**Identificación del problema:**

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

En el mundo existen muchos tipos de enfermedades, cerebrales, óseas, respiratorias, metales entre otras, pero entre las más graves se encuentran las relacionadas con el corazón. Las enfermedades más comunes relacionadas a problemas en el corazón son la Angina, Infarto de miocardio y la Hipertensión (Henríquez, 2018). Estas enfermedades relacionadas al corazón, también llamadas enfermedades coronarias, deben ser tratadas a tiempo de lo contrario esto puede llevar graves consecuencias incluso pueden terminar en la muerte. En la ciudad de Cleveland en los Estados Unidos un hospital se vio interesado en la problemática de identificar a tiempo estas enfermedades realizó un estudio a una muestra de 304 personas de esta ciudad para identificar si estas contaban con problemas en el corazón o no tomando una serie de datos y exámenes.

En un *dataset* la información se puede encontrar registrada de manera consistente y esta no es la excepción, se cuenta con información valiosa y representativa pero no está estructurada de la manera adecuada, por lo cual si se quisiera realizar un *análisis a la antigua* sería necesario tener un amplio equipo encargado de leer y analizar cada uno de estos casos registrados de manera manual en los que se incluyen más de 13 parámetros (Columnas), en otras palabras ¡sería necesario analizar 4.256 datos uno por uno! Por si esto fuera poco, suponiendo que esta labor tiene éxito, a su vez es necesario realizar un análisis profundo de esta información no basta con solo saber el contenido de la muestra, la verdadera importancia radica en poder inferir si una persona tiene problemas cardiacos o no y además poder filtrarla dependiendo a las necesidades de quien esté haciendo uso de esta información. Sería muy *engorroso* para un médico buscar en el *dataset* un paciente que tenga las mismas características una por una y mirar si este tendrá problemas en el corazón o no, es muy probable que simplemente no lo encuentre y que este se equivoque en la búsqueda. Estas y demás labores de cierto modo son inalcanzables si se hacen manualmente pero un *software* puede resolver en unos pocos segundos, la idea es que mediante la carga de datos y el Machine Learning se pueda facilitar la labor de los médicos a la hora de determinar si una persona tiene o puede tener problemas en el corazón, una labor que debe realizarse con sumo cuidado debido a que se trata de vidas humanas.

**Identificación de necesidades:**

· El hospital necesita que la información de los pacientes sea mostrada en pantalla.

· El hospital necesita que sea posible filtrar dicha información por los diferentes campos que la componen

· El hospital necesita mostrar gráficos a partir de la información de los pacientes

· El hospital necesita clasificar a los nuevos pacientes entre los que tiene problemas de corazón y los que no.

· El hospital necesita almacenar la información de los nuevos pacientes y su clasificación.

**Definición del problema:**

Un hospital de Cleveland ha realizado una investigación sobre personas que tienen y no tienen problemas en el corazón. A cada persona de la muestra se le ha recopilado valiosa información sobre diferentes exámenes coronarios y se ha registrado si los resultados de estos exámenes concluyen en si tiene problemas en el corazón o no. Se requiere ordenar, visualizar mediante tablas y gráficos de manera intuitiva la información, además de poder inferir si una persona tendrá problemas en el corazón de manera concisa y guardar estas en un nuevo *dataset*.

**Recopilación de la información necesaria**

El dolor de pecho es un síntoma común que se presenta en personas con un mal flujo sanguíneo al corazón y que puede significar un riesgo de ataque al corazón (Chen, s.f.) si se realiza un examen este pude dar como resultado 3 valores.

1. Angina

2. Asintomático

3. Anormal

El electrocardiograma es un examen que registra la actividad eléctrica del corazón y permite detectar si una persona ha tenido algún infarto o si tiene arritmias cardiacas este puede generar alguno de los siguientes resultados (Mañero, 2019)

1. Normal

2. Anormal

3. Hipertenso

La angina inducida por ejercicio es si una persona presenta dolor de pecho al someterse a alguna actividad física (Chen, s.f.)

*Slope*: Es una pendiente que representa una tolerancia al ejercicio físico (R S Finkelhor, 1986)

Árboles y Machine Learning: Un árbol de decisión aplicado en el Machine Learning es conjunto de datos que se forman (Valga la redundancia) en forma de árbol dependiendo a una función que el programador asigna, este conjunto de datos tiene muchas utilidades, sus dos principales son poder determinar si un set de datos de entrada pertenece a una determinada clase, esta puede ser clasificada por un resultado binario (Verdadero/Falso) o por más de un resultado. En ciertos casos en los que el árbol no sea muy grande puede permitir visualizar de manera intuitiva cómo está funcionando la clasificación en el *dataset*.

Regresión y Machine Learning: La regresión en el Machine Learning es una técnica en la cual el resultado es un valor numérico en un conjunto infinito de soluciones como estimar que cantidad de productos se va a vender en un supermercado (Heras, 2020)

**Estado del arte**

**HELPSALUD**

Es un proyecto con digitalizar el sector salud a través de soluciones de software aplicando Machine Learning e Inteligencia Artificial, su objetivo final es el siguiente “Proporcionar el mejor diagnóstico, pronóstico o estimación de riesgos, según sea el caso, que sirva de ayuda al personal clínico en la toma de decisiones acerca de sus pacientes permitiendo un tratamiento más personalizado y eficaz” (ITI, 2017)

**Impresión 3D**

Aunque a primera vista la impresión 3D no parezca tener relación con el sector salud y mucho menos con el Machine Learning esto no puede estar más alejado de la realidad, combinando el uso de *big data*, Machine Learning e inteligencia artificial fue posible la impresión 3D del primer cráneo completo en Titania que le fue implantado a una persona de 60 años de edad exitosamente (Aon, 2020)

**Especificación de requerimientos funcionales**

|  |  |
| --- | --- |
| **RF 1** | Tabular información del Dataset |
| **Resumen** | El programa permite cargar y mostrar el *dataset* de los registros de pacientes del hospital. Estos registros están compuestos de: [edad, sexo, dolor de pecho, presión sanguínea en reposo, nivel de colesterol, glucemia en ayunas, electrocardiograma, ritmo cardíaco, angina inducida por ejercicio y un indicador de problemas cardíacos] |
| **Entradas** | **N/A** |
| **Salida** | Se muestra la información tabulada |

|  |  |
| --- | --- |
| **RF 2** | Filtrar la información por campos |
| **Resumen** | El programa permite filtrar los datos mostrados según los diferentes campos del *dataset*. |
| **Entradas** | **Campo y rango de valores de ese campo** |
| **Salida** | Se filtra la información |

|  |  |
| --- | --- |
| **RF 4** | Mostrar gráficos |
| **Resumen** | El programa permite visualizar los 5 tipos gráficos que describen la distribución de la información según los diferentes campos que la componen. Los tipos de gráfico que podrán visualizarse son: barras, dispersión, torta.. |
| **Entradas** | **N/A** |
| **Salida** | Se muestran los gráficos. |

|  |  |
| --- | --- |
| **RF 5** | Entrenar modelo |
| **Resumen** | El programa permite entrenar un modelo de clasificación utilizando el *dataset* inicial. Este modelo permitirá clasificar acertadamente nuevos pacientes para determinar si estos podrían tener problemas cardíacos o no. |
| **Entradas** | **N/A** |
| **Salida** | Se entrena el modelo. |

|  |  |
| --- | --- |
| **RF 5** | Clasificar un nuevo paciente |
| **Resumen** | El programa permite clasificar a un paciente para saber si puede tener problemas cardíacos o no. Esto se realiza utilizando el modelo entrenado y se realiza con una precisión mayor al 95% |
| **Entradas** | **Los valores del paciente para los campos definidos anteriormente.** |
| **Salida** | Se clasifica el paciente en: “Tiene problemas cardíacos” y “No tiene problemas cardíacos” |

|  |  |
| --- | --- |
| **RF 6** | Guardar paciente |
| **Resumen** | El programa permite guardar el registro de un paciente clasificado luego de cargar el *dataset* inicial. |
| **Entradas** | **N/A** |
| **Salida** | Se guarda el nuevo paciente clasificado. |

**Requerimientos no funcionales**

|  |  |
| --- | --- |
| **RNF 1** | Usar técnicas de Machine Learning |
| **Resumen** | El programa debe hacer uso de técnicas de Machine Learning para la clasificación del los pacientes. |

|  |  |
| --- | --- |
| **RNF 2** | Usar dos implementaciones de árboles de decisión |
| **Resumen** | El programa debe hacer uso de dos implementaciones de árboles de decisión: una propia del equipo desarrollador y otra utilizando librerías ya definidas para esta técnica de machine learning. |

**Búsqueda de soluciones creativas**

En este apartado se presentarán varias ideas que se formularon durante la planeación del proyecto independientemente de su conveniencia o no.

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

{Java} 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.

{Bloques} Crear la aplicación en un lenguaje de programación por bloques como lo puede ser App inventor y Scratch.

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

{Python} 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.

{Excel} Crear la aplicación en Excel aplicación que muchas personas usan y conocen hoy en día.

{C#} 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 aplicación del Machine Learning**

{Clasificación Propia} Hacer uso de la técnica de clasificación por clases implementando un árbol de decisión propio asignándole determinadas funciones que también son propias

{Clasificación Papel} Implementar la técnica de clasificación con un árbol de decisión a papel y lápiz y clasificar cada línea del *dataset* de esta manera

{Regresión Propia} Hacer uso de la técnica de regresión implementando un árbol de decisión propio asignándole determinadas funciones que también son propias

{Clasificación Librería} Hacer uso de la técnica de clasificación por clases haciendo uso de una librería de Machine Learning que ofrece el lenguaje de programación

{No Machine} El uso de Machine Learning no es necesario en esta ocasión por lo cual se podría omitir

{Regresión Librería} Hacer uso de la técnica de regresión haciendo uso de una librería de Machine Learning que ofrece el lenguaje de programación

●**Ideas para la carga del *dataset***

{Métodos Propios} Hacer la carga del *dataset* mediante el uso de métodos propios del lenguaje de programación escogido combinado con métodos creados por nosotros.

{Métodos en librerías} Hacer la carga del *dataset* mediante el uso exclusivo de métodos propios del lenguaje de programación escogido.

{Manualmente} 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**

{JavaFx} Contemplar la posibilidad de realizar la interfaz gráfica mediante el uso de la librería gráfica de Javafx.

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

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

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

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

{Sin Interfaz} 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:**

**Java 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 del cual se puede encontrar mucha documentación al respecto para facilitar la realización del programa.

**Bloques 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.

**Adaptar 3:** Tomar una aplicación de código abierto que fue desarrollada por alguien más 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, por ende, esta idea se descarta.

**Python 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 facilitan el desarrollo de la aplicación.

**Excel 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, se descarta esta idea.

**C# 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 esta y tiene muchas librerías referentes a la geolocalización.

**Ideas para la aplicación del Machine Learning**

**Clasificación Propia 1:** Hacer una clasificación propia por clases implementando un árbol de decisión propio es una excelente alternativa debido a que permite hacer un Machine Learning que se ajusta perfectamente a las necesidades del problema planteado permitiendo una personalización excepcional. Por otro lado, permite la comprensión a fondo del Machine Learning.

**Clasificación Papel 2:** Crear un árbol de decisión a papel puede ser muy buena idea a la hora de hacer un pequeño ejemplo para identificar la estrategia y después extrapolarla con una técnica más avanzada, sin embargo, si se quisiera hacer todo el árbol con esta estrategia debido al tamaño del DataSet es totalmente inviable tanto para los desarrolladores como para el usuario final por su complejidad tanto de realización como de uso, se descarta la idea.

**Regresión Propia:** Hacer una regresión propia implementando un árbol de decisión propio es una excelente alternativa. Desafortunadamente el problema en cuestión no cumple con los requisitos para hacer una regresión.

**Clasificación Librería:** Hacer una clasificación haciendo uso de librerías del lenguaje de programación C# con su respectivo árbol de decisión creado es una excelente alternativa, permite hacer un Machine Learning que se ajusta perfectamente a las necesidades del problema generando un balance perfecto entre rendimiento y personalización.

**No Machine:** No hacer uso de Machine no es viable debido a que hacer uso de otras tecnologías limitaría al programa a solo a funcionar en unos casos específicos que muy pocas veces o nunca se dan, por lo cual no sería posible cumplir con el objetivo principal que es determinar si una persona tiene problemas en el corazón o no.

**Regresión Librería:** Hacer una regresión por librería implementando el respectivo árbol de decisión es una excelente alternativa desafortunadamente el problema en cuestión no cumple con los requisitos para hacer una regresión.

**Ideas para la carga del DataSet**

**Métodos Propios:** 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

**Métodos en librerías:** 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.

**Manualmente:** 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:**

**JavaFx:** 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.

**Swing:** 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.

**AWT:** 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 esta, sin embargo, otras tecnologías como lo son Swing y Javafx más modernas han ocasionado que la tecnología AWT se quede obsoleta.

**WPF:** 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 del usuario final.

**Windows Forms:** 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.

**Sin Interfaz:** 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**

**Conocimiento sobre el tema:**Hace referencia a las destrezas generales que tienen los integrantes del equipo para poder llevar a cabo la realización alguna de las alternativas presentada en la rúbrica. La calificación es numérica y se realiza de 0 a 5.

**Necesidad o valor para el cliente:** Hace referencia al valor agregado o beneficios que la alternativa genera al cliente o usuario final. La calificación es numérica se realiza de 0 a 5.

**Facilidad de desarrollo:** Hace referencia a que tan rápido se le puede dar finalización a la alternativa en cuestión, si el cliente final requiere que la aplicación se termine rápidamente esta adquirirá más valor. La calificación es numérica y se realiza de 0 a 5

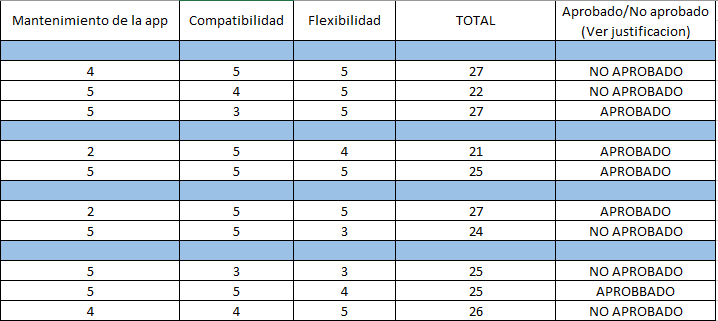
**Mantenimiento de la App:**Hace referencia a la facilidad que le tomara al desarrollador hacer una modificación a la alternativa después entregado el producto final. La calificación es numérica y se realiza de 0 a 5

**Compatibilidad:**Hace referencia al posicionamiento que tiene el uso de una alternativa, en otras palabras, si las herramientas que usa la alternativa se han usado en proyectos diferentes a este en repetidas ocasiones esta tenderá a tener un abanico de compatibilidad mayor tanto con dispositivos como con otras herramientas que respecta al lenguaje de programación. La calificación es numérica y se realiza de 0 a 5.

**Flexibilidad:**Hace referencia a la capacidad de personalización de la alternativa al problema que se está tratando, es decir, si una alternativa no hace uso de librerías del lenguaje de programación y esta es diseñada para el problema en cuestión la calificación tenderá al alta. La calificación es numérica y se realiza de 0 a 5.

**Nuevos Conocimientos:** Hace referencia al valor de aprendizaje como desarrolladores que conlleva implementar una alternativa antes que otra. La calificación no es numérica y se usa en caso de que los desarrolladores quieran incursionar en su aprendizaje activo y a la vez se pueda cumplir con todas las necesidades del cliente o usuario de manera óptima. Si una alternativa cuenta con este apartado esta será aprobada.





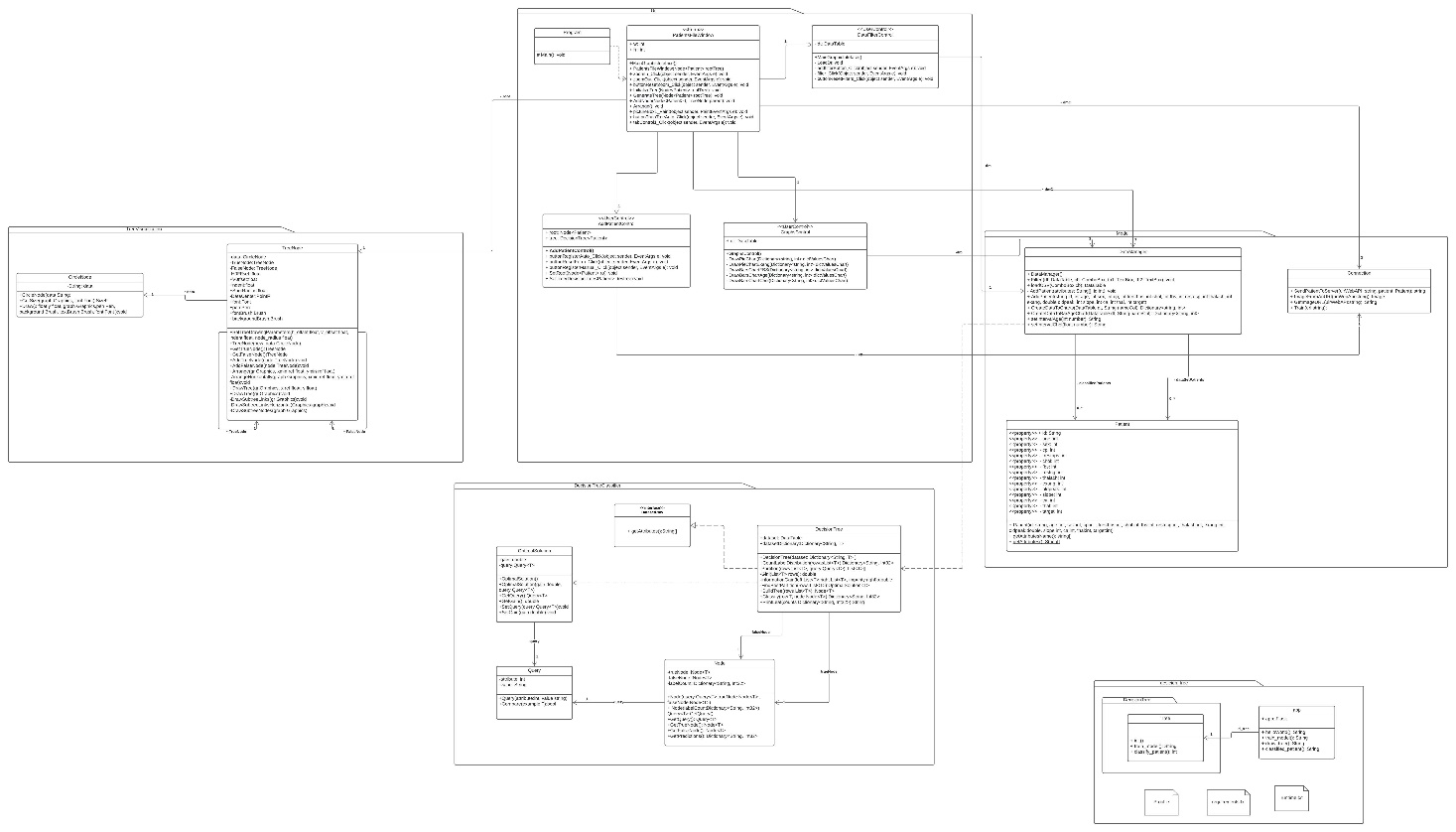
**Lenguaje de programación:** Como lenguaje de programación se optó por hacer uso de C#, aunque este obtuvo la misma calificación que el lenguaje de programación Java se decidió tener por aparte de la rúbrica en este segmento el criterio de calificación “Nuevos Conocimientos” y C# obtuvo un alto valor 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.

**Uso de Machine Learning:** Para el uso del Machine Learning aplicado a la calificación se optó por aprobar las dos ideas que se consideraron viables, que fueron hacer uso de clasificación propia junto con árbol de decisión propio y la clasificación por librerías con su respectivo árbol de decisión. Aunque ambas ideas cumplen la misma función y la calificación de una es mayor que la otra, se consideró el criterio de evaluación “Nuevos Conocimientos” ya que implementar la clasificación de manera propia permitirá una comprensión exhaustiva del Machine Learning aplicado al filtrado de información.

**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 permite obtener la flexibilidad requerida por el usuario final. Por otro lado, esta solución fue la que obtuvo la 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.

**Diagrama de clases**



**Síntesis reflexiva**

El proyecto cumplió con todas los requerimientos y expectativas iniciales e incluso se superaron gracias a “pensar fuera de la caja” aplicando tecnologías y lenguajes de programación que en ningún momento se solicitaban pero que representaban un aumento considerable en el rendimiento y mantenimiento del programa, por otra parte, esto aporto importantes conocimientos que serán muy útiles tanto para futuros semestres como para el ámbito profesional, aun así, el proyecto no queda exento de muchas mejoras que con un poco mas de tiempo se pudieron haber aplicado, por un lado el *dataset* escogido pudo ser mejor ya que por falta de registros la precisión del programa pudo ser mucho mejor. Además, la interfaz de usuario, aunque cumple con todas las necesidades, es muy mejorable en el aspecto gráfico, esta podría lucir mejor.

En cuanto a la implementación, no hay mucho que mencionar, el código es completamente legible y fácil de trabajar.

# **Referencias**

Aon. (04 de Agosto de 2020). *NOA*. Obtenido de https://noa.aon.es/machine-learning-aplicaciones-salud/

Chen, M. A. (s.f.). *MedlinePlus*. Obtenido de https://medlineplus.gov/ency/patientinstructions/000775.htm

Henríquez, K. (30 de Septiembre de 2018). *elsalvador.com*. Obtenido de https://www.elsalvador.com/vida/salud/7-enfermedades-cardiacas-mas-comunes-a-nivel-mundial-2/523277/2018/

Heras, J. M. (29 de Septiembre de 2020). *IArtificial.net*. Obtenido de https://www.iartificial.net/clasificacion-o-regresion/#Regresion

ITI. (01 de Enero de 2017). *ITI Investigate To Innovate*. Obtenido de https://www.iti.es/proyectosidi/helpsalud-machine-learning-salud/

Mañero, M. R. (14 de Agosto de 2019). *Fundación Española del corazon* .

R S Finkelhor, K. E. (Agosto de 1986). *Pubmed.gov*. Obtenido de https://pubmed.ncbi.nlm.nih.gov/3739881/

Data set:<https://www.kaggle.com/ronitf/heart-disease-uci>