**INFORME DEL DISEÑO DEL MÉTODO DE LA INGENIERÍA**

**Contexto problemático**

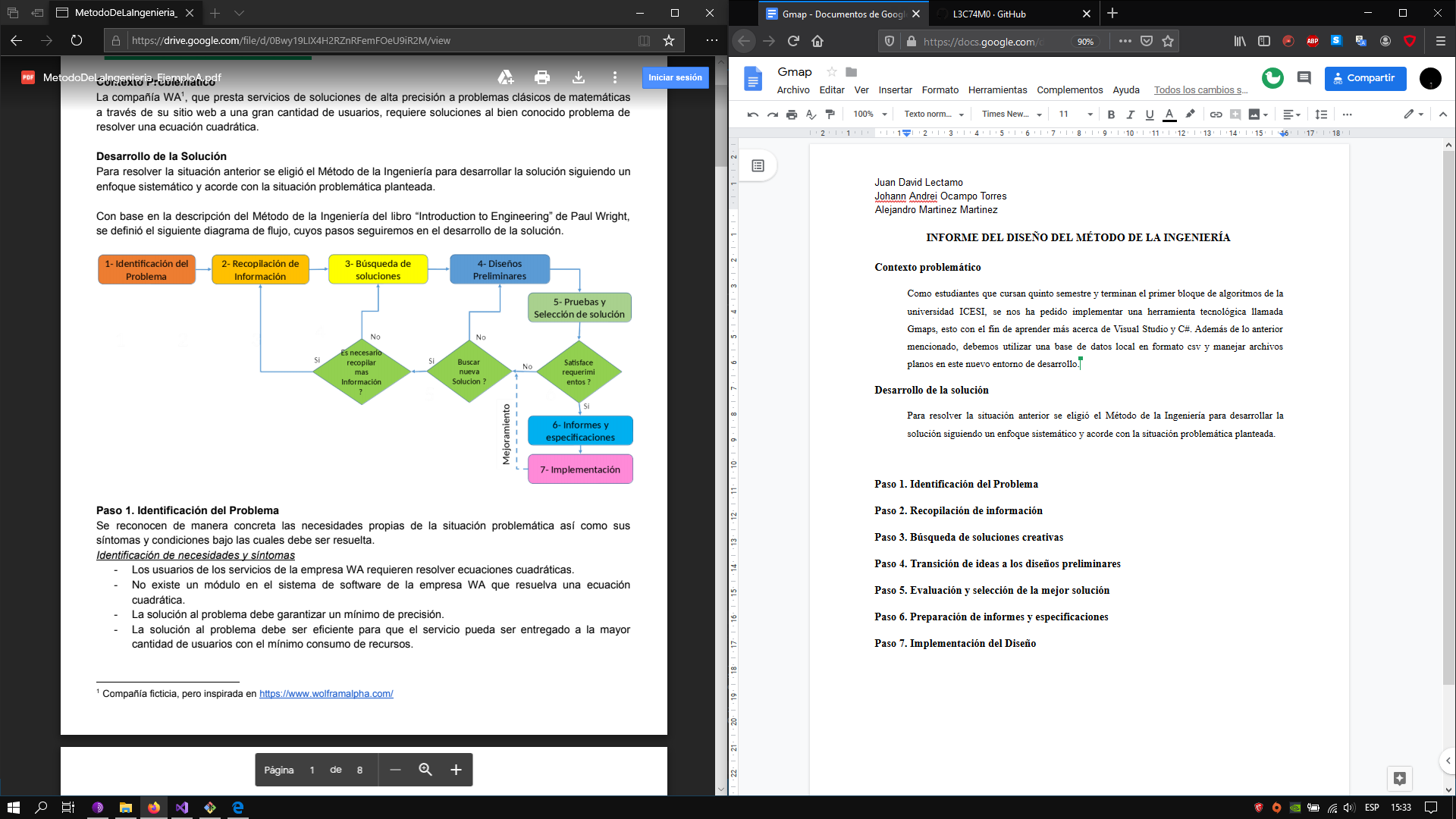
La página data world busca hacer su información más llamativa y alusiva, de esta manera los usuarios podrán encontrar más relaciones interesantes entre las informaciones que pueden estar analizando y si les interesa descargar el archivo CSV. Esto permite a la comunidad llevar a cabo investigaciones con grandes cantidades de datos que pueden ser bastantes útiles ya sean en campos de la ciencia, medicina, ingeniería, etc. La información que es potencial para esta página por lo general está muy oculta, y si es encontrada tiende a estar desorganizada lo cual hace que sea menos legible a la hora de buscar patrones relevantes para las investigaciones futuras.

Algunos tipos de información necesitan tener una visualización predominantemente estadística. En este caso en particular se precisa mejorar la forma en que se muestra la información de los vuelos albergados en la página de información ya que puede llegar a ser confusa a simple vista. Con esto se espera que investigadores(Un ejemplo pueden ser los analistas creadores de big data) puedan tener una vista más general de la información y decidir si es potencial para su uso o no.

**Desarrollo de la solución**

Para resolver la situación anterior se eligió el Método de la Ingeniería para desarrollar la solución siguiendo un enfoque sistemático y acorde con la situación problemática planteada.

Con base en la descripción del Método de la Ingeniería del libro “Introduction to Engineering” de Paul Wright, se definió el siguiente diagrama de flujo, cuyos pasos seguiremos en el desarrollo de la solución.



**Paso 1. Identificación del Problema**

Identificación de necesidades y síntomas

* Se debe mostrar toda la información de interés, organizada y sin sesgos o modificaciones.
* Se debe mostrar en el mapa de norteamérica uno a uno los vuelos realizados.
* Se debe mostrar lo más entendible e intuitivo posible.
* Los retrasos en los vuelos a menudo interfieren con los planes y obligaciones de los viajeros.

Definición del problema

Usar el archivo .csv obtenido de la pagina [www.data.world](http://www.data.world) acerca de vuelos en enero de 2017 en los Estados Unidos y representar de forma gráfica las conexiones entre ciudades.

**Paso 2. Recopilación de información**

Acerca de la pagina data.world

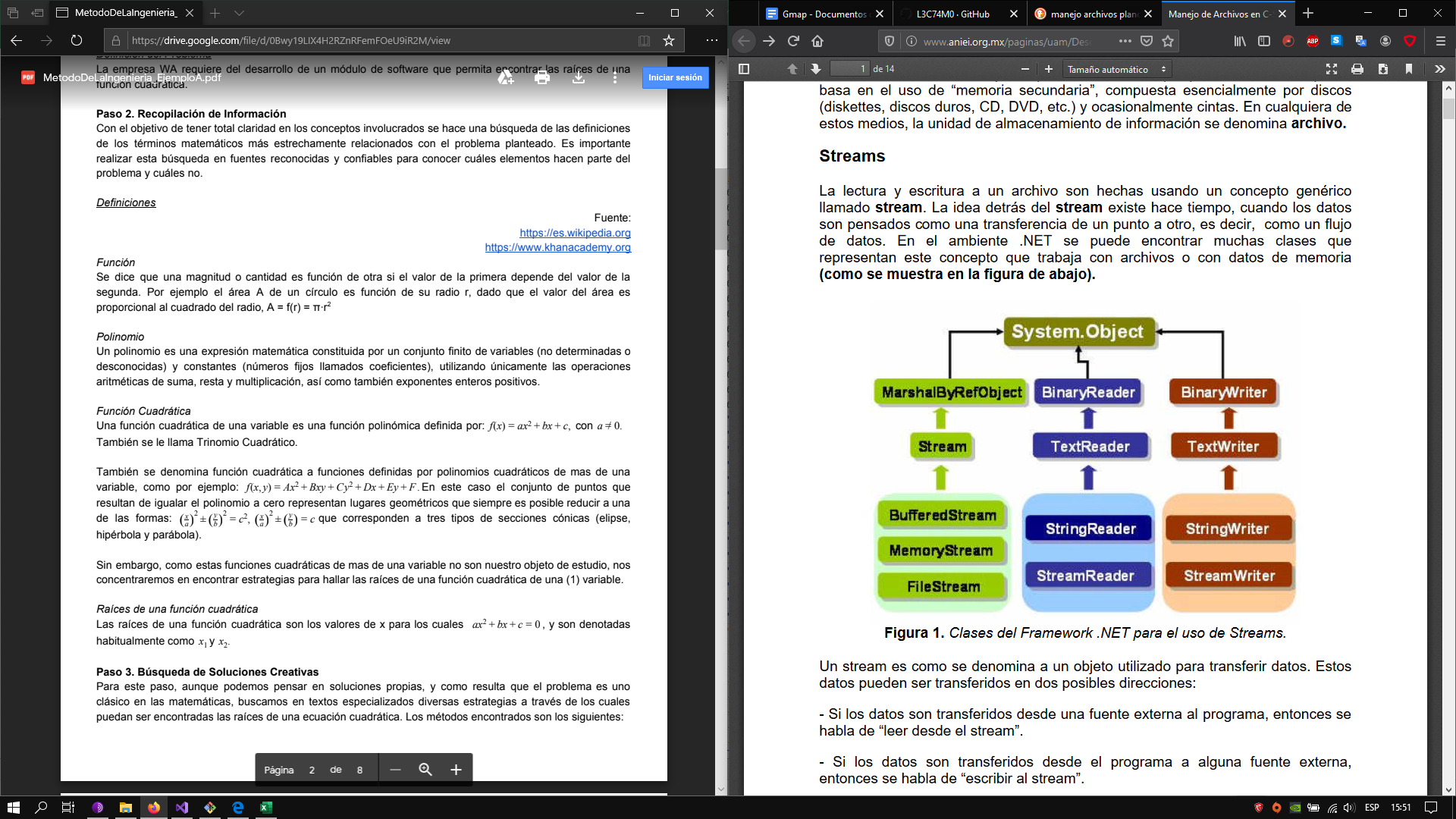
Esta página fue creada con el propósito de manejar grandes conjuntos de datos y que puedan ser utilizados en diferentes investigaciones. La misma comunidad de esta página es la que busca, organiza, estudia y sube la información que les haya parecido útil para los demás participantes.

<https://www.youtube.com/watch?v=0fCE6B-gWlg&list=LL66JEoZTlrqi0t5hCWhn0WA&index=264>

<http://www.aniei.org.mx/paginas/uam/Descargas/Recursos/archivos_csharp.pdf>

Acerca del manejo de archivos en C#

La lectura y escritura a un archivo son hechas usando un concepto genérico llamado stream. La idea detrás del stream existe hace tiempo, cuando los datos son pensados como una transferencia de un punto a otro, es decir, como un flujo de datos. En el ambiente .NET se puede encontrar muchas clases que representan este concepto que trabaja con archivos o con datos de memoria (como se muestra en la figura de abajo).



El uso de streams para la lectura y escritura de archivo es directa pero lenta. Por esta razón la clase BufferedStream existe y es más eficiente. Puede ser utilizado por cualquier clase de stream. Para operaciones de archivo es posible utilizar FileStream, donde el buffering está ya incluido.

Las clases más relacionadas con la escritura y lectura de archivos (File Input/Output o File I/O) son:

* FileStream, cuyo propósito es lectura y escritura de datos binarios (no de texto legible), a cualquier archivo de tipo binario, aunque se puede utilizar para acceder a cualquier tipo de archivo, inclusive los de texto.
* StreamReader y StreamWriter, las cuales están diseñadas para lectura y escritura de archivos de texto. Estas clases se asumen como de un nivel más alto que FileStream.

Usualmente, la ruta de un archivo contiene el carácter ‘\’, que en C# se utiliza como carácter de control para símbolos especiales (como el cambio de línea: ‘\n’). Sin embargo, entendiendo que no es el mismo sentido el que se le quiere dar en la interpretación de rutas de archivos (por ej: “C:\Mis documentos\Programas\ejemplo.cs”), se utiliza una sintaxis particular, anteponiendo el símbolo ‘@’ antes del string con la ruta del archivo. Es decir:

string ruta archivo = @”C:\Temp\archivo.txt”;

Acerca del uso de Gmaps en Visual Studio

<http://www.independent-software.com/gmap-net-beginners-tutorial-adding-clickable-markers-to-your-map-updates-for-vs2015-and-gmap-1-7.html>

GMap.NET le permite agregar tantos marcadores como desee a su mapa, con iconos de su elección (o incluso mapas de bits personalizados). La adición de marcadores se realiza mediante código y requiere los siguientes pasos:

* Cree una superposición de mapa que contendrá su colección de marcadores.
* Cree un nuevo marcador y agrégalo a la superposición.
* Agregue la superposición al mapa.

Todo lo que desee colocar en un mapa (ya sean marcadores, polígonos o rutas) debe vivir en una superposición. Puede mezclar marcadores, polígonos y rutas en la misma superposición, o puede agregarlos a superposiciones separadas. El último enfoque le permite mostrar u ocultar todos los marcadores de una vez, mientras deja visibles los polígonos y las rutas, pero depende de usted.

**Paso 3. Búsqueda de soluciones creativas**

**Para la representación de los datos**

Alternativa 1. Utilizar 2 pantallas diferentes a la vez, una que permita ver el vuelo trazando su inicio y su fin, la otra pantalla llevara la información restante del vuelo (año,mes, salida y destino) según la base de datos de data.world.

Alternativa 2. Utilizar una pantalla en la cual se muestre tanto el gmap con el trazo del vuelo y sus respectivos datos en los costados.

Alternativa 3. Además de mostrar el vuelo también mostrar el vuelo mediante un avión y que este haga la animación del vuelo según el tiempo que se tomó.

Alternativa 4. Crea una tabla de vuelos que muestren los que están más a la hora para salir y colorear el estado con más vuelos.

**Paso 4. Transición de ideas a los diseños preliminares**

**Para la representación de los datos**

**Alternativa 1:** Esta alternativa presenta en los datos divididos en 2 partes de la pantalla donde una es gráfica mostrando en el mapa en donde inició el vuelo y donde termina, lo cual permite tener un orden claro de los datos de vuelos extraídos de la base de datos. Sin necesidad de que el usuario tenga que cambiar de ventana o pantalla para ver la información buscada.

**Alternativa 2:** Es fácil de implementar pero la información puede ser muy confusa y desagradable para leer debido a que está superpuesta en el mapa.

**Alternativa 3:** La animación es representativa lo cual le daría un toque más profesional a la interfaz ya sería mucho más cómoda de ver, pero el problemas es el tiempo que tenemos para la implementación y que no nos hemos adaptado a la nueva IDE y el lenguaje de c#.

**Alternativa 4**:La información se mostrará completa, pero el problema está en que no se haría nada diferente a lo que la página muestre en sus datos, por lo tanto no es viable.

**Para la eficiencia a la hora de cargar los datos**

**Alternativa 1:** Guardar la información de los vuelos en un arrayList de tipo Vuelo.

**Alternativa 2:** Guardar la información de los vuelos en un array.

**Alternativa 3:** Guardar la información de los vuelos en un árbol binario de búsqueda.

**Alternativa 4**: No guardar la información de los vuelos y acceder a ellos desde internet.

**Paso 5. Evaluación y selección de la mejor solución**

**Para la visualización de los datos**

**Criterio 1: Comprensión de la información.**

* [3]Bastante clara
* [2]Clara
* [1]Nada clara

**Criterio 2:Representación de los datos-**

* [3]Bastante representativo
* [2]Suficientemente representativo
* [1]Nada representativo

**Criterio 3:Costo de la implementación.**

* [1]Alta
* [2]Media
* [3]Baja

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Criterio 1** | **Criterio 2** | **Criterio 3** | **Total** |
| **Alternativa 1** | **[3]** | **[3]** | **[2]** | **8** |
| **Alternativa 2** | **[3]** | **[2]** | **[3]** | **7** |
| **Alternativa 3** | **[3]** | **[3]** | **[1]** | **7** |
| **Alternativa 4** | **[1]** | **[1]** | **[3]** | **5** |

**Para la eficiencia a la hora de cargar los datos**

**Criterio 1: Costo de implementación.**

* [3] Poco tiempo
* [2] Tiempo medio
* [1] Mucho tiempo

**Criterio 2: Facilidad para el manejo de los archivos**

* [3] Fácil
* [2] No tan fácil
* [1] Difícil

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Criterio 1** | **Criterio 2** | **Total** |
| **Alternativa 1** | **[3]** | **[2]** | **5** |
| **Alternativa 2** | **[3]** | **[2]** | **5** |
| **Alternativa 3** | **[2]** | **[1]** | **3** |
| **Alternativa 4** | **[3]** | **[3]** | **6** |

Se escogió la alternativa 4, debido a que el costo de implementación es bajo y además nos permite manejar los archivos independiente de el equipo donde nos encontremos, de esta manera no debemos de cargar la base de datos en local y además podemos cambiarla por otra base de datos de vuelos con el mismo formato y el programa funcionará correctamente en todas las ocasiones. Además de lo anterior, de esta manera es muy poco probable que el archivo se corrompa.

**Paso 6. Preparación de informes y especificaciones**

**Paso 7. Implementación del Diseño**