Proyecto Integrador I

Profesor: Juan Manuel Reyes

Proyecto final del curso - Método de la Ingeniería

Integrantes

Santiago Rodas Rodriguez

Juan Fernando Martinez

Alejandro Garcia

Gonzalo De Varona

Santiago de Cali, mayo de 2021

Fase 1: Identificación del problema

Estados Unidos es el tercer país más poblado del mundo con un total de 330 millones de habitantes. En él, aproximadamente se encuentran 106 millones de viviendas donde las personas de los diferentes estados habitan. Además, se tiene en cuenta que en un futuro este mercado avance de forma directa con el aumento de habitantes en cada una de las regiones nativas. Es por esta razón que en algunas ocasiones las compañías de seguro, impuestos de registro y la administración gubernamental, se ven paralizadas por el número de peticiones o encargos que cada día manejan.

Tomando en cuenta esta situación, se necesita realizar un software de aprendizaje autónomo con la idea de agilizar los procesos de compra y venta de cada una de las viviendas solicitadas, donde se nece dominando los siguientes requerimientos funcionales:

1. Leer y cargar el dataset “House Sales in King County, USA” proveniente de Kaggle, tomando en cuenta variables como: fecha, precio, número de pisos, pies cuadrados construidos y cantidad de habitaciones, entre otras.
2. Visualizar los datos del dataset, en formato de tabla, de manera adecuada en cualquier instante de la ejecución del programa.
3. Filtrar los datos apropiadamente teniendo en cuenta el tipo de variable:
    A) Categórica: Proveer una lista de los distintos valores posibles de la variable.
    B) Numérica: Permitir escribir un rango para filtrar por los datos.
    C) Cadena: Dar un campo para digitar la palabra.
    D) Fecha: Posibilitar la selección de un rango de fechas.
4. Ordenar los datos de menor a mayor y viceversa de acuerdo a cierta variable que es escogida por el usuario.
5. Establecer un criterio o serie de criterios para predecir el precio del inmueble a partir de los demás factores presentados en el dataset, estos son: número de pisos, habitaciones y baños, pies cuadrados construidos, entre otros. El criterio debe obtenerse empleando herramientas creadas por el equipo y también herramientas ya disponibles previamente.
6. Clasificar una vivienda nueva designando un rango de precio con base en el o los criterios establecidos con anterioridad, y los datos del nuevo hogar.
7. Generar gráficos estadísticos analizando los datos brindados por el dataset seleccionado con anterioridad. Serán un total de 5 designados de la siguiente manera: gráfico de líneas, circular, relieve, barras (2 con diferentes datos analizados).

8. Diseñar, desarrollar e implementar el experimento con el cual se va a analizar los criterios obtenidos a partir del dataset, analizando algunos factores variables de salida, número de repeticiones, niveles o tratamientos.

9. Ejecutar el experimento anteriormente implementado para lograr obtener algunos resultados frente a los de herramientas usadas (propias y externas). Logrando así evaluar de manera significativa el experimento, valorando los factores estudiados, pero principalmente, concluir una comparación exhaustiva entre la implementación propia y la librería externa.

10. Realizar un análisis estadístico para conocer la naturaleza, características, estado y factores que intervienen en todo el experimento planteado. Además, se evidenciaría una base sólida con gráficas, dependencias e independencias.

11. Diseñar una interfaz gráfica de usuario donde se logre realizar cada una de las funcionalidades deseadas para el proyecto:

    A) Visualización y filtración de los datos
    B) Gráficos estadísticos
    C) Muestra de los criterios para clasificar nuevos datos
    D) Experimentación

Fase 2: Recolección de la información necesaria

Para solucionar este problema y realizar un correcto software, es indispensable conocer algunas herramientas importantes que permitan cumplir con cada uno de los requerimientos funcionales anteriormente descritos. Algunas de estas son:

Machine-Learning: Es una habilidad indispensable para hacer sistemas capaces de identificar patrones entre los datos para hacer predicciones. Esta tecnología está presente en un sinfín de aplicaciones como las recomendaciones de Netflix y Spotify o las respuestas inteligentes de Gmail y el habla de Siri y Alexa. Además, es capaz de convertir una muestra de datos en un programa informático capaz de extraer inferencias de nuevos conjuntos de datos para los que no ha sido entrenado previamente

Los algoritmos de Machine-Learning se dividen en tres categorías, siendo las dos primeras las más comunes:

- Aprendizaje supervisado: estos algoritmos cuentan con un aprendizaje previo basado en un sistema de etiquetas asociadas a unos datos que les permiten tomar decisiones o hacer predicciones. Un ejemplo es un detector de *spam* que etiqueta un *e-mail* como *spam* o no dependiendo de los patrones que ha aprendido del histórico de correos (remitente, relación texto/imágenes, palabras clave en el asunto, etc.).

- Aprendizaje no supervisado: estos algoritmos no cuentan con un conocimiento previo. Se enfrentan al caos de datos con el objetivo de encontrar patrones que permitan organizarlos de alguna manera. Por ejemplo, en el campo del *marketing* se utilizan para extraer patrones de datos masivos provenientes de las redes sociales y crear campañas de publicidad altamente segmentadas.

- Aprendizaje por refuerzo: su objetivo es que un algoritmo aprenda a partir de la propia experiencia. Esto es, que sea capaz de tomar la mejor decisión ante diferentes situaciones de acuerdo a un proceso de prueba y error en el que se recompensan las decisiones correctas. En la actualidad se está utilizando para posibilitar el reconocimiento facial, hacer diagnósticos médicos o clasificar secuencias de ADN.

Algunas aplicaciones de esta rama tecnológica pueden ser:

- Motores de búsqueda
- Diagnóstico médico
- Detección de fraudes con el uso de tarjetas de crédito
- Análisis del mercado de valores
- Clasificación de secuencias de ADN
- Reconocimiento del habla
- Robótica
- Minería de datos
- Big Data

Árbol-de-decisión: Esta herramienta nos ayuda a la hora de tomar decisiones correctas cuando se tienen grandes cantidades de datos, y se presenta como posibles ramas en las que se plantean diferentes soluciones a un escenario continuo de sucesos. Asimismo, tomando en cuenta cada complejidad situada en diferentes planteamientos, la tecnología hablada presenta diferentes uso como:

- Plantear problemas desde diferentes puntos de vista.
- Analizar todas las posibles soluciones a una misma decisión.
- Estimar factores cuantitativos cuando el problema lo permita.
- Tomar decisiones en base a información real y confiable.
- Analizar diferentes alternativas y probabilidades que cada rama puede generar.

Los árboles de decisión están formados por nodos, vectores de números, flechas y etiquetas.

- Cada nodo se puede definir como el momento en el que se ha de tomar una decisión de entre varias posibles, lo que va haciendo que a medida que aumenta el número de nodos aumenta el número de posibles finales a los que puede llegar el individuo. Esto hace que un árbol con muchos nodos sea complicado de dibujar a mano y de analizar debido a la existencia de numerosos caminos que se pueden seguir.

- Los vectores de números serían la solución final a la que se llega en función de las diversas posibilidades que se tienen, dan las utilidades en esa solución.

- Las flechas son las uniones entre un nodo y otro y representan cada acción distinta.

- Las etiquetas se encuentran en cada nodo y cada flecha y dan nombre a cada acción.

Además, los árboles de decisión debe cumplir una serie de reglas:

1. Al comienzo del juego se da un nodo inicial que no es apuntado por ninguna flecha, es el único del juego con esta característica.

2. El resto de los nodos del juego son apuntados por una única flecha.

3. De esto se deduce que hay un único camino para llegar del nodo inicial a cada uno de los nodos del juego. No hay varias formas de llegar a la misma solución final, las decisiones son excluyentes.

Por último, estas decisiones que se eligen son lineales, ya que a medida que vas seleccionando entre varias opciones se van cerrando otras, lo que implica normalmente que no hay marcha atrás. En general se podría decir que las normas siguen una forma condicional: Opción 1 -> opción 2 -> opción 3 -> Resultado Final X. Estas reglas suelen ir implícitas en el conjunto de datos a raíz del cual se construye el árbol de decisión.

Gráficos: Una parte indispensable del programa es generar gráficos estadísticos que permitan entender el problema de manera significativa. Por ende, es de gran importancia reconocer cuales pueden ser algunos tipos de gráficos que podemos utilizar, y cuales son de gran necesidad para el problema planteado.

Lector-de-archivos: Para utilizar el sistema de forma eficaz, es necesario leer rápidamente un dataset con toda la información existente hasta el momento. De esta manera no solo se evita perder datos significativos, sino que también permite filtrar casillas por cualquier tipo de limitación que el mismo usuario interponga.

Además de esas funcionalidades, se complementa la información con los requerimientos no funcionales encontrados en el sistema:

1. Depositar el código fuente, y cualquier otro cambio en el sistema en la plataforma GitHub.

2. Utilización de una librería externa de C# que nos ayude a resolver la clasificación y análisis de datos planteado en el enunciado, tomando en cuenta el dataset elegido con anterioridad. Para hacerlo, se toma en cuenta este video como una base para iniciar.

3. Diagramar la base del software por medio de clases (UML), y objetos (con base al primero respectivamente). En este caso, se tomará en cuenta la herramienta Visual Paradigm Online para visualizar, modificar y agregar todas las clases, requerimientos y paquetes encontrados en el sistema.

4. Programar e implementar el programa en el lenguaje C# (Visual Studio).

5. Codificar e implementar un árbol de decisión como la estructura primaria del programa. Además, codificar un algoritmo que permita clasificar toda la información de manera correcta y efectiva.

6. Diagramar 3 secuencias importantes del programa para tener un mayor entendimiento en algunos aspectos del software.

7. Utilizar un dataGridView y un splitContainer para desarrollar el requerimiento funcional número 2, donde se basa principalmente en la visualización y organización de la tabla.

8. Al momento de ejecutar el experimento, evidencia una buena toma de datos para realizar un futuro análisis exhaustivo, donde por medio de ANOVA logremos encontrar conclusiones importantes del software.

9. Manejar una complejidad temporal baja, ya que si estamos manejando grandes cantidades de datos, es mejor evidenciar una buena conducción de tiempo y resultados.

10. Al momento de diseñar y desarrollar el experimento en la iteración 4, definir apropiadamente cada uno de los factores que se deben de tomar en cuenta, incluyendo el número de repeticiones y la cantidad de niveles apropiados.

Fase 3: Búsqueda de soluciones creativas

Para llevar a cabo esta etapa, se toman en cuenta 3 fases significativas:

- Analizar los requerimientos indicados para el proyecto, diseñando un problema cuya solución pueda plantearse en términos de lo solicitado en el enunciado.

- Dialogar de manera significativa entre los integrantes del grupo de trabajo para determinar la funcionalidad principal del proyecto, claramente modelado como un software futuro.

- Definir los atributos y exigencias propias del problema elegido para iniciar el desarrollo principal.

Para tomar en cuenta las fases anteriormente mencionadas, se utiliza la herramienta Brainwriting como método principal al momento de generar ideas creativas:

1. Realizar una interfaz gráfica cómoda con el usuario, permitiendo así no solo tener una correcta visualización del programa, sino que también sea eficaz al momento de utilizar el software. Ganando tiempo, y sobre todo, agilizando procesos.

   Comentario: No es específico en cuanto al diseño del programa o el backend.

2. Implementar un splitcontainer que sea agradable a la vista del usuario, y que al mismo tiempo sea eficiente al momento de usar el programa con todas sus funcionalidades.

   Comentario: Es una buena idea para hacer los tiempos del programa más rápidos.

3. Generar los 5 gráficos estadísticos dados en los requerimientos funcionales, tomando en cuenta el API oficial de Microsoft, validando los constructores, propiedades, métodos, eventos, etc.

   Comentario: Excelente idea para tener buenas prácticas de programación.

4. Desarrollar un mensaje emergente que dé a entender al usuario el error de filtrar los datos con nodos que no existen en el sistema. Además, es importante destacar que tanto la tabla como las gráficas serán visualizadas en un solo formulario, facilitando el filtrado y acelerando algunos componentes del sistema.

Comentario: Buena idea, ya que está tomando en cuenta la eficacia del usuario.

5. Guardar los gráficos generados por el usuario en un archivo pdf, esto con la idea de almacenar la información y que esta no sea borrada del sistema.

   Comentario: Se puede hacer, pero no es necesario.
6. Guardar la información obtenida del dataset en un dataGridView para evitar la creación de un objeto adicional, ya que este es necesario para la visualización de los datos.

   Comentario: ¡Muy buena idea, felicitaciones!

7. Separar en pestañas la tabla, los gráficos y los otros elementos del sistema.

   Comentario: Puede ser una alternativa

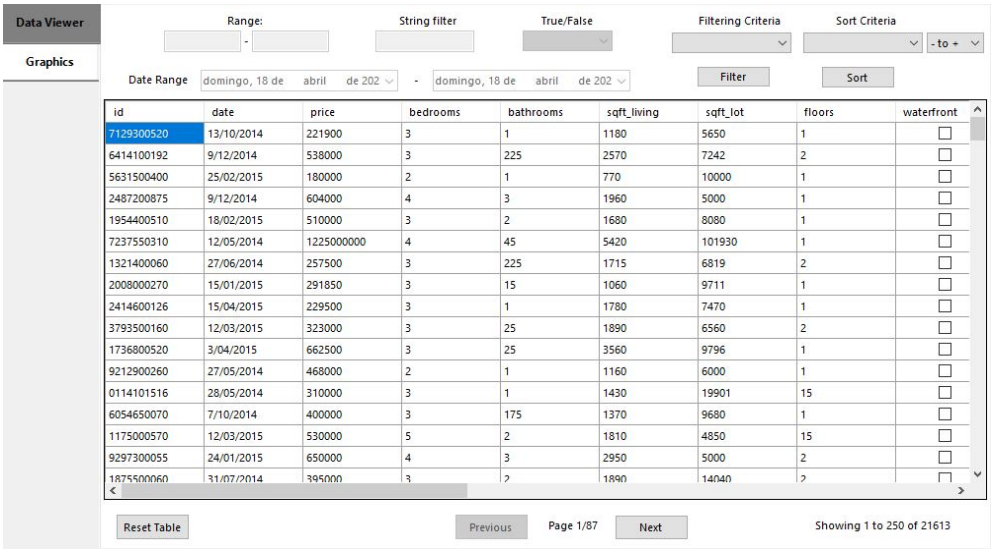
Fase 4: Pasar de la idea principal al diseño preliminar

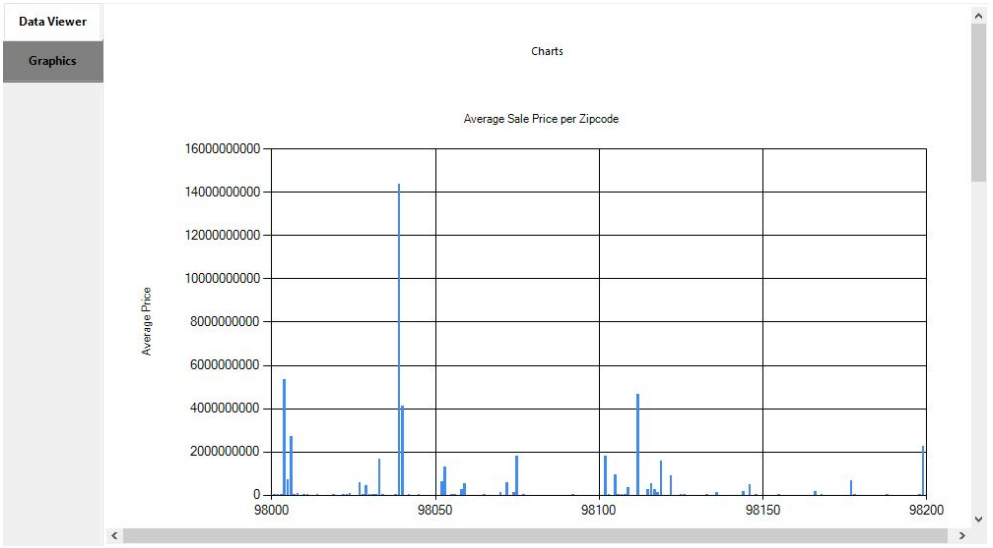
- Se descarta la idea de generar un archivo pdf con los gráficos generados, ya que en un primer lugar no es estrictamente necesario, y en segundo lugar los tiempos de entrega no dan a basto con todas las funcionalidades planeadas.

- También se tiene en cuenta cada idea de la interfaz gráfica, pero no son útiles al momento de pensarla o crearla, ya que en pocas palabras solo debemos diseñar algo útil y claro al mismo tiempo.

- Se toma en cuenta las ideas de diseñar el formulario en pestañas, utilizando un dataGridView para mostrar la información y enviar un mensaje de error cuando el usuario se equivoca filtrando los datos con información que no existe en el sistema.

Modelo principal: Utilización de pestañas.

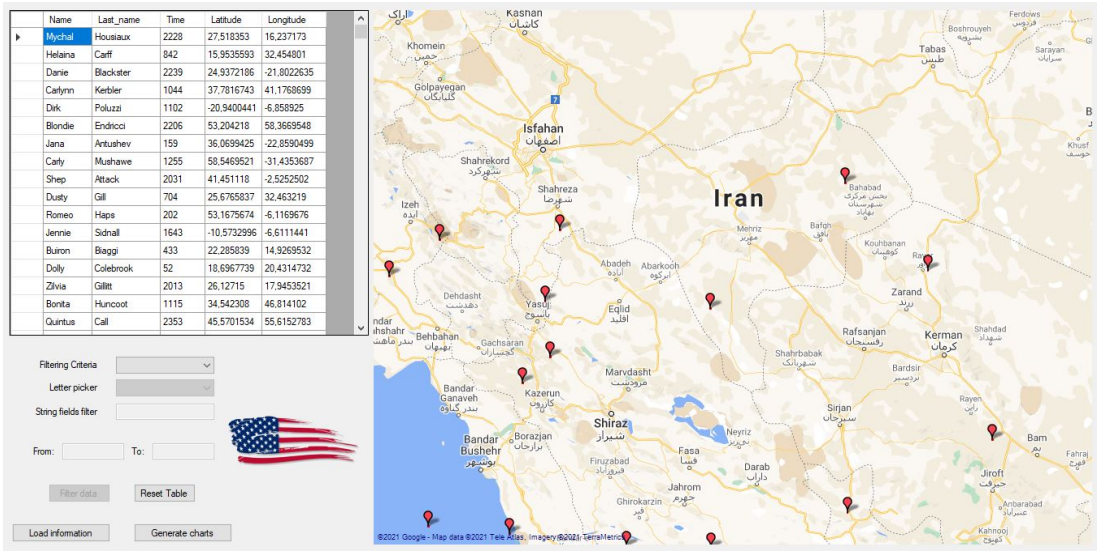


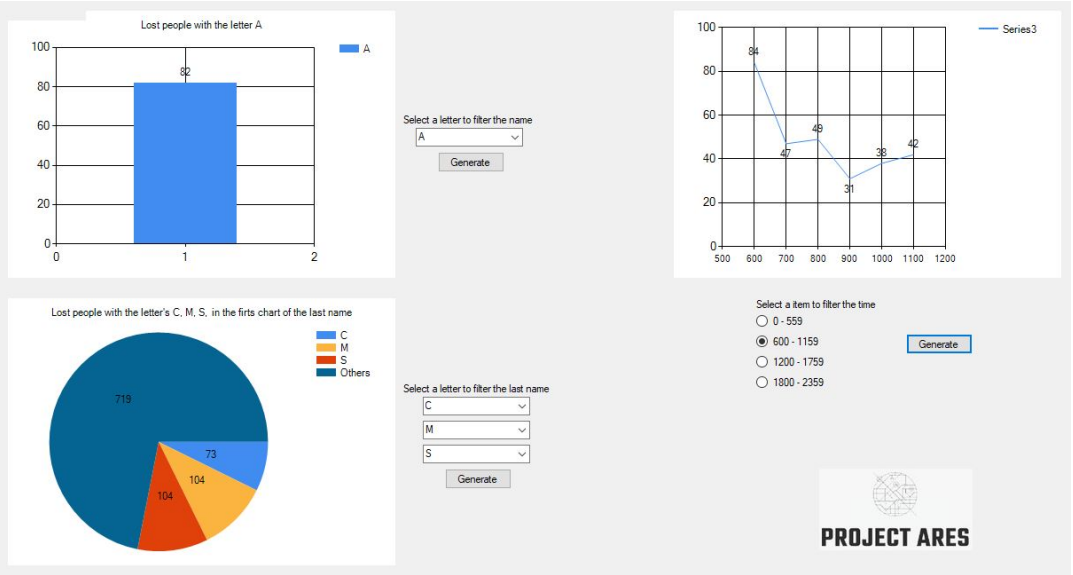


La primera decisión importante que se toma, es el uso de un dataset inmobiliario con un total de 20 mil datos utilizables en el sistema. Con las funciones preliminares de Visual Studio, se espera leer los datos de manera progresiva para ser mostrados en la tabla circundante, y filtrarlos por medio de alguna acción especial.

Además de eso, también se toma la decisión de implementar todo el código tomando en cuenta la orientación a objetos, ya que esta nos brinda grandes beneficios al momento de utilizar la información y/o tener un código mucho más estructurado.

Por último, se toma en cuenta el proyecto-personal-anteriormente-creado para tomar algunas funcionalidades básicas del sistema, como la lectura, filtración y muestra de los datos existentes en un dataset. Con esto, no solo se espera reducir los tiempos de trabajo, sino también mejorar las habilidades computacionales y algoritmicas en cada uno de los integrantes del grupo de trabajo.





Fase 5: Evaluación y selección de la solución

- Para mayor facilidad del usuario, se decide tomar la idea de manejar toda la información relacionada del sistema en una sola ventana principal. Es decir, la interfaz gráfica estará basada en el modelo principal anteriormente presentado: mostrando las pestañas en el lado izquierdo junto a la tabla, y las otras opciones en el lado derecho.

- Para guardar y almacenar la información se decide utilizar la programación orientada a objetos, ya que sería mucho más eficiente al momento de filtrar los datos, ahorrando tiempos de sistema y mermando la complejidad temporal del software. Asimismo, es más competente, ya que obviamente se requiere modificar el dataGridView cuando se realice cualquier tipo de acción.

Por último, se decide utilizar las siguientes herramientas:

VisualStudio: Es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador la creación de software, permitiéndonos desarrollar aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET, algunos lenguajes que podemos encontrar son: Visual Basic, Visual C# y Visual C + +.

GitHub: Es un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código.

VisualParadigm: Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

GoogleDrive: Permite crear carpetas para almacenar y subir archivos de cualquier tipo. Además, logra producir y modificar documentos en línea, manejando diferentes formatos de procesador de textos, planillas de cálculo, editor de diapositivas, etc. Por último, también permite elaborar formularios para encuestas o exámenes.

Fase 6: Preparación de reportes, planos y especificaciones

- Diagrama-de-clases.
- Diagrama-de-objetos.

Fase 7: Implementación del diseño

- Repositorio.