + Z_B}{2}$$

$(x_{ab}, y_{ab}) =$ son las coordenadas en el plano del punto que corresponde al vértice del triángulo por el que pasa esa bisectriz, encontradas previamente por la intersección de las dos rectas de altura que lo generan, mediante el procedimiento explicado en el apartado anterior.

3.2.- Coordenadas Solares:

Las coordenadas del Sol se calculan mediante la teoría VSOP87 completa, esta consiste, en el uso de miles de términos periódicos para posicionar, en este caso, la Tierra con respecto el Sol.

El uso de esta teoría es complicado y largo y se explica con mayor detalle en el apartado de; **Coordenadas de los Planetas**. Por lo que aquello que aquí no se explique, se deberá consultar en ese apartado. Para el caso de la Tierra, se utilizarán la siguiente cantidad de términos periódicos:

Descripción de los términos:	***Número de términos:***
Términos para L (longitud celeste geocéntrica):	*1080*
Términos para B (latitud celeste geocéntrica):	*348*
Términos para R (distancia de separación Tierra-Sol):	*997*
Total:	***2425***

Como es lógico en este trabajo no se van a mostrar todos los términos periódicos usados para la Tierra, ya que su extensión es inmensa. Pero, para ver el modelo/formato que estos tienen, a fin, de saber cómo se deben de usar dentro del algoritmo de cálculo, el lector deberá de consultar el apartado de **Coordenadas de los Planetas** donde, como ya se ha dicho, se explica todo esto con mayor detalle.

El cálculo de las coordenadas del Sol puede realizarse del siguiente modo:

1. Calcularemos el JDE para el momento dado.

2. Calcularemos la centuria juliana (T) del siguiente modo:

$$T = \frac{JDE - 2451545}{36525}$$

3. Realizamos el sumatorio de los términos periódicos de la Tierra, obteniendo el valor de $L, B\ y\ R$ para el momento dado (ver apartado de **Coordenadas de los Planetas**).
4. Calculamos el valor de la nutación en longitud $(\Delta\psi)$ y oblicua $(\Delta\varepsilon)$, además del valor que tenga en ese momento la oblicuidad de la eclíptica (ε). Tal y como se explicó en apartados anteriores.

5. Obtenemos las coordenadas eclípticas geocéntricas del Sol para el momento dado, de la siguiente forma:

$$\lambda' = L + 180 - 1.397 * T - 0.00031 * T^2 - \frac{0.09033}{3600}\ [º]$$

$$\boxed{\beta = -B + \frac{0.03916 * (\cos(\lambda') - \sin(\lambda'))}{3600}\ [º]}$$

Y la longitud aparente del Sol como:

$$\boxed{\lambda = L + 180 - \frac{0.09033}{3600} + \Delta\psi - \frac{20.4898}{3600 * R}\ [º]}$$

6. Una vez calculadas las coordenadas eclípticas geocéntricas del Sol, se pueden calcular, a partir de estas; las coordenadas ecuatoriales y horizontales del Sol, tal y como se explicó en apartados anteriores.

3.2.1.- Semidiámetro solar:

Para calcular el semidiámetro (s) y el diámetro aparente del Sol (sp), operaremos de la siguiente forma:

$$s = \frac{959",63}{R}\ ["]$$
$$sp = s * 2\ ["]$$

Donde;

$R\ =$ distancia que separa la Tierra del Sol para el momento dado y expresada siempre en unidades astronómicas.

3.2.2.- Paralaje solar:

El paralaje horizontal ecuatorial (π) del Sol pese a ser un valor muy pequeño, del orden de los *9",* se puede calcular para el momento dado y con una muy buena precisión, del siguiente modo:

$$\sin(\pi) = \frac{\sin(8".794)}{R}$$

Donde;

$R =$ Distancia de separación entre la Tierra y el Sol para el momento dado (ver apartados anteriores).

3.3.- Coordenadas de la Luna:

En este apartado se describirá el algoritmo que utiliza el programa para hallar las coordenadas de la Luna en cualquier momento. Se utilizará un método que aporta una alta precisión, en concreto, el error máximo es de unos 10" en longitud y unos 4" en latitud. Dicho error es ínfimo para la precisión que requiere este programa.

El programa será capaz de calcular los siguientes datos de la Luna:

1. La Ascensión Recta $(\propto)$.
2. La Declinación (δ).
3. El Horario del Lugar (hL).
4. La distancia de separación entre la Luna y la Tierra para el momento dado (Δ).
5. La longitud celeste o longitud eclíptica geocéntrica (λ).
6. La latitud celeste o latitud eclíptica geocéntrica (β).
7. El Azimut (A).
8. La Altura (h).
9. El paralaje ecuatorial horizontal (π).
10. El semidiámetro (*s*) y el diámetro aparente (*sp*).
11. La fracción iluminada del disco lunar (*k*).
12. El ángulo de fase (*i*).
13. La elongación (ψ).
14. La fase lunar en que se encuentre.
15. La hora de paso por el meridiano de Greenwich para la fecha dada (*PmG*), es decir, la hora de culminación de la Luna.

El algoritmo detallado para el cálculo de las coordenadas y efemérides lunares que el programa implementa es el ***ELP2000*** y por ser bastante extenso se explica en el anexo del presente trabajo.

3.4.- Coordenadas de los Planetas:

Para hallar las coordenadas de los planetas des de Mercurio hasta Neptuno (incluida la Tierra) se utiliza la teoría $VSOP87$ que viene del francés; *"Variations Séculaires des Orbites Planétaires".* Esta teoría consiste en una larga colección de términos periódicos para cada

planeta mediante los cuales se describe su movimiento. La agrupación de términos se hace para cada planeta en función de las 3 series siguientes:

L; la longitud eclíptica del planeta.

B; la latitud eclíptica del planeta.

R; la distancia de separación del planeta al Sol.

Cada una de las tres agrupaciones de términos periódicos se dividen en subgrupos, así pues podemos tener para un determinado planeta; $L0,\ L1,\ L2 \ldots, B0, B1, \ldots, R0, R1$ … por ejemplo.

El número de términos periódicos utilizados para cada planeta es muy elevado (del orden de los millares) por lo que no se van a mostrar (ya que de hacer-lo su extensión seria excesiva). Sin embargo, lo que sí que se muestra es un resumen del número de términos a usar para cada planeta, así como la forma en que se deben de utilizar dentro del algoritmo, tal y como aparece en la siguiente tabla:

Nombre del Planeta	***Número de términos por serie.***			***Total del Planeta***
	L	***B***	***R***	
Mercurio	*2808*	*1620*	*2399*	***6827***
Venus	*671*	*426*	*585*	***1682***
Tierra	*1080*	*348*	*997*	***2425***
Marte	*2393*	*915*	*2175*	***5483***
Júpiter	*1484*	*530*	*1469*	***3483***
Saturno	*2358*	*966*	*2435*	***5759***
Urano	*1578*	*515*	*1896*	***3989***
Neptuno	*681*	*290*	*958*	***1929***

Cada serie está dividida en "*n*" sub-series, así por ejemplo; la serie; $\boldsymbol{L}$, contendrá las sub-series; $\boldsymbol{L_0},\ \boldsymbol{L_1},\ \boldsymbol{L_2}, \ldots, \boldsymbol{L_n}$. Y cada una de estas sub-series contendrá un número variable de agrupaciones de 3 términos, que corresponden a tres coeficientes denominados como; $\boldsymbol{A},\ \boldsymbol{B}\ \boldsymbol{y}\ \boldsymbol{C}$. Lo podemos ver mejor con la siguiente tabla (a modo de plantilla) que muestra algunos de los términos periódicos usados para *Urano*;

Para la serie L:			
Sub-serie L_0:	A	B	C
	5.48129294	0	0
	0.09260408	0.89106422	74.7815986
	0.01504248	3.62719262	1.48447271
	0.00365982	1.89962189	73.2971259
	0.00272328	3.35823711	149.563197
	0.00070328	5.39254432	63.7358983
	0.00068893	6.09292489	76.2660713
	0.00061999	2.2695204	2.96894542

	0.00061951	2.85098908	11.0457003
	0.00026469	3.14152088	71.8126532
	0.00025711	6.11379843	454.909367
	. . .	. . .	. . .
Sub-serie L_1:	A	B	C
	75.0254312	0	0
	0.00154458	5.24201658	74.7815986
	0.00024456	1.71255705	1.48447271
	9.2578E-05	0.42844639	11.0457003
	75.0254312	0	0
	0.00154458	5.24201658	74.7815986
. . .	. . .	. . .	. . .
Para la serie B:			
Sub-serie B_0:	A	B	C
	0.01346278	2.61877811	74.7815986
	0.00062341	5.08111176	149.563197
	0.00061601	3.14159265	0
. . .	. . .	. . .	. . .
Para la serie R:			
Sub-serie R_0:	A	B	C
	19.2126485	0	0
	0.88784984	5.60377527	74.7815986
	0.03440836	0.32836099	73.2971259
	0.02055653	1.7829517	149.563197
	0.00649322	4.52247298	76.2660713
	0.00602248	3.8600382	63.7358983
	0.00496404	1.40139935	454.909367
	. . .	. . .	. . .
Sub-serie R_1:	A	B	C

	0.01479896	3.67205705	74.7815986
	0.00071212	6.22601007	63.7358983
	0.00068627	6.13411265	149.563197
	0.00020857	5.24625494	11.0457003
	0.01479896	3.67205705	74.7815986
	. . .	. . .	. . .
Sub-serie R_2:	A	B	C
	0.0002244	0.69953119	74.7815986
	4.727E-05	1.69901641	63.7358983
	1.6819E-05	4.64833552	70.8494453
	1.4338E-05	3.52119918	149.563197
. . .	. . .	. . .	. . .

Por ejemplo para calcular el valor total de la serie L deberíamos de hacer lo siguiente;

$$L = L_0 + L_1 * \tau + L_2 * \tau^2 + L_3 * \tau^3 + \cdots + L_n * \tau^n$$

Donde "τ" corresponde al milenio juliano en que nos encontremos, es decir;

$$\tau = \frac{JDE - 2451545}{365250}, o\ bien;\ \tau = \frac{T}{10}$$

Cada sub-serie de $L\ (L0,\ L1, \ldots, L_n)$ se calcula como el sumatorio de todos los términos periódicos que compongan dicha serie, puestos de la siguiente forma;

$$A * \cos(B + C * \tau)$$

Así pues, para el caso de $L0$ y suponiendo que esta serie tenga m_0 términos, su valor vendría dado tras realizar la siguiente operación;

$$L0 = \sum_{i=1}^{m_0} A_i * \cos(B_i + C_i * \tau)$$

Generalizando obtenemos la expresión siguiente;

$$L = \left[\sum_{i=1}^{m_0} A_i * \cos(B_i + C_i * \tau)\right] *\ \tau^0 + \left[\sum_{i=1}^{m_1} A_i * \cos(B_i + C_i * \tau)\right] * \tau^1 + \left[\sum_{i=1}^{m_2} A_i * \cos(B_i + C_i * \tau)\right] * \tau^2 + \cdots + \left[\sum_{i=1}^{m_n} A_i * \cos(B_i + C_i * \tau)\right] * \tau^n$$

Se operará **de la misma forma que hemos hecho en L**, para calcular el valor de $B\ y\ R$, solo que en vez de utilizar las sub-series referidas a L se deberán utilizar las sub-series de B ó de R según la serie, de entre las dos, que se quiera calcular.

3.5.- Coordenadas de las Estrellas:

El programa manejará hasta 9110 estrellas distintas de las que puede conocer sus coordenadas; horizontales, ecuatoriales y eclípticas geocéntricas, además de otros datos de carácter astronómico. Para ello utiliza el catálogo estelar; *"Bright Star Catalogue"* en su quinta revisión. Dicho catálogo es uno de los cinco más utilizados en el mundo y uno de los que aporta una mayor información para las estrellas más importantes conocidas del universo. En concreto muestra todas las estrellas conocidas hasta la magnitud +6,5 y parte de las más importantes entre las magnitudes de; +6,5 hasta +7,96. Como se ha dicho es un catálogo muy completo y utilizado en todo el mundo de la astronomía por lo que frecuentemente podremos ver referencias hacia él en textos especializados de la NASA y otros documentos técnicos sobre astronomía. Por todo ello se ha decido usar este catálogo estelar, y no otro, para implementar el programa.

¿Qué es un catálogo estelar?

Un catálogo estelar es un documento en el que figuran para cada estrella todos aquellos datos que se consideren de interés sobre ella. Su estructura nos la podemos imaginar como una tabla o como una base de datos. Cada estrella es un registro que se compone de *"n"* campos los cuales deben ser debidamente cumplimentados para esta. Así pues tendremos que para cada estrella hay informaciones de diversos tipos como son desde; las distintas formas de nombrarlas, hasta la magnitud visual, espectro electromagnético y movimiento propio que esta tenga, entre otros. Así pues, por ejemplo, para el catálogo usado en SPAC (el *"Bright Star Catalogue"*) tendremos que para cada estrella hay, principalmente, la siguiente información;

- Identificaciones incluidas en otros catálogos.
- Identificaciones de estrella doble o múltiple.
- Indicaciones de variabilidad y designadores de variable.
- Posiciones ecuatoriales para los equinoccios de referencia; B1900.0 y J2000.0
- Coordenadas galácticas.
- Datos fotoeléctricos fotométricos UBVRI (si los hay para esa estrella).
- Tipo espectral de acuerdo al sistema de clasificación MKK.
- Movimiento propio (J2000.0).
- Paralaje (PHE).
- Datos de velocidad radial y de velocidad de rotación.
- Información sobre estrellas múltiples (número de componentes, separación y diferencia de magnitud entre ellas).

El algoritmo de cálculo de las coordenadas de las estrellas es una tarea bastante larga y compleja, por lo que su explicación detallada se desarrolla en el anexo del presente trabajo.

4.- Estructura de los datos:

La base de datos necesaria para este proyecto consiste únicamente en los datos facilitados por el catálogo estelar, es decir, en aquellos valores constantes los cuales tomen cada uno de los campos de cada estrella y los cuales se utilizan como valores constantes para realizar el cálculo de la generación de coordenadas de una estrella. Un catálogo estelar no es más que una tabla dónde cada estrella (registro) contiene la información necesaria para cada campo del catálogo (campo de la base de datos). El catálogo estelar se modela con la tabla siguiente:

CatalogoEstelar(HR, name, dm, hd, sao, fk5, irflag, multiple, ads, adscomp, varID, RAJ2000, DEJ2000, GLON, GLAT, Vmag, n_Vmag, u_Vmag, B-V, u_B-V, R-I, SpType, n_SpType, pmRA, pmDE, n_Parallax, Parallax, RadVel, n_RadVel, l_RotVel, RotVel, u_RotVel, Dmag, Sep, MultID, MultCnt, NoteFlag)

Es decir, SPAC solo necesitará consultar para la estrella demandada por el usuario cada uno de los valores asociados a su registro dentro de la tabla. Dichos valores son constantes que podrían incluso haberse declarado como tal dentro del propio código del programa, pero por manejar cerca de 10000 estrellas distintas no se ha optado por esta solución ya que aumentaría notablemente el espacio de memoria que el programa necesitaría tener reservado cuando se ejecutase. Así pues se ha hecho el siguiente enfoque teniendo en cuenta estos dos aspectos claves:

- Por un lado la simplicidad de los datos a almacenar y la inamovilidad de los mismos, es decir, los datos almacenados son en realidad constantes por lo que solo necesitarán ser consultadas por el programa en el momento de realizar los cálculos de coordenadas de una estrella, por lo que no podrán ni variar ni se ser invalidadas por lo que no existe mayor gestión en los mismos que la inserción inicial de dichos datos dentro de la tabla y la posterior consulta.
- Por otro lado, hay que tener en cuenta que esta fuente de información debe de poderse incorporar de forma íntegra y con relativa facilidad en el programa realizado para Windows Mobile y que por no ser este un tipo de programa cliente servidor y al utilizarse únicamente para el soporte de la generación de coordenadas, este deberá integrar dicha base de datos estelar de la forma más simple posible y con el menor requisito de software adicional y dependencias posibles.

Por lo que se ha decidido realizar una base de datos rudimentaria creada des de cero y gestionada a través del propio programa mediante archivos de datos donde se estructurará la información de la tabla *CatalogoEstelar*. Dichos archivos de datos son generados con formato XML mediante el serializador XML facilitado por .NET; **XMLSerializer**.

Las 9110 estrellas que componen el catálogo estelar estarán contenidas en dicha tabla y esta a su vez será almacenada en un archivo de datos serializado en XML dentro de la carpeta de la aplicación.

Dicho archivo de datos, se divide en 100 partes, es decir en realidad habrá 100 archivos de datos y no solo uno, en los que su clave primaria será el campo "*HR"* y lo que se hará será ordenar las 9110 estrellas según su HR dentro de la tabla y se particionará en 100 partes contiguas ordenadas del archivo de la parte 0 a la parte 99. Cada archivo se denominará como ***data<numero_parte>.xml,*** así habrán los archivo data0.xml, data1.xml, …, data99.xml. Cuando el usuario quiera buscar la estrella HR=12 el programa aplicará la función *hash* que le devuelva el número de archivo *xml* (para el caso de HR=12 se encontraría en *data0.xml*) que contiene dicha estrella y lo leerá encontrando así los datos asociados a la estrella que el usuario buscaba. La finalidad de esta estructura es la de no tener que leer todo el catálogo estelar cuando el usuario quiera calcular las coordenadas de una estrella sino solo cargar la parte del archivo del catálogo estelar que contiene esa estrella ahorrando memoria y tiempo de procesador.

¿Qué pasaría si el usuario quisiera buscar una estrella por cualquiera de sus otros siete nombres posibles (si es que esa estrella admite tantas formas de denominarla)? Para ese caso el programa también genera siete archivos xml de datos de respaldo a la base de datos del catálogo estelar en los que se encuentra para ese otro nombre una clave forana al nombre HR (clave primaria) asociado para esa estrella, de modo que una vez conocido el nombre HR de la estrella buscada, el programa ya sabrá en que archivo xml de la base de datos debe consultar, de entre los 100 que tenemos que componen la base de datos del catálogo estelar. Dichos archivos de soporte de índices son;

- in_DM.xml → Contiene los índices para las estrellas que admitan también denominación *Durchmusterung Identification.*
- in_FK5.xml → Contiene los índices para las estrellas que admitan también denominación de catálogo *FK5.*
- in_HD.xml → Contiene los índices para las estrellas que admitan también denominación *Henry Draper Catalog Number*.
- in_Name.xml → Contiene los índices para las estrellas que admitan también denominación *Bayer i/o Flamsteed*.
- in_quick.xml → Contiene los índices para las estrellas que tengan nombre propio en latín o griego (tan solo habrá unas 200 estrellas).
- in_SAO.xml → Contiene los índices para las estrellas que admitan también denominación con número de catálogo SAO.
- in_VarID.xml → Contiene los índices de todos aquellos posibles alías adicionales, que sin pertenecer a ningún registro ni catálogo estelar oficial, pueden ser utilizados para referirse a una estrella.

Con este modelo se consigue que el propio programa sea el gestor de su base de datos. Generando así una solución óptima, simple y fácil de portar a otras plataformas como la móvil, gracias a no depender de ningún otro servicio de gestión de base de datos.

La interfaz que el programa tiene para leer los datos de una estrella es la anteriormente presentada interfaz; **IBuscadorEstrella.**

4.1.- Guardar y cargar cálculos astronómicos:

Los cálculos astronómicos que realice el usuario serán guardados como archivos de extensión *".ger"* estos archivos serán guardados como una serialización XML del objeto de la clase *SituaciónObservada* que los contenga mediante el uso de la clase .NET; System.Xml.Serialization.XmlSerializer. Se recuerda que *SituaciónObservada* es un contenedor de observaciones y que en él residen todos los datos que el usuario ha introducido en la aplicación para posicionarse.

El programa permitirá un explorador de archivos dónde el usuario pueda indicarle la ruta en la que desea guardar su cálculo, o bien, la ruta y archivo dónde se halla el cálculo astronómico que desea abrir.

4.2.- Guardar y cargar preferencias:

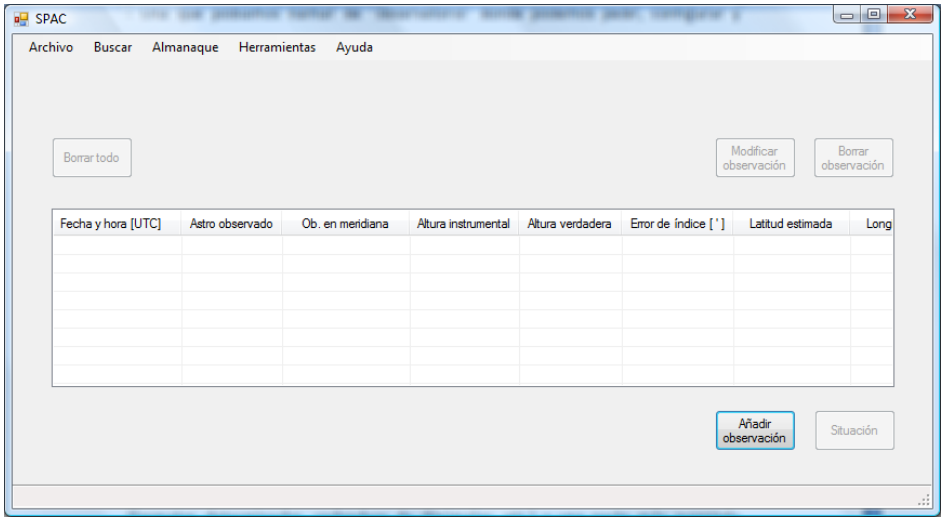
Las preferencias de la aplicación se guardaran en el directorio de usuario de Windows para la aplicación principal de SPAC y para la aplicación móvil se guardarán en el mismo directorio de la aplicación bajo el nombre de archivo *pref.xml* ya que Windows Mobile es un sistema operativa mono-usuario. También se guardarán como una serialización en XML de la clase *Preferencias* de SPAC y SpacMobile. La carga y modificación de las preferencias es totalmente transparente al usuario, siendo el programa quien realiza esta tarea de forma automática.

4.3.- Guardar y cargar estrellas favoritas:

En la clase Preferencias tendremos una lista con aquellas estrellas más utilizadas por el usuario, dicha lista se actualizará cada vez que el usuario seleccione una estrella, incrementando así el contador del número de veces que el usuario ha elegido esa estrella y se actualizará la lista de estrellas favoritas en *Preferencias*. Así pues la lista con las estrellas favoritas del usuario (aquellas que el usuario ha utilizado más veces para posicionarse) se serializará como un atributo más dentro de las preferencias del usuario mediante XML. Y por consiguiente cada usuario de Windows tendrá su propia lista de estrellas favoritas, es decir, aquellas que más haya utilizado. Por lo tanto cada vez que se modifique la lista de estrellas favoritas del usuario deberán serializarse de nuevo las preferencias dentro del directorio de usuario correspondiente, a fin, de que los cambios en esta lista se reflejen en el archivo de preferencias del usuario.

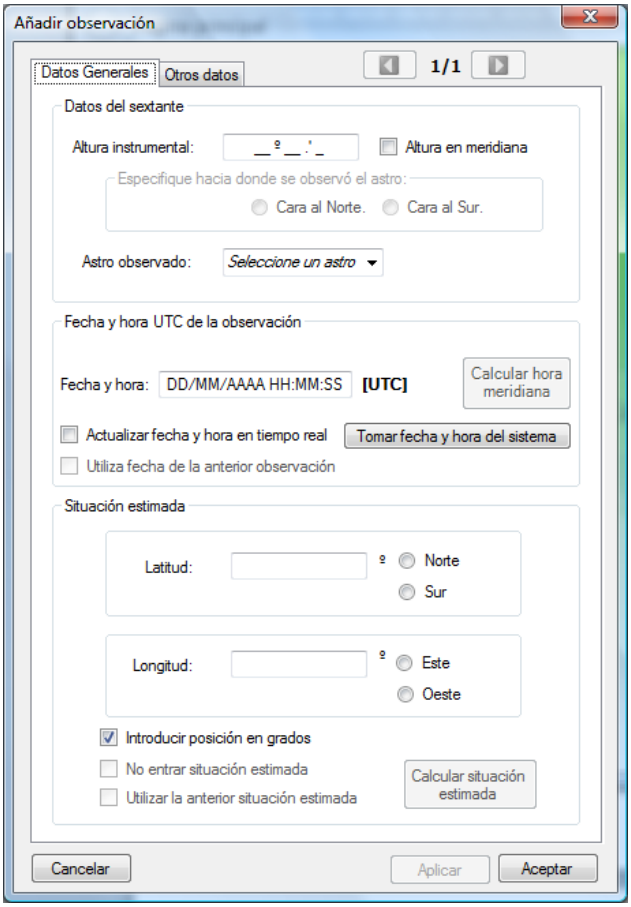
5.- Diseño visual:

Se ha realizado bajo *Windows Forms* un diseño preliminar sobre el que partir de base para la siguiente fase de implementación. Como formulario principal del programa tendríamos el siguiente;



La idea es tener un *list view* donde podamos entrar cada una de nuestras observaciones realizadas, de modo que cuando tengamos suficientes como para situarnos, el programa activaría el botón "Situación" (aquí mostrado como; *no enable*) a fin de informar al usuario de que con las observaciones entradas ya es capaz de calcular su situación geográfica.

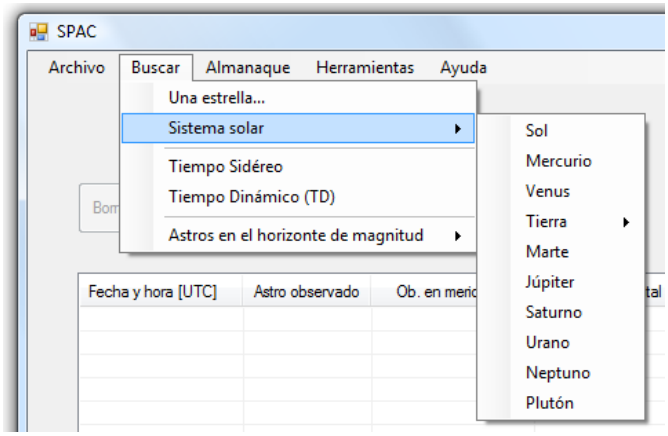
Si diéramos clic en "Añadir observación" se presentaría el formulario de *Añadir observación*. Aquí indicaríamos todo lo necesario a saber, para la observación que hallamos realizado del astro. Podemos, además, en la casilla "Otros datos" variar los datos de sextante, presión atmosférica, temperatura y elevación del nivel del mar, con los que el programa trabaja por defecto (faltaría diseñar este formulario).



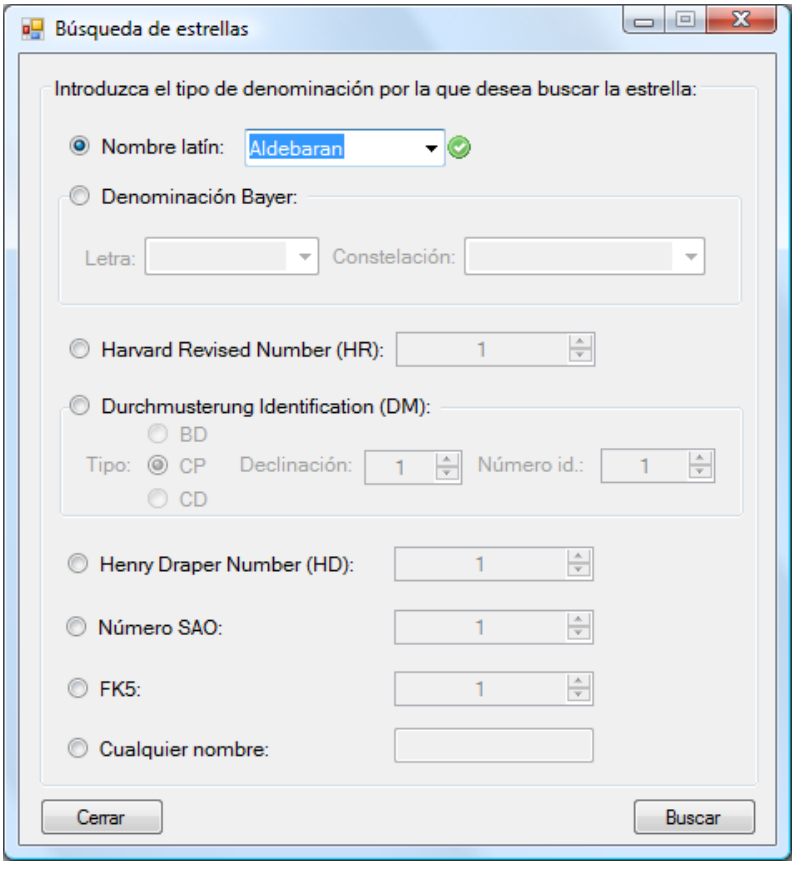
Una vez que demos clic en "Aceptar" se nos añadiría la observación a la lista de observaciones (*list view*) del formulario principal.

5.1- Diseño del menú "Buscar":

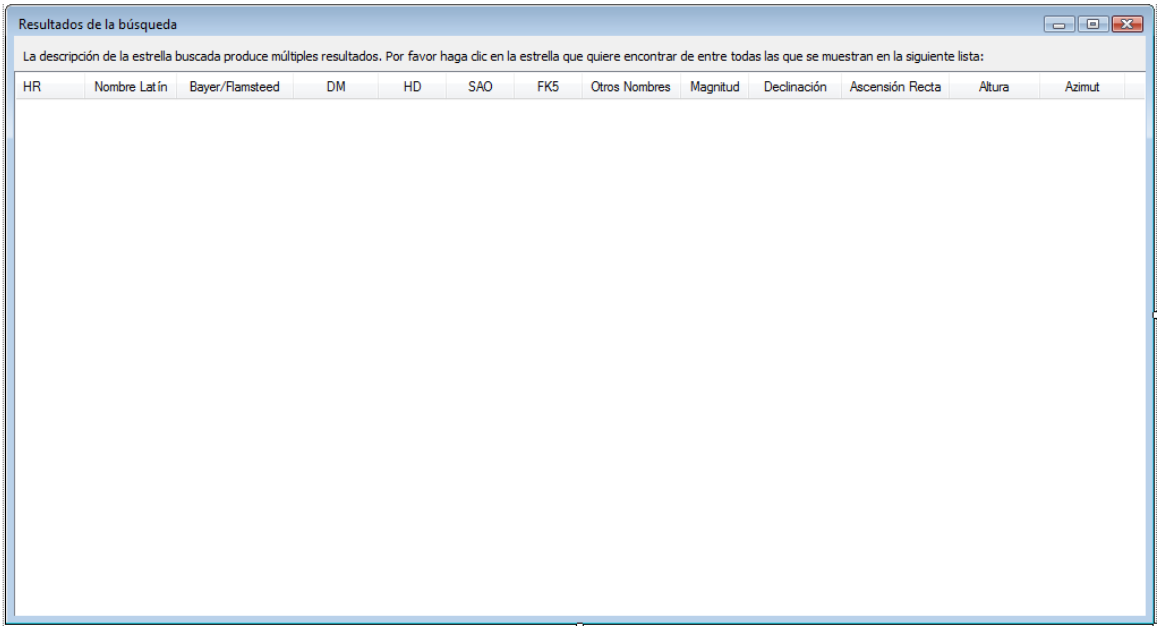
En este menú podremos encontrar todas las principales prestaciones que SPAC nos ofrece para la parte del "observatorio". Es decir, podremos ver el cielo visible en un momento dado, buscar determinados astros, ver la información en tiempo real de los planetas, de la Luna, etc. Más concretamente el menú tiene las siguientes opciones;



Como vemos podremos rápidamente pedirle la información de cualquier planeta del sistema solar, además de la Luna y el Sol. A falta de implementar las funciones de cálculo astronómico para cada astro no se pueden mostrar los formularios asociados cada uno de ellos con la información para cada astro, pero a groso modo sería un formulario donde aparecen todas las informaciones relativas al astro seleccionado. En la opción *Busca>Una estrella…* tendríamos la capacidad de buscar de entre todas las estrellas que manejaría el programa aquella/s que nos interesa;



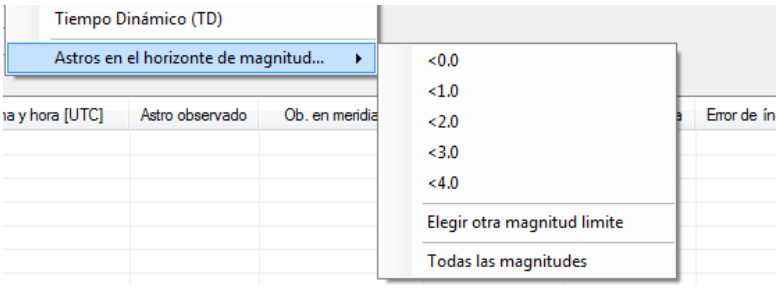
Si la búsqueda produce más de un resultado entonces se mostraría otro formulario dónde se tabularán todas aquellas estrellas que respondan al nombre/denominación por la que las hemos buscado, el formulario se compone en su esencia de un control *list view*, como sigue;



El usuario seleccionaría la estrella que buscaba de entre las que se muestren y entonces le aparecerían los datos de esa estrella. El formulario de visualización de los datos de un astro siempre seria el mismo, sea ese astro que se, es decir; una estrella, un planeta o una luna.

Volviendo a la pantalla principal y en Buscar también tendríamos las opciones Tiempo Sidéreo y Tiempo Dinámico, mostrarían un *DialogBox* que indique la fecha en cada una de estas dos formas para la fecha que tenga el usuario por defecto y sobre la que toma como base el programa para hacer los cálculos (véase más adelante en el formulario de preferencias).

La opción “Astros en el horizonte de magnitud” permitiría mostrar-le al usuario, para su situación geográfica, previamente configurada en las preferencias del programa, aquellos astros hasta la magnitud visual elegida por el usuario dentro del desplegable que fuesen visible (es decir; tengan altura positiva, o lo que es lo mismo; estén sobre su horizonte), como se ve en la imagen le estas opciones tendrían el siguiente aspecto;

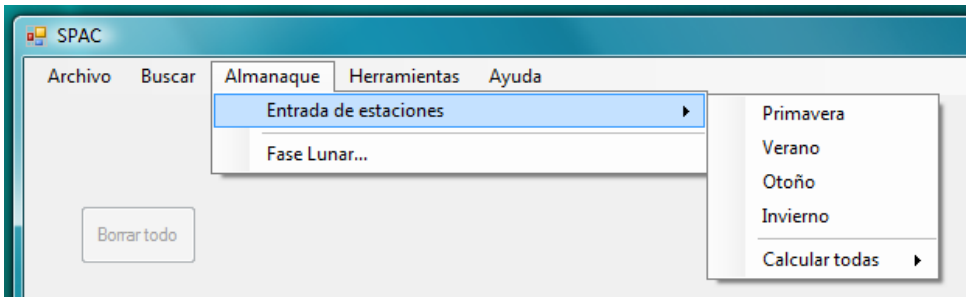


5.2.- Diseño del menú "Almanaque:

Este menú pretende implementar una muestra de las prestaciones que pueden derivarse a nivel de predicción y establecimiento de las efemérides haciendo uso de los datos astronómicos calculados por el programa. De momento se permiten dos opciones; la de calcular la entrada de las estaciones y la de calcular la fase lunar. De todos modos, opciones

tan comunes como podrían ser; orto y ocaso de los astros, efemérides de los planetas o eclipses, se podrían llegar a desarrollar.

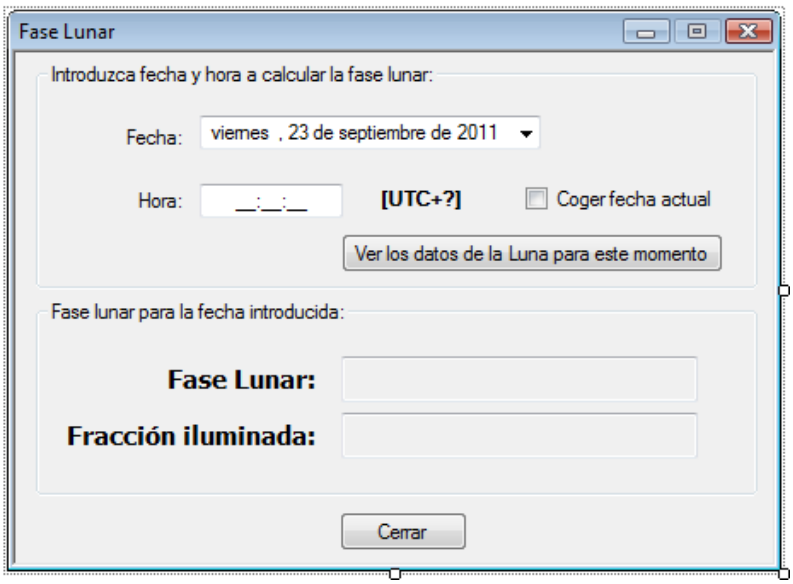
Si dentro del menú; "Almanaque" seleccionamos; "Entrada de estaciones" se nos abrirá un desplegable que tendría el siguiente aspecto;



Si seleccionamos una estación entonces se nos dirá cuándo entrará esta a partir de la fecha que tenga configurada el usuario en el programa. Por ejemplo si queremos saber cuándo entra la primavera daríamos clic en "Primavera" y se nos mostraría un cuadro de diálogo donde aparecería la fecha para la entrada de la primavera.

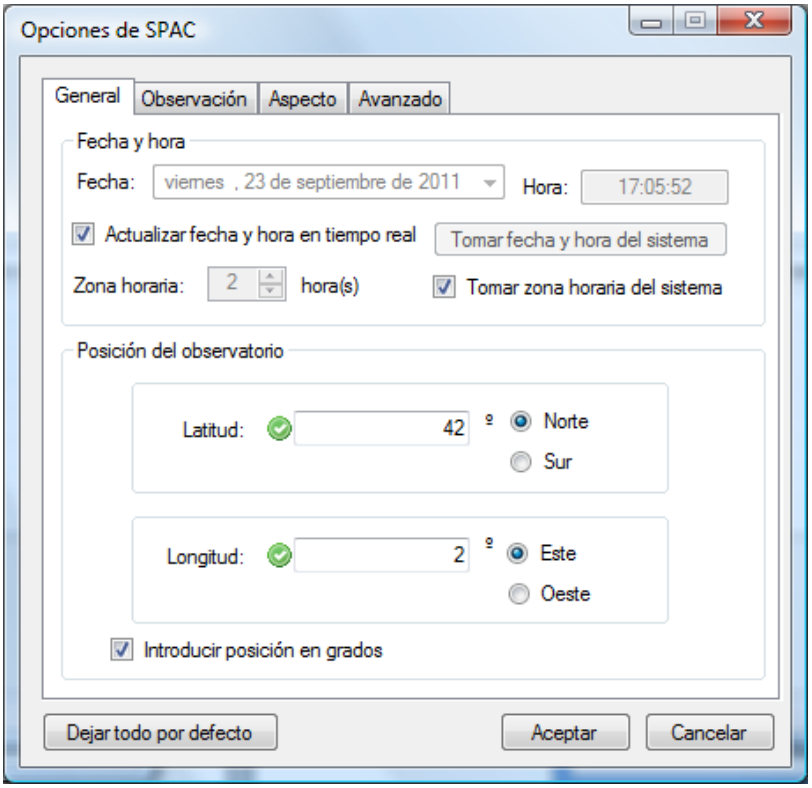
La búsqueda de la entrada de las estaciones se realizará a partir de una búsqueda dicotómica sobre la longitud celeste del Sol (en los algoritmos que se utilicen para el cálculo de coordenadas del Sol ya se explicarían todos estos conceptos de astronomía para poder comprender que se está haciendo en cada momento), de modo que, cuando esta sea múltiple de $\pi/2$ radianes, entrará la siguiente estación del año.

Por último la opción de menú de; Fase Lunar… permitiría ver la fase Lunar (y fracción de disco lunar visible, en porcentaje) en la fecha y hora que nos hallamos. Tendría un aspecto similar a lo siguiente;



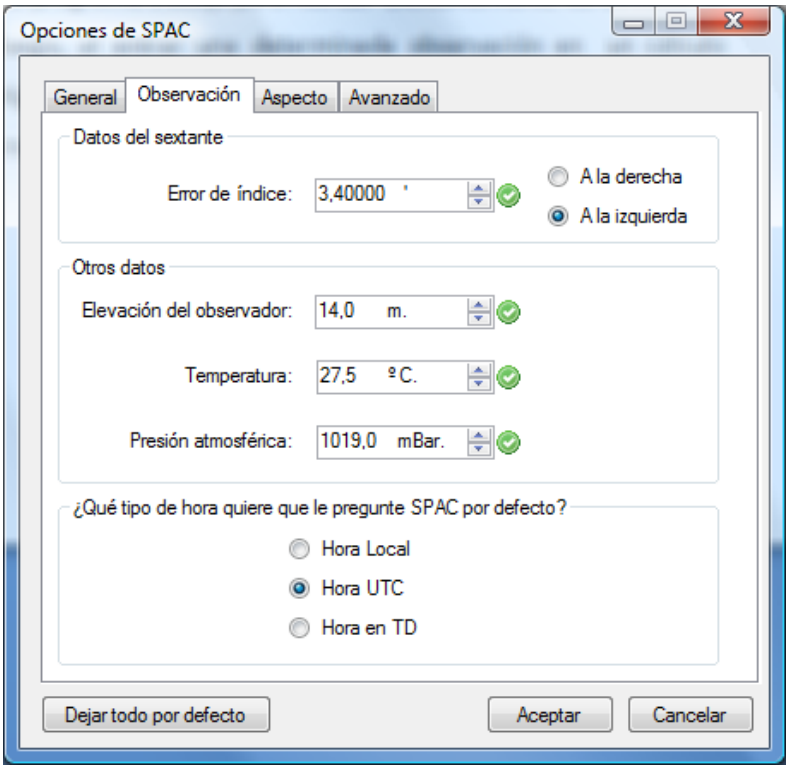
5.3.- Diseño del menú “Herramientas”:

En; *Herramientas>Opciones de SPAC...* podemos configurar el entorno de trabajo de SPAC. La ventana de opciones de SPAC tiene el siguiente aspecto;

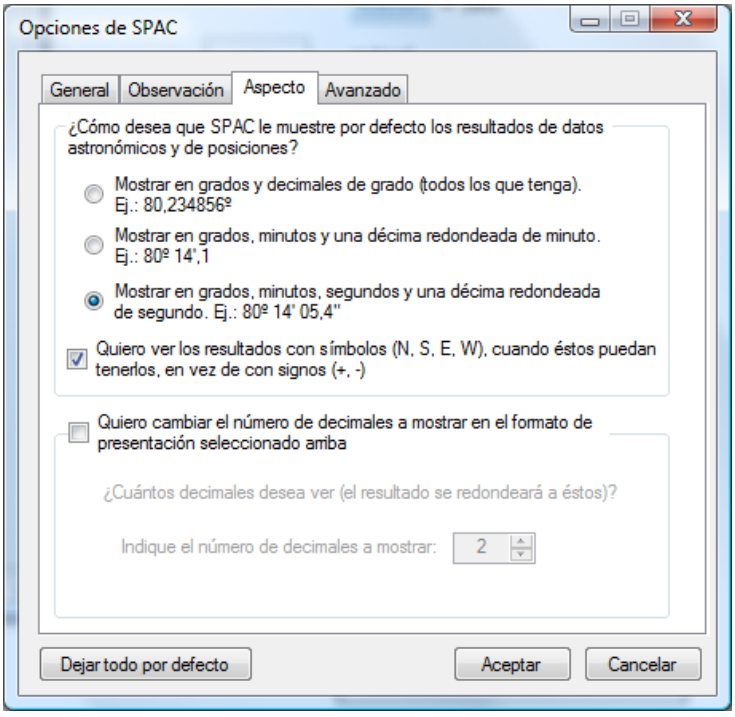


En esta primera pestana de *"General"*, podemos establecer la fecha y hora (o decidir vincularla a la del sistema, de modo que siempre tome como referencia esta), la zona horaria (también se puede vincular al uso horario del sistema) y la latitud y longitud del observador, con la que queremos que trabaje el programa para la parte de "Observatorio", es decir aquella con la que el programa nos calculará la posición de los astros cuando le pidamos datos de los mismos fuera de los cálculos astronómicos de posicionamiento (los *ticks* en verde, se han programado de forma inicial para que aparezcan con este aspecto cuando el usuario valida correctamente los campos, si por ejemplo el usuario pusiera una latitud mayor a 90º ó una longitud mayor a 180º el control .NET *ErrorProvider* mostraría dos aspas cruzadas en rojo al lado del control mal validado).

Después en la pestaña *"Observación"* podríamos establecer los datos por defecto con los que queremos que el programa trabaje para realizar exclusivamente los cálculos de posicionamiento. Si, luego, al entrar una determinada observación en un cálculo quisiéramos cambiar alguno de estos campos dejados como de defecto para todo el programa, también tendríamos la opción de hacerlo, tal y como veremos más adelante. El aspecto de esta pestaña sería el siguiente;

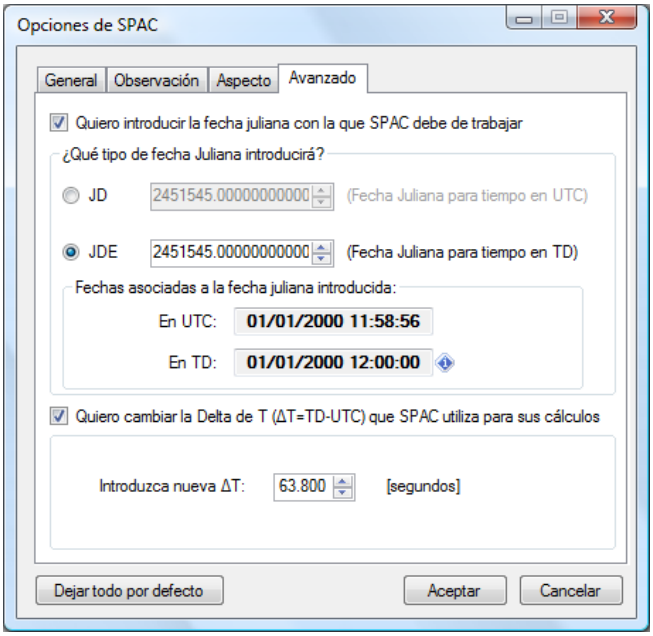


En "Aspecto" podremos decirle a SPAC como queremos que nos muestre los resultados numéricos. La ventana tiene el siguiente aspecto;



Por último la pestaña "Avanzado" daría a ofrecer al usuario una parametrización avanzada de los aspectos de la conversión de tiempo que utilizará SPAC internamente, para convertir cualquier fecha dada en fecha TD. El caso es que podemos decidir si el programa ha de utilizar su modelo de conversión (que se basaría en un modelo de predicción de $delta\ de\ T$), o por contra, establecer nosotros mismos la $delta\ de\ T$ con la que queremos que trabaje, a su vez,

podremos decirle en fecha juliana si queremos que todas las efemérides se calculen para una determinada fecha juliana.



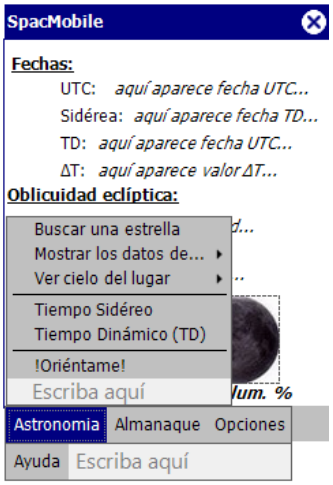
Hay que decir, que las preferencias del entorno de programa se guardan de forma independiente para cada usuario de Windows, por lo que en un mismo sistema es posible tener diferentes configuraciones de preferencias de SPAC (una para cada usuario de Windows). Las preferencias serían en si una clase *serializable* y se guardarían como tal dentro del directorio de usuario de Windows.

5.4.- Complementos para móvil:

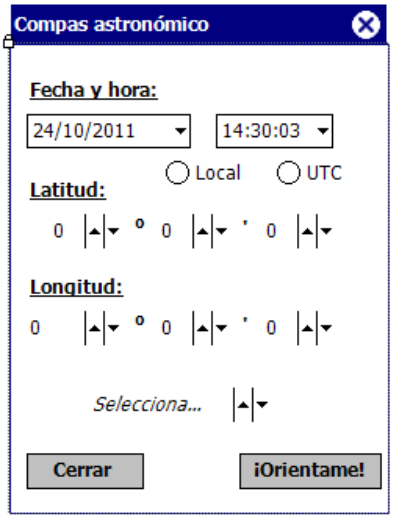
Se implementarán las funcionalidades siguientes para Windows Mobile 6.0 o superior:

- Obtener los datos del posicionamiento de un astro demandado para un momento dado. Esto es poder buscar entre los astros que maneja SPAC (Sol, Luna, Planetas y Estrellas) y mostrar los datos de dicho astro para el momento deseado.
- Función ¡Oriéntame! el usuario entrará su posición actual y el programa le mostrará dónde está el norte verdadero mediante la implementación de un compás solar ó lunar ó estelar (el propio programa ya calcula si para la posición del observador es de día o de noche o si siendo de noche hay luna nueva).
- Mostrar la fase lunar para una fecha dada.
- Determinar el momento de entrada de cada estación para un año dado.
- Mostrar el cielo del lugar. Esto es enumerar los astros que el usuario puede ver desde su posición actual en cada momento.

El aspecto inicial de la aplicación es el de presentar un resumen con la predicción de los eventos astronómicos más relevantes para la posición, hora y datos astronómicos que haya fijado previamente el usuario en las preferencias del programa. Es decir, se mostrará información; de la hora que tiene el usuario en diferentes bases de tiempo, de la oblicuidad de la eclíptica prevista por el programa para la fecha en curso, de cuando sucederá el mediodía solar para la posición del usuario actual y se muestra la fase actual de la luna y la fecha en que entrará la siguiente estación. Estos dos últimos datos tendrán una imagen que cambiará según la fase en que se halle la luna y otra según la siguiente estación que entre.

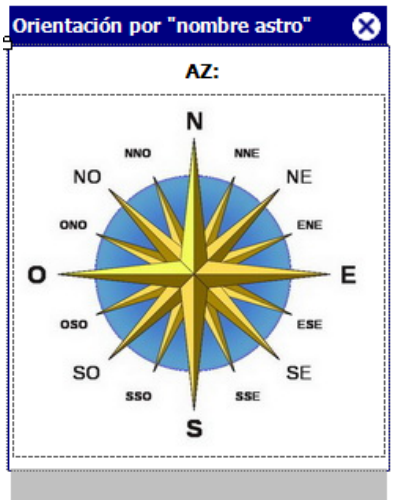


En los menús podemos ver las opciones de; buscar una estrella, mostrar los datos de los astros de la vía láctea, mostrar el cielo del lugar, calcular el tiempo sidéreo y dinámico y la función ¡Oriéntame!.



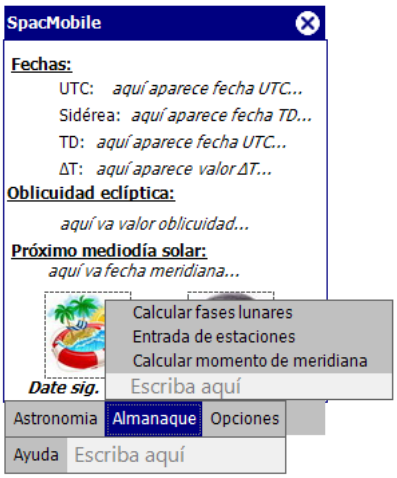
La función ¡Oriéntame! Implementa un compás astronómico, es decir, un instrumento tal que sabiendo la situación del usuario y la hora en la que éste está nos indica dónde se encuentra el norte verdadero. ¿Cómo lo hace? Muy simple, calcula el azimut de un astro para la situación geográfica del usuario y la fecha y hora de este, luego muestra una imagen de una brújula dónde indica el azimut del astro calculado. Para que el usuario pueda orientarse

únicamente debe de alinear el azimut del astro calculado sobre la imagen de la brújula mostrada con el astro físico en sí que está observado. Si por ejemplo nos orientamos por el Sol y el azimut obtenido para nuestra hora y lugar es de 45º deberemos orientar el punto cardinal de la brújula NE (que es lo mismo que 45º) con el Sol que estamos observando, haciendo así que todos los demás puntos cardinales (norte, sur, este y oeste) sean conocidos. Esta herramienta es útil en el supuesto de que sepas dónde estás (latitud y longitud) pero no poseas ningún instrumento adicional (véase brújula) para conocer dónde está el norte. Además, tiene un valor añadido y es que no obtendremos el norte magnético sino el verdadero, de modo que, si estamos en lugares con declinaciones magnéticas elevadas (polos y zonas de ruido magnético) podremos conocer el norte verdadero terrestre mientras que si en esa situación tuviésemos también una brújula esta nos indicaría el magnético que tendría un error enorme y no sería para nada fiable.



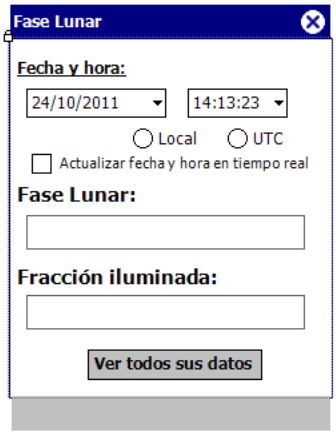
El astro por el que nos orientamos será automáticamente elegido por *SpacMobile*, teniendo en cuenta que:

- Si estamos de día el programa elegirá el Sol.
- Si es de noche y no hay Luna nueva el programa elegirá la Luna.
- Para todos los otros casos, el programa elegirá la estrella más brillante del firmamento que pueda ver el usuario para su hora y posición.



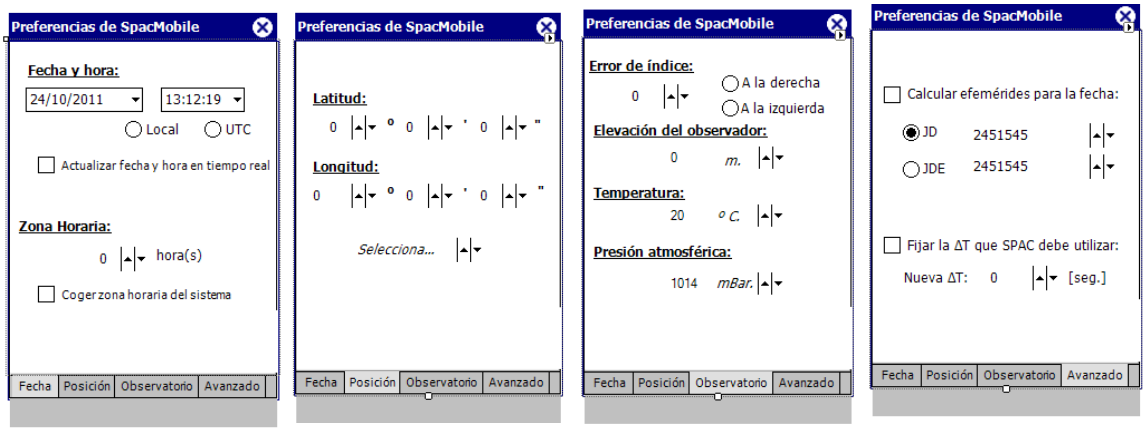
En el menú Almanaque podemos encontrar las opciones para el cálculo de la predicción de determinadas efemérides relevantes tales como; las fases lunares y la entrada de las estaciones.

Para las fases lunares tendremos un formulario en el que por defecto nos aparecerá la fecha y hora predefinida en las preferencias del programa por el usuario, no obstante podremos cambiarla. Para la fecha que elijamos nos aparecerá la fase lunar y el porcentaje de disco lunar visible, además tendremos la opción de ver todos los datos de la Luna para ese momento, en los que tendremos múltiples informaciones adicionales sobre el estado de la Luna y sus coordenadas astronómicas.



Por último en; *"Opciones>Preferencias de SpacMobile…"* tendremos todas las variables de entorno de usuario que pueden ser parametrizadas según el usuario lo desee. Tales como; fijar fecha y hora, zona horaria, temperatura, presión atmosférica, situación geográfica, elevación

con respecto el nivel del mar, delta de T a utilizar, fecha Juliana, etc. Prácticamente se incluyen tantas opciones como las dadas en la aplicación de escritorio principal (SPAC).



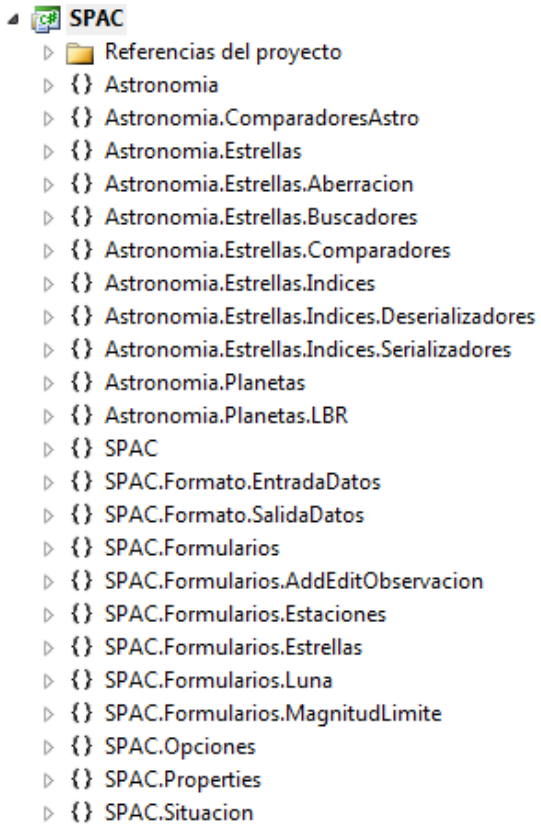
IMPLEMENTACIÓN:

1.- Espacios de nombres del programa:

Tanto en el proyecto de escritorio de Windows (SPAC) como en el de Mobile (SpacMobile) hay dos espacios de nombres claramente diferenciados; un primero y común en ambas aplicaciones orientado solamente a contener bajo él todas las clases que implementan el motor de cálculo astronómico del que se nutre el programa y un segundo espacio de nombres dedicado a contener las clases que constituyen propiamente la aplicación que estemos tratando (o bien la de escritorio de Windows o bien la de Mobile). El primer espacio de nombres tiene como raíz "**Astronomia.**" y el segundo espacio de nombres tienen como raíz "**SPAC.**"; en el caso de la aplicación para escritorio de Windows, o bien, "**SpacMobile.**"; en el caso de la aplicación para Windows Mobile.

La idea es poder trabajar con dos *namespaces* claramente distintos con la finalidad de que el motor de cálculo astronómico sea lo más independiente posible a las clases, formularios y configuraciones más intrínsecas de cada aplicación final que lo utilice. Así, en un futuro, dicho motor de cálculo astronómico, podría perfectamente ubicarse en una librería externa al programa principal que la llamase, por ejemplo, en la forma de una librería dinámica (".dll") mejorando así el reaprovechamiento del código y también mejorando notablemente la portabilidad.

Una visión global de todo el espacio de nombres del proyecto de *SPAC* sería la siguiente:



1.1.- El espacio de nombres de Astronomía:

Bajo la raíz de **Astronomia** hallamos todo el espacio de nombres del programa dedicado a la generación de la efemérides astronómicas, es decir, al cálculo de todos los datos astronómicos de los que posteriormente la aplicación principal se alimentará. Más concretamente podemos encontrar las siguientes ramificaciones:

- ***Astronomia.Planetas***: Aquí encontraremos todas las clases dedicadas al cálculo de las coordenadas, efemérides y datos astronómicos referidos a los planetas del sistema solar. En este se ubica la clase: *CoordenadasPlanetas* que sirve para generar las datos astronómicos de cualquier planeta . Dicho espacio de nombres contiene a su vez a:
    - ***Astronomia.Planetas.LBR*** dedicado a contener la interface *IPlanetasLPlanetasLBR* y la implementación de dicha interface en las clases que determinan los componentes de términos periódicos según la teoría VSOP87 (ver anexo) que el programa implementa de cada Planeta, las clases que aquí encontraremos serán las de; Mercurio, Venus, Marte, Júpiter, Saturno, Urano, Neptuno y Plutón.
- ***Astronomia.Estrellas***: Espacio de nombres dedicado exclusivamente a contener las clases e interfaces dedicadas al cálculo de los datos astronómicos de las estrellas. La clase principal aquí será *CoordenadasEstrella*. De nuevo sobre este espacio de nombre podemos distinguir los siguientes sub-espacios de nombres:
    - ***Astronomia.Estrellas.Aberracion***; dedicada a contener los términos aberración estelar y las funcionalidades necesarias para poder utilizarlos.
    - ***Astronomia.Estrellas.Buscadores***; contiene todas las clases que permiten la búsqueda de una determinada estrella según todos los campos posibles de búsqueda que permita el catálogo estelar utilizado en el programa.
    - ***Astronomia.Estrellas.Comparadores***; contiene todos los criterios de ordenación posibles para ordenar el catálogo estelar utilizado.
    - ***Astronomia.Estrellas.Indices***; que a su vez contiene a;
        - *Astronomia.Estrellas.Indices.Serializadores*
        - *Astronomia.Estrellas.Indices.Deserializadores*

      Permite la gestión de los archivos de datos referidos a las estrellas, dicha gestión es totalmente transparente al usuario.

En la raíz de *Astronomia* encontramos la clase padre abstracta de; *CoordenadasAstro* la cual implementa las funcionalidades generales de cualquier astro, tal y como se explicó en apartados anteriores. De esta clase heredan las clases hijas; *CoordenadasSol, CoordenadasLuna, CoordenadasPlanetas* y *CoordenadasEstrellas*.

2.- Clases de utilidades:

Dentro del proyecto encontramos un conjunto de clases de utilidades (*utility*) destinadas a implementar métodos que son frecuentemente invocados des de diferentes clases y partes del programa. Estas clases implementan funciones muy genéricas, tales como; calcular razones trigonométricas, calcular conversiones entre sistemas de tiempo, calcular la fecha Juliana,

realizar búsquedas generales, validar la entrada y salida de datos, convertir resultados, etc. Como en la mayoría de clases de utilidades los métodos que exponen son de tipo estático. Dichas clases de utilidades serían las siguientes:

- *Astronomia.**Buscar***: implementa todas las funciones necesarias para buscar un evento astronómico determinado dentro de la trayectoria de un astro. En concreto se permitirán hallar; el momento de meridiana de cualquier astro y el momento de entrada de las estaciones (con una distinción entre si se quiere buscar con mucha precisión, que implicará un mayor coste computacional, o con una precisión moderada, mejorando la carga computacional pero penalizando la precisión de los resultados obtenidos finales).
- *Astronomia.**FechaJuliana***: que a su vez hereda de **ConvertirFecha** y que permite realizar todas las operaciones necesarias sobre una determinada fecha expresada en una determinada base de tiempo; Local, UTC o TD y a su vez, calcular las fechas Julianas asociadas tanto; JD cómo JDE.
- *Astronomia.**Mates***: clase que complementa las funcionalidades de la clase auxiliar **System.Math** ya proporcionada en el Framework de .NET. En esta encontraremos métodos que permiten operaciones matemáticas que la anterior clase de **System.Math** no da soporte y que el programa necesita utilizar con bastante frecuencia.
- *Astronomia.**Trigonometria***: aquí se implementarán todos los métodos trigonométricos que tampoco podemos encontrar en **System.Math** y a los que el programa necesita recurrir muy frecuentemente. Esencialmente serán métodos relacionados con el cálculo de funciones de trigonometría esférica, además, de algunos concretos de trigonometría plana que se deben de implementar de alguna forma específica.
- *Astronomia.**TiempoSidereo***: clase que contiene los métodos estáticos que permiten el calcular para un momento de tiempo dado, el valor de tiempo sidéreo asociado.
- *SPAC.Formato.EntradaDatos.**ConvertirEntrada***: clase dedicada a convertir los datos entrados por el formato elegido del usuario en aquellos que el programa entienda. Por ejemplo si un usuario expresa su latitud en la forma de la cadena de texto; $53º\ 12'\ 36{,}0"\ S$ entonces esta clase, mediante su método *ConvertToDouble_mskLatitud(…)*, convertirá dicha entrada alfanumérica en un número decimal de precisión doble y expresado en grados tal que; $-53{,}21\ [º]$ (se recuerda las latitudes Sud se eligen, por convenio, como valores negativos) con el que el programa sí que podrá operar.
- *SPAC.Formato.SalidaDatos.**Presenta***: realiza el trabajo inverso de la anterior clase de ConvertirEntrada, es decir, dado un valor interno del programa, esta clase, lo convertirá al formato de presentación del mismo que el usuario haya elegido. Como ejemplo tendríamos que si el programa contiene la anterior latitud de; $-53{,}21\ [º]$ ahora entrará como argumento este valor a su método estático, *Latitud(latitud:double):string* y dicho método le devolverá la cadena alfanumérica de; $53º\ 12'\ 36{,}0"\ S$ que era la que originalmente le había introducido el usuario al programa.

Hay que decir que las clases de utilidades de *Astronomia* son idénticas tanto en *SPAC* como en *SpacMobile*. De forma similar pasa en las clases de utilidades de; *Presenta* y *ConvertirEntrada*,

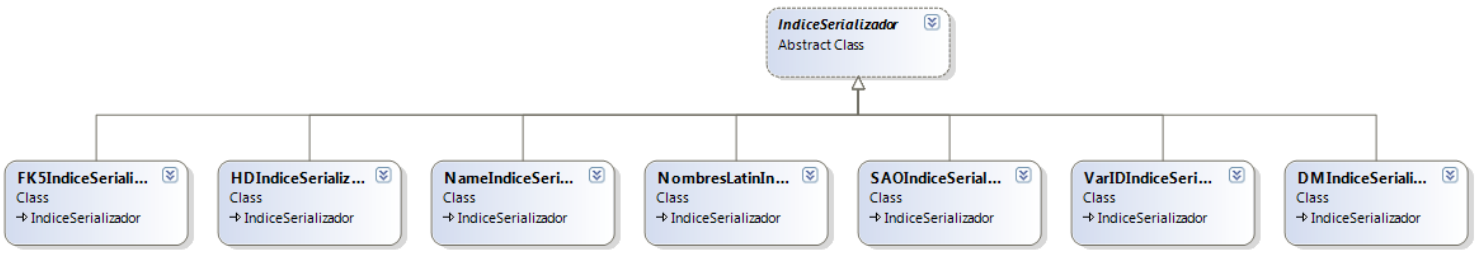
no obstante, a pesar de que se expongan los mismo métodos públicos y estos realicen las mismas funcionalidades, la implementación intrínseca de los mismos puede haber cambiado ligeramente entre las versiones de *SPAC* y *SpacMobile* debido a los requisitos de programación para cada una de las plataformas a las que van destinadas; PC y Windows Mobile.

2.1.- Clases "serializadoras" de estrellas:

Para implementar toda la estructura de datos explicada en la parte de "Estructura de datos" del capítulo de Análisis y diseño del presente trabajo, se han desarrollado las siguientes clases de utilidades, que permiten todo lo referido a la generación de los archivos serializados, que posteriormente SPAC utilizará como fuente de datos en los cálculos referidos a las estrellas.

El método público y estático **SerializarCatalogo()** de la clase **SerializadorEstrellas** perteneciente al espacio de nombres; *Astronomia.Estrellas* implementar la funcionalidad de serializar el catálogo estelar de referencia (por defecto el programa trabaja con; *Bright Star Catalogue*, ver más información en anexo) dado en formato simple de tipo CSV (columnas separadas por punto y coma) en las cien partes de archivo (ver estructura de datos) que implementan la totalidad de la tabla **CatalogoEstelar** en formato XML. A su vez esta clase permite la serialización de los índices para la búsqueda rápida de estrellas y los componentes de la aberración estelar.

La serialización de los índices para la búsqueda rápida de las estrellas según sus hasta siete denominaciones posibles, se realiza ayudada de la clase abstracta **IndiceSerializador** situada en; *Astronomia.Estrellas.Indices.Serializadores* esta se especializa en cada clase hija que implementa el método de serialización concreto para cada tipo de denominación posible de una estrella. La estructura de datos que modelaría esta relación de herencia seria el siguiente:



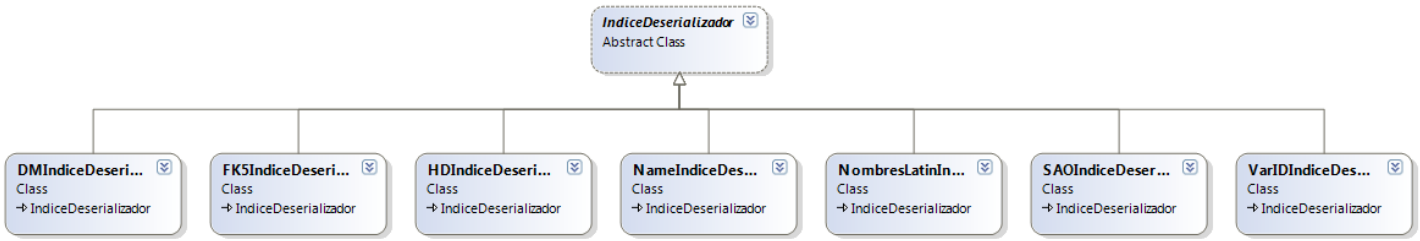
Como ya se dijo en apartados anteriores los archivos XML de los índices de búsqueda rápida se hallan en la carpeta local del programa de; **/data/index**

2.2.- Clases “deserializadoras” de estrellas:

Estas clases realizan el trabajo inverso a las clases serializadoras presentadas en el punto anterior. Es decir, gestionan todo lo referido a la recuperación de los datos que previamente han sido serializados referidos a las estrellas.

La clase de utilidades; **DeserializadorEstrellas** implementa todos los métodos estáticos necesarios para obtener el catálogo estelar completo del sistema o bien aquel fragmento de catálogo (archivo) que contiene la estrella demandada. Además en su atributo estático **CatalogoEstelar** mantiene en memoria todo aquello que en algún momento ya halla cargado del catálogo estelar, de modo que, si el usuario vuelve a demandarlo el programa comprobará primero que el catalogo estelar no este cargado ya en memoria antes de recurrir a deserializarlo de nuevo. La idea es que implemente una especie de funcionalidad de *cache* sobre el catalogo estelar.

Para deserializar los índices tendremos, igual que pasaba antes, una clase padre abstracta denominada como; **IndiceDeserializador** de la que heredan las clases hijas que especializan la forma en cómo hay que deserializar cada tipo de índice. Su estructura seria como la que sigue:



EVOLUCIÓN TEMPORAL FINAL:

Cabe destacar que se ha conseguido seguir bastante bien el planteamiento inicial de planificación temporal. No obstante, como es costumbre que pase a lo largo del desarrollo de un proyecto ha habido fases en que se ha requerido un esfuerzo adicional de trabajo con respecto aquel que se había previsto inicialmente, muy especialmente en las siguientes partes:

- Dentro de la fase de Análisis y diseño, se ha tenido que realizar un trabajado adicional importante al inicialmente previsto a la hora de analizar los algoritmos astronómicos que implementarían posteriormente el motor de generación de efemérides astronómicas del programa. Se ha tenido que recurrir a la lectura de múltiple bibliografía técnica sobre astronomía para establecer una descripción precisa de los algoritmos astronómicos que posteriormente se iban a desarrollar.

- Dentro de la fase de implementación, hay que destacar que el sobresfuerzo realizado en la fase previa de análisis y diseño centrado en dejar lo más exhaustivamente posible la descripción de cada uno de los algoritmos astronómicos a implementar, produjo aquí su fruto al poder centrarse puramente en la implementación basada en la descripción clara y concisa que previamente se había hecho de cada algoritmo y no habiendo de rectificar o rediseñar ningún algoritmo. No obstante, hay que decir que ha sido una fase muy larga y tediosa por la gran cantidad de código y clases que se han necesitado escribir y posiblemente haya sido la parte más larga y costosa del proyecto.
  Especialmente compleja fue la implementación de los algoritmos que trataban la generación de las coordenadas de las estrellas y de los planetas. Destacamos también que la implementación en Windows Mobile consumió el tiempo que se le había previsto, encontrándonos con pocas sorpresas en su adaptación gracias a un estudio previo bien hecho sobre aquello que se debía de portar a esta nueva plataforma.

- En la fase final, se presentó un esfuerzo en la síntesis y estructuración de las ideas que aparecerían en la presente memoria y a su vez en la presentación del producto final. La evolución temporal de esta última parte ha sido correcta.

De las anteriores reflexiones podemos observar como el proyecto ha evolucionado bastante adecuado a la planificación temporal inicial. Pese a que hemos visto como algunas partes habrían requerido una planificación temporal inicial más extensa podemos concluir como que la evolutiva temporal general seguida ha sido bastante acertada para el tipo de proyecto realizado y su complejidad.

ANÁLISIS DE COSTES:

Para hacer el análisis de costes se considerará como jornada completa a aquella que produzca una dedicación exclusiva de seis horas al desempeño de alguna tarea del proyecto.

Se consideran las siguientes tarifas para cada perfil laboral:

- Jefe de equipo: 70 €/hora
- Analista programador: 55 €/hora
- Programador: 40 €/hora
- Diseñador: 40 €/hora

Para hacer la estimación del coste del producto consideraremos que nada más empleamos a una sola persona de cada tipo de perfil laboral.

Podemos establecer el siguiente cuadro de resumen de costes en base al número de jornadas completas dedicadas en cada una de las fases del proyecto:

	Jefe de equipo	**Analista programador**	**Programador**	**Diseñador**
Objetivos y planificación	3	8	-	-
Análisis y diseño	5	15	-	10
Implementación	2	10	30	5
PARCIAL DE JORNADAS:	*10*	*33*	*30*	*15*
PARCIAL DE COSTE:	4200€	10890€	7200€	3600€
COSTE FINAL:	**25.890 €**			

Como se puede apreciar el coste final estimado del producto es de **25.890€**.

CONCLUSIONES:

Como el lector ha podido ir viendo a través del presente trabajo se ha realizado un programa que permite el posicionamiento mediante observación astronómica y que además permite realizar de forma completa cualquier función realizable por un planetario. Es decir, se ha conseguido un producto que cubre dos áreas interesantes; por un lado la de localización terrestre y por el otro la de la observación astronómica. Aquí vemos, por tanto, una posible evolución del presente producto en alguna de estas dos áreas, o incluso, en las dos. Por un lado encontramos multitud de programas de planetario en Internet, pero, ¿cuáles de ellos son equiparables a este? la respuesta es; "no tantos". Esto es, si tenemos en cuenta que lo realmente interesante de SPAC es su motor de generación astronómica, realizado des de cero y creado a medida para poder permitir una gran parametrización por parte del usuario final, con el fin, de que hasta el usuario más avanzado vea en él una opción de programa de referencia para sus observaciones, entonces podemos pensar que este proyecto puede resultar una alternativa interesante para muchos usuarios aficionados a la observación y navegación astronómica. Por otro lado, hasta la fecha no hay un programa que combine sus capacidades de planetario (generación de efemérides astronómicas) con prestaciones adicionales para poder posicionarse realizando así de forma completamente automática los tediosos cálculos astronómicos, que por el momento, quienes los practican, los hacen manualmente por falta de *software* especializado. En mi opinión personal esto último es un gran logro, en tanto que, dicho sistema es sustitutivo completo de la resolución manual de hasta los problemas más complejos de navegación astronómica, añadiendo;

- Rapidez; en termino medido un cálculo de posicionamiento astronómico resuelto de forma completa (sin uso de aproximaciones, tablas americanas, reglas de simplificación, etc.) requiere "a una persona experta en ello" del orden de 45 a 60 minutos para resolverlo. Con SPAC este proceso se puede realizar en segundos, que es el tiempo necesario para introducir los datos en el programa.

- Seguridad adicional por doble banda:
    - Por un lado no existe posibilidad de fallo humano en la resolución de los cálculos, puesto que es el ordenador quien los realiza.
    - Por otro lado los resultados obtenidos siempre son de mayor precisión que aquellos que se obtendrían de forma manual. Este punto requiere una explicación adicional:
        - En la resolución manual nunca se utilizará tanta precisión como usa internamente SPAC (64 bits de coma flotante) y si se "llegara a usar" nunca llegaríamos a satisfacer la siguiente condición;

- los datos astronómicos con los que trabajáramos, sobre aquellos astros que hubiésemos observado para posicionarnos, serian completamente más imprecisos que aquellos con los que SPAC pueda trabajar, ya que en la resolución manual dichos datos son obtenidos de publicaciones periódicas de los principales observatorios de cada país, en las que nos salen tabulados cada cierto intervalo de tiempo, por lo que muy fácilmente el momento de nuestra observación no será aquel que coincide con el instante de tiempo tabulado, teniendo que interpolar y "suponer" cuál hubiese sido la efeméride para ese momento de ese astro. SPAC, sin embargo, generará para cada instante de tiempo demandado por el usuario la efeméride del astro observado de forma exacta, sin interpolaciones ni aproximaciones.

- Autonomía; al generar SPAC todas las efemérides astronómicas del momento no necesitaremos llevar encima publicaciones actualizadas con los datos astronómicos de los astros, es decir, el almanaque astronómico del año en curso. Podremos, por tanto, simplemente prescindir de él y solo necesitaremos un ordenador con SPAC instalado.

- Redundancia; por la rapidez en que puede obtener resultados y por la seguridad adicional que da sobre estos, SPAC puede ser considerado como una alternativa real al GPS. Así SPAC ofrece redundancia ante un sistema como el de GPS que por su política de control y gestión (militar) no ofrece ninguna garantía de servicio a sus usuarios, esto supone que la señal puede aparecer distorsionada, modificada o simplemente desaparecer en cualquier momento y sin previo aviso. Entonces SPAC puede ser usado como alternativa para orientarse por una navegación tradicional como es la astronómica. Muchas veces pasa que navegando por zonas conflictivas la señal de GPS viene dada con un error inducido de varios, o incluso centenares, de millas. Dicho error no es detectado tan fácilmente por el marino, ya que al no tener otro sistema de posicionamiento redundante con el que contrastarlo, acabará tomando la situación del GPS como la válida cuando esta claramente no lo es. Por lo tanto ante la sospecha por parte del marino de que la señal de GPS está siendo adulterada éste podría contrastarla con aquella que le daría SPAC mediante observación astronómica y así podría ver sin mayores problemas que el GPS está siendo alterado y que por tanto no es un instrumento de navegación fiable para esas aguas.

LÍNEAS FUTURAS:

Como posibles líneas futuras de investigación y desarrollo de este proyecto podemos encontrar las que siguen;

- Hacer una interfaz gráfica más visual y llamativa en la que el cielo del lugar no aparezca simplemente tabulado, sino que tenga una representación tridimensional de la bóveda celeste del lugar permitiendo al usuario moverse por esta y explorar el cielo.

- Mejorar la portabilidad llevándolo a nuevas plataformas, especialmente a las de dispositivos móviles que funcionen con los sistemas; iOS, Android y Windows Phone. La portabilidad a sistemas iOS y Android es viable sin necesidad de recurrir forzosamente a la rescritura del código fuente gracias a proyectos como *Mono Touch* de *Novell* en los que se implementa un compilador de *C#* para estos sistemas i una implementación alternativa del *.NET Framework* para que puedan seguir funcionando.

- Desarrollar un sistema de posicionamiento integral, en el que la parte humana quede eliminada. Es decir, la aplicación, con ayuda de un sistema robótico/automático debería de ser capaz de reconocer astros y tomar la altura a estos de forma automática, sin que el usuario tenga que realizar ninguna medición con el sextante.

- Explotar en mayor medida toda la potencia de cálculo que ofrece el motor de cálculo astronómico aquí presentado, esto es, calcular efemérides de momentos astronómicos claves que, por el momento, no se dan como opción de cálculo. Especialmente interesante, seria utilizar el motor existente para poder predecir los eclipses.

BIBLIOGRAFÍA:

- ***Microsoft Visual C# 2008**, John Sharp. Serie: Paso a paso. Ed.: Anaya Multimedia. ISBN: 978-84-415-2449-1*
- ***Microsoft Visual C# 2005**, James Foxall. Serie: El libro de… Ed.: Anaya multimedia. ISBN: 978-84-415-2121-6*
- ***Visual C#**, Fco. Javier Ceballos, Serie: Enciclopedia de Microsoft, 2º edición, Ed.: Ra-Ma. ISBN: 9788478978106*
- ***Microsoft MSDN Library: Visual Studio 2008**, Microsoft Corporation.*
- ***Microsoft MSDN Library: Visual Studio 2010**, Microsoft Corporation.*
- ***Fonaments de Programació.** Ed.: UOC.*
- ***Programació Orientada a l'Objecte.** Ed.: UOC.*
- ***Estructura de la Informació.** Ed.: UOC.*
- ***Estructura i Tecnologia de Computadors.** Ed.: UOC.*
- ***Bases de dades I.** Ed.: UOC. ISBN: 84-9788-299-7*
- ***Enginyeria del programari.** Ed.: UOC. ISBN: 84-9788-065-X*
- ***Ingeniería de software orientada a objetos con UML, JAVA e Internet.** Alfredo Weitzemfeld, Ed.; Thomson Learning. ISBN: 970-686-190-4*
- ***Astronomical algorithms**, Jean Meeus, 2º edition. Ed.: Willman-Bell. ISBN: 0-943396-61-1*
- ***Astronomy with your Personal Computer**, Peter Duffett-Smith, 2º edition. Ed.: Cambridge University Press.*
- ***http://www.nasa.gov/**, Documentación online variada sobre astronomía y páginas asociadas.*
- ***Almanaque náutico; 2007, 2008, 2009, 2010 y 2011.** Publicaciones de defensa. Ed.: Ministerio de defensa. ISBN: 9781-233-6*
- ***Tratado de astronomía general y náutica,** José L. de Ribera y Egea. Ministerio de defensa.*
- ***Astronomía y navegación**, José María Moreu Curbera y Martínez Jiménez, 3ª edición. Tomos: **I, II y III**. Ed.: VIGO. ISBN: 84-85645-01-4, 84-404-0253-8.*
- ***Problemas de navegación**, José María Moreu Curbera, Ed.: VIGO. ISBN: 84-400-3741-4*

- ***Astronomia Nautica (navigazione astronomica)**, Editore Ulrico Hoepli Milano, 5ª edizione. Ed.: Ristampa.*
- ***Teoría de la astronomía**, Abel Camblor Ordiz, Ed.: Iberediciones.*
- ***Navegación costera**, Jaime Vaquero, 6ª edición. Ed.: EDICIONES PIRÁMIDE. ISBN: 84-368-1121-6*
- ***Física para la ciencia y la tecnología** (vol.: I, II y libro de problemas), Paul A. Tipler y Gene Mosca. Ed. REVERTÉ, 5ª edición.*
- ***Astronomia fonamental**, Vicent J. Martínez, Joan A. Miralles, Enric Marco y David Galadí-Enríquez. Ed. PUV, 2ª edición. ISBN: 978-84-370-6896-1*
- ***Teoría de Eclipses, ocultaciones y tránsitos**, F. Javier Gil Chica, Universidad de Alicante. Ed. II. Título. ISBN: 84-7908-270-4*