**Método de la Ingeniería**

**Juan Ossa Ossa**

**Felipe García**

**Esteban Yusunguaira**

**Universidad ICESI**

**Facultad de Ingeniería**

**Ingeniería de Sistemas**

**Cali, Valle del cauca**

**2020**

**Identificación del problema**

**Descripción del contexto problemático (causas y síntomas)**

En el estado de MaryLand en los Estados Unidos desde el año 2012 mensualmente se han estado registrando los accidentes vehiculares, estos pueden ir desde accidentes que incluyen personas heridas hasta accidentes que en el mejor de los casos solo incluyen daños a la propiedad privada de los implicados. Las tres principales causas de accidentes en el Estado de MaryLand son: Saltarse los semáforos en rojo, no hacer las paradas entre cruces necesarias y conducir de noche.

Si bien es cierto, como se mencionó anteriormente, estos datos se van registrando de manera consistente mensualmente para poco a poco ir conformando una base datos más completa, esto no es suficiente, imagine revisar las más de 7000 líneas una por una en la que se incluye la identificación del accidente, la fecha y hora en que ocurrió, el tipo de accidente y la ubicación geográfica, esto sería supremamente costoso y *engorroso*. Es aquí donde inicia nuestro trabajo, los datos deben ser procesados, analizados e ilustrados de alguna manera para que estos lleguen a ser información útil para quien en algún momento, en este caso el objetivo es ilustrar esta información de una manera intuitiva para que las personas del estado de MaryLand puedan estar al tanto de los sectores con mayor riesgo de accidentalidad para que estas tomen las precauciones necesarias para reducir el índice de accidentalidad y en el caso de las autoridades para que estas puedan hacer los respectivos análisis de estos sectores e identificar si hace falta señalización o si se pueden tomar cartas en el asunto para reducir los accidentes.

**Identificación de necesidades:**

* Se necesita una interfaz gráfica amigable con el usuario
* Se requiere tabular la información de la base de datos
* Se requiere mostrar en el mapa los diferentes marcadores donde ha ocurrido un accidente
* Se requiere filtrar el tipo de accidente categóricamente
* Se requiere filtrar Cc Number por cadena
* Se requiere filtrar la latitud, longitud y la fecha numéricamente
* Se necesita filtrar los marcadores que aparecen en el mapa por tipo de accidente
* Se necesita mostrar un gráfico de barras con la cantidad de accidentes ocurridos desde el año 2012 hasta el año 2020
* Se necesita mostrar un diagrama de torta mostrando la participación de los diferentes tipos accidentes
* Se necesita mostrar un diagrama de puntos que muestre la cantidad de accidentes en los diferentes meses de lo llevado en el 2020

**Definición del problema**

Se requiere realizar un programa que permita visualizar los diferentes tipos de accidentes que han ocurrido desde el año 2012 hasta el año 2020 en un mapa con un marcador, además de tabular toda la información (Base de datos) en una tabla datos, además de mostrar de la cantidad accidentes ocurridos desde el año 2012 hasta el año 2020, la participación de los diferentes tipos accidentes y la cantidad de accidentes en los diferentes meses de lo llevado en el 2020 en un diagrama de barras, torta y puntos respectivamente para una óptima interpretación de la información del usuario final.

**Recopilación de la información necesaria**

**Gmaps**

GMap.NET es una poderosa herramienta gratis, multiplataforma y de código abierto .NET control. Permite usar ruteo, geolocalización, direcciones y mapas de Google, Yahoo!, Bing, OpenStreetMap, ArcGIS, Pergo, SigPac, Yandex, Mapy.cz, Maps.lt, iKarte.lv, NearMap, OviMap, CloudMade, WikiMapia, MapQuest en Windows Forms y Presentation, soporta el uso de caché y corre en Windows mobile (radioman.lt, 2013)

**Maryland y sus estadísticas de accidentes**

Según The Maryland Highway Safety Plan(2018), en 2016 el estado de Maryland experimentó un aumento de más del 18% en muertes por accidentes de tráfico respecto al año pasado. Esto alertó a la National Highway Traffic Safety Administration, los cuales han tomado como prioridad, durante los últimos años, reducir los casos de accidentes viales en Maryland.

**Estado del arte**

**Waze**

Con base en la ubicación geográfica del usuario, la aplicación muestra diferentes marcadores que describen desde otros usuarios hasta puntos de embotellamiento. Cuando estos marcadores son seleccionados, muestran información relevante según su tipo; como la velocidad de un conductor, distancia entre nosotros y el marcador, embotellamiento moderado o alto, entre otros.

Este comportamiento va muy de la mano con las necesidades del proyecto ya que se necesita que el usuario pueda identificar la información de los accidentes de una manera cómoda y eficiente.

**SOSmart**

Utilizando diferentes sensores del teléfono móvil del usuario, esta aplicación es capaz de detectar automáticamente un accidente de auto y alertar a una red de contactos sobre él. Esta red obtendrá tanto los datos personales del usuario como la ubicación del accidente a través de un mapa. Si bien el objetivo del proyecto no es brindar ayuda de en tiempo real, es una buena base para una funcionalidad que puede ser agregada en el futuro.

**Tourist Eye**

Se trata de una aplicación que marca en un mapa las locaciones más emblemáticas de una zona y muestra información turística al respecto. Lo interesante de este software es que es posible pre cargar los mapas y toda la información de los marcadores, de modo que no es necesario estar conectado a internet para hacer uso de ellos. Esta funcionalidad puede ser muy útil cuando se hacen viajes en auto ya que generalmente hay zonas en donde se pierde la cobertura.

**Especificación de requerimientos funcionales**

|  |  |
| --- | --- |
| **RF1** | **Crear una interfaz grafica para el usuario** |
| **Resumen** | Permite al usuario visualizar las diferentes funcionalidades del programa y ejecutarlas a su necesidad cuando este lo requiera |
| **Entrada** | N/A |
| **Salida** | La interfaz es mostarda |

|  |  |
| --- | --- |
| **RF2** | **Tabular información de Dataset** |
| **Resumen** | Permite cargar un DataSet que se divide en CC Number(Identificador de accidente), Fecha, Hora, Tipo de Accidente y Ubicación (Georreferencia) |
| **Entrada** | Archivo de tipo CSV |
| **Salida** | La información es cargada y tabulada |

|  |  |
| --- | --- |
| **RF3** | **Mostrar el mapa de MaryLand** |
| **Resumen** | Permite visualizar un mapa del estado de MaryLand y sus alrededores con el proveedor de mapas de Google |
| **Entrada** | N/A |
| **Salida** | El mapa es mostrado |

|  |  |
| --- | --- |
| **RF4** | **Marcar los puntos donde han ocurrido accidentes** |
| **Resumen** | Permite visualizar el usuario con un marcador de color rojo los lugares en el estado de MaryLand donde han ocurrido accidentes de transito desde el año 2012. Estos marcadores contienen información relevante a los accidentes. |
| **Entrada** | N/A |
| **Salida** | Los puntos son mostrados en el mapa |

|  |  |
| --- | --- |
| **RF5** | **Filtrar el tipo de accidente categóricamente** |
| **Resumen** | Como su nombre lo indica permite filtrar los accidentes por categorías, estas categorías son: Injury Crash, Property Crash, PI y PD |
| **Entrada** | N/A (Un tipo de accidente de la cinta de opciones) |
| **Salida** | La tabla es filtrada categóricamente |

|  |  |
| --- | --- |
| **RF6** | **Filtrar la latitud, longitud numéricamente** |
| **Resumen** | Permite filtrar las zonas donde han ocurrido accidentes por un rango numérico, accidentes que han ocurrido desde una latitud X hasta una latitud Y y accidentes que han ocurrido desde una longitud F hasta una longitud Z |
| **Entrada** | Rango de longitudes deseadas por el usuario |
| **Salida** | La latitud y longitud es filtrada tanto en la tabla como en el mapa numéricamente. |

|  |  |
| --- | --- |
| **RF7** | **Filtrar fecha numéricamente** |
| **Resumen** | Permite filtrar los accidentes por un rango de fechas |
| **Entrada** | Rango de fechas, Fecha inicial y Fecha final |
| **Salida** | La fecha es filtrada tanto en la tabla como en el mapa |

|  |  |
| --- | --- |
| **RF8** | **Filtrar identificador de accidente (CC number) por cadena** |
| **Resumen** | Permite filtrar un accidente por un identificador ingresado por el usuario |
| **Entrada** | Identificador |
| **Salida** | Tanto la tabla como el mapa se filtran según el indicador. |

|  |  |
| --- | --- |
| **RF9** | **Filtrar los marcadores que aparecen en el mapa por tipo de accidente** |
| **Resumen** | Permite al usuario seleccionar y filtrar los marcadores que se muestran en el mapa por el tipo de accidente que se desee visualizar ya sea Injury Crash, Property Crash, PI o PD |
| **Entrada** | Un tipo de accidente |
| **Salida** | Los marcadores son filtrados |

|  |  |
| --- | --- |
| **RF10** | **Mostrar un gráfico de barras con la cantidad de accidentes ocurridos desde el año 2012 hasta el año 2020** |
| **Resumen** | Permite al usuario visualizar en un gráfico de barras la cantidad de accidentes que han ocurrido desde el año 2012 hasta el año 2020 facilitando el entendimiento de la información |
| **Entrada** | N/A |
| **Salida** | El gráfico de barras es mostrado |

|  |  |
| --- | --- |
| **RF11** | **Mostrar un diagrama de torta mostrando la participación de los diferentes tipos accidentes** |
| **Resumen** | Permite al usuario visualizar un diagrama de torta con los diferentes tipos accidentes con su respectiva cantidad que han ocurrido desde el año 2012 hasta el año 2020 |
| **Entrada** | N/A |
| **Salida** | El diagrama de torta es mostrado |

|  |  |
| --- | --- |
| **RF12** | **Mostrar un diagrama de puntos que muestre la cantidad de accidentes en los diferentes meses de lo llevado en el 2020** |
| **Resumen** | Permite al usuario visualizar un diagrama de puntos con la cantidad de accidentes que han ocurrido en cada mes del año 2020 |
| **Entrada** | N/A |
| **Salida** | El diagrama de puntos es mostrado |

**Búsqueda de soluciones creativas**

* **Ideas para la selección del lenguaje de programación**

{1} Crear la aplicación en el lenguaje de programación Java, un lenguaje muy popular usado en más de 3 billones de dispositivos en el mundo y usar alguna librería que permite visualizar un mapa en la aplicación.

{2} Crear la aplicación en un lenguaje de programación por bloques como lo puede ser Appinventor y Scratch.

{3} Buscar una aplicación que ya alguien haya creado en cualquier tipo de lenguaje y adaptarla a las necesidades del cliente.

{4} Crear nuestra aplicación en el lenguaje de programación Python, un lenguaje que en la última década ha tenido un crecimiento y un avance considerablemente alto.

{5} Crear la aplicación en excel aplicacion que muchas personas usan y conocen a dia de hoy.

{6} Crear la aplicación desde 0 en el lenguaje de programación C# haciendo uso de las diferentes herramientas que este ofrece para la visualización de datos.

* **Ideas para la visualización del mapa**

{1} Hacer uso de la librería gmaps en Java para visualizar el mapa en el programa desarrollado.

{2} Hacer uso de la librería de gmaps.Net.Windows Forms para visualizar el mapa en el programa desarrollado.

{3}Pegar una imagen del mapa de MaryLand en la interfaz del programa desarrollado.

{4} No usar ningún mapa y simplemente mostrar la latitud y la longitud para que el usuario manualmente escriba las mismas en su mapa de preferencia.

{5} Omitir este punto.

* **Ideas para la carga del DataSet**

{1} Hacer la carga del DataSet mediante el uso de métodos propios del lenguaje de programación escogido combinado con métodos creados por nosotros.

{2} Hacer la carga del DataSet mediante el uso exclusivo de métodos propios del lenguaje de programación escogido.

{3} No hacer uso de un DataSet si no que por el contrario el usuario escriba. manualmente cada una de las ubicaciones que desee usar en el programa.

* **Ideas para la creación de la interfaz gráfica del usuario**

{1} Contemplar la posibilidad de realizar la interfaz gráfica mediante el uso de la librería gráfica de javafx.

{2} Realizar toda la interfaz gráfica mediante el uso de la librería gráfica swing.

{3} Realizar toda la interfaz gráfica mediante el uso de la librería gráfica AWT.

{4} Realizar toda la interfaz gráfica del programa mediante el uso de las librerías gráficas WPF.

{5} Realizar toda la interfaz gráfica del programa mediante el uso de las librerías gráficas de windows forms.

{6} No hacer ninguna interfaz gráfica y solamente usar la consola de comandos.

**Transición de la formulación de ideas a los diseños preliminares**

* **Ideas para la selección del lenguaje de programación:**

**Alternativa 1:** Realizar la aplicación en Java puede ser una muy buena alternativa ya que este lenguaje tiene una alta compatibilidad con diferentes dispositivos electrónicos y diferentes sistemas operativos, además de que es un lenguaje de programación en el que se puede encontrar mucha documentación al respecto que puede facilitar la realización del programa.

**Alternativa 2:** A pesar de que la programación por bloques resulta ser muy fácil e intuitiva es poco efectiva ya que se encuentra muy limitada por las funciones que el desarrollador de este “lenguaje” ha creado, y puede que una simple función resulte muy compleja de desarrollar.

**Alternativa 3:** Aunque tomar una aplicacion de código abierto ya desarrollada por alguien más esto puede llegar a traer principalmente dos problemas, el primero es que estas cuentan con funciones desarrolladas que no se necesitan y otras que se necesitan pero no se comprende por completo su función y en segundo lugar se ve en juego la parte moral ya que se ha pedido una aplicación hecha a medida.

**Alternativa 4:** Puede resultar viable ya que Python es un lenguaje de programación que ha tenido un alto apogeo en la última década, cuenta con una basta documentación y resulta poco complicado programar. Además, posee una gran cantidad de librerías que facilitarían el desarrollo de la aplicación.

**Alternativa 5:** Aunque excel es una herramienta que es conocida en todo el mundo y resulta ser muy útil para diferentes actividades de cálculo de datos, se ha requerido una aplicación realizada a medida y excel puede llegar a limitar mucho cumplir este requerimiento

**Alternativa 6:** Realizar la aplicación desde cero en el lenguaje de programación C# sería una gran opción, ya que brinda una alta flexibilidad a la hora de realizar todas las funcionalidades requeridas por el usuario. Además, cuenta con una amplia documentación que se encuentra en la red que facilita la programación de la misma y tiene muchas librerías referentes a la geolocalización.

* **Ideas para la visualización del mapa:**

Alternativa 1: Hacer uso de gmaps para Java puede ser una solución viable aunque puede llegar a tomar más tiempo de lo esperado debido a la falta de experticia en el uso de esta librería.

Alternativa 2: gmaps.Net.Windows Forms es una librería con la cual el equipo de desarrollo está familiarizado y cuenta con una amplia documentación que facilitará su desarrollo. Ademas gmaps.Net permite seleccionar el proveedor de mapas Google, el cual es compatible con los diferentes dispositivos que se encuentran en el mercado lo que dará mucho valor a las funcionalidades previamente señaladas.

Alternativa 3: Mostrar una imagen con marcadores precargados en vez de cargar un mapa de proveedores como Google y Bing pueden facilitar la programación de la aplicación, sin embargo el usuario tendría que ingresar cada una de las más de 7000 ubicaciones geográficas lo cual solo dificulta la vida del usuario final cuando en realidad debería generar calidad de vida.

Alternativa 4: Aunque el no implementar un mapa facilita mucho el trabajo del programador esto no puede ser considerado una solución ya que el usuario final esta haciendo un trabajo adicional que el programa fácilmente podría hacer de manera automática

Alternativa 5: No completar este punto podría tener como consecuencia un usuario final insatisfecho al cual no se le cumplio con las necesidades presentadas al inicio de esta documentación.

* **Ideas para la carga del DataSet**

Alternativa 1: Combinar el uso de métodos propios del lenguaje de programación escogido con métodos que se crearán en el transcurso del desarrollo de la aplicación trae lo mejor de ambos mundos, eficiencia y flexibilidad que permitirá adaptarse a las necesidades del usuario final

Alternativa 2: Hacer la carga del Dataset mediante el uso exclusivo de métodos del lenguaje de programación es una solución viable ampliamente usada en la industria del desarrollo de software, aunque haga falta experticia por parte de los programadores en este ámbito no deja de ser una solución efectiva al problema.

Alternativa 3: Aunque el no implementar un DataSet facilita mucho el trabajo del programador esto no puede ser considerado una solución ya que el usuario final está haciendo un trabajo adicional que el programa fácilmente podría hacer de manera automática y no se está cumpliendo con las necesidades del usuario final.

* **Ideas para la creación de la interfaz gráfica del usuario:**

Alternativa 1: Realizar una interfaz gráfica en javafx brinda un amplio abanico opciones para el desarrollo de una interfaz amigable con el usuario final, asimismo es una tecnología que lleva muchos años en el mercado y se pueda encontrar mucha documentación al respecto en la red, por otro lado se cuenta con la experticia necesaria para llevar a cabo su uso.

Alternativa 2: Desarrollar la interfaz gráfica mediante el uso de Swing puede traer muchos beneficios en el aspecto de que resulta más fácil programar la interfaz con esta tecnología lo que en consecuencia lleva a un menor tiempo en el desarrollo de la aplicación, sin embargo no resulta ser viable a largo plazo ya que genera un alto acoplamiento al código base del programa, lo que quiere decir que a largo plazo dificulta el mantenimiento del programa.

Alternativa 3: Desarrollar la interfaz gráfica mediante el uso de AWT puede traer muchos beneficios en el aspecto de que es una herramienta que lleva muchos años en el mercado y se encuentra mucha documentación de la misma, sin embargo otras tecnologías como lo son Swing y Javafx más modernas han ocasionado que la tecnología AWT se quede obsoleta.

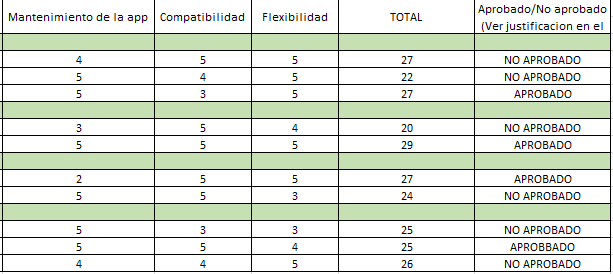
Alternativa 4: Hacer uso de Windows Presentation Foundation (WPF) trae muchos beneficios como lo son que presenta un bajo acoplamiento con el código base del programa lo que facilita su mantenimiento a largo plazo, es una de las herramientas más nuevas en el entorno de desarrollo .NET una de sus desventajas es que cuenta con menos documentación que tecnologías mencionadas anteriormente lo que genera un tiempo de desarrollo mayor, aun así resulta una solución viable que se adapta a las necesidades el usuario final.

Alternativa 5: Usar Windows Forms para el desarrollo de la interfaz gráfica es una solución viable para la aplicación ya que permite la creación de una interfaz gráfica intuitiva y amigable con el usuario. Aunque cuenta con más acoplamiento al código base que WPF cuenta con una amplia documentación que facilita el desarrollo y cumpliendo con las principales necesidades del usuario final.

Alternativa 6: No hacer una interfaz gráfica no puede ser considerada una solución porque no cumple con las necesidades del usuario final y no permitirá el uso adecuado de la aplicación.

**Evaluación y selección de la mejor solución**

****

****

**Lenguaje de programación:** Como lenguaje de programación se optó por hacer uso de C#, aunque este obtuvo la misma calificación que el lenguaje de programación Java se decidió tener por aparte de la rúbrica en este segmento el criterio de calificación “Nuevos Conocimientos” y C# obtuvo un alto valor en este segmento ya que permitirá adquirir conocimientos que antes no se tenían sobre un lenguaje que antes era desconocido permitiendo incursionar en nuevas fronteras.

**Visualización del Mapa:** Para la visualización del mapa se decidió hacer uso de gmaps.Net.Windows Forms por dos principales razones, en primer lugar al haber decidido hacer uso del lenguaje de programación C# el uso de gmaps en Java no era posible ya que no se trabajará en este lenguaje. En segundo lugar, se tuvo en cuenta experticia que se tiene sobre el tema y se detectó que con gmaps en Java se contaba con poca experiencia que tendría como consecuencia un tiempo de desarrollo mayor innecesariamente.

**Carga del Dataset:** Para la carga del Dataset se decidió hacer un uso mixto de métodos tanto propios de C# si no también métodos que se desarrollarán desde cero debido a que la combinación de ambas estrategias permiten obtener la flexibilidad requerida por el usuario final. Por otro lado, esta solución fue la que obtuvo el mayor puntuación en la rúbrica.

**Interfaz gráfica de usuario:** Para la visualización de la interfaz de usuario hubo una difícil decisión entre implementar Windows Forms y WPF ya que WPF presentaba contundentes beneficios sobre Windows Forms como lo es el desacoplamiento que este ofrece sobre el código general sin embargo dos aspectos fueron decisivos para decantarse por Windows Forms, por un lado Windows Forms contaba con mayor documentación que facilita el desarrollo de la aplicación por otro lado se tiene más experticia en Windows Forms cuando en WPF además de que se tiene más bien poca experiencia con esta tecnología, WPF cuenta con menos documentación lo que se traduciría en un mayor tiempo de desarrollo sin dar valor agregado al usuario final ya que ambas cumplen con la función de mostrar la información de manera amigable en pantalla de la misma manera.

**Diagramas**

[**Diagrama de clases y objetos**](https://docs.google.com/document/d/1rmJdoiHNYfx3PeH7QsEuhkQeCZAK4HvZ_cvTyG_JWgg/edit?usp=sharing)

**Síntesis reflexiva**

El proyecto cumplió con todas los requerimientos y expectativas iniciales, sin embargo, podría darse una solución más completa y no limitarse al uso de filtros. El dataset escogido podría aprovecharse de muchas maneras pero dadas las especificaciones de la consigna, sus limitaciones tecnológicas (C# y el uso de frameworks) y el tiempo máximo de entrega, no será posible. Incluso podría pensarse en el uso de este dataset como base para una aplicación que pudiera brindar ayudas en tiempo real y crecer según nueva información que provean sus usuarios.

En cuanto a la implementación, pudo crearse una estructura de datos propia con la información del dataset en vez de trabajar directamente con las tablas de datos. De este modo el código sería más legible y fácil de trabajar.

**Referencias**

* Maryland Department of Transportation. (2018). Maryland Highway Safety Plan. Obtenido de National Highway Traffic safety Administration: https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/maryland\_fy2018\_hsp.pdf
* radioman.lt. (16 de Abril de 2013). GMaps.NET. Obtenido de Code Project: https://www.codeproject.com/Articles/32643/GMap-NET-Great-Maps-for-Windows-Forms-and-Presenta