**David Alejandro López Torres 17300155 8D1 – Proyecto 01**

Con base a todos los elementos expresados en las investigaciones del parcial, se propone desarrollar un estudio completo del proyecto de titulación propuesto por parte de mí equipo: Andrés Huerta Vásquez y Daniel Tejeda para la carrera de desarrollo de Software, poniendo especial énfasis en la aplicación de los conceptos de los que se ha hablado en los otros 4 documentos. El proyecto se denomina: “En desarrollo de una PWA (Progressive Web App) para la solución de EDO (Ecuaciones Diferenciales Ordinarias)”. El proyecto se considera **innovador** porque pretende solucionar un problema de una manera muy diferente a como se ha atacado por parte de otras soluciones análogas, además de que la implementación final resulta ser una agrupación inteligente de diferentes elementos tecnológicos que constituyen en su totalidad al proyecto, sumando algunas partes importantes que nos corresponde desarrollar. La propuesta se desenvuelve en el campo académico ya sea visto desde el lado del estudiante o del investigador. Por la diferencia entre estos dos usuarios, el sistema se modelaría para adaptarse a el conocimiento y necesidad de cada usuario (estudiantes e investigadores). En cuanto a los recursos necesarios, se tienen las siguientes notas: El sistema está pensado para funcionar como una PWA (Progressive Web App), de modo que el usuario final puede acceder desde cualquier dispositivo móvil (al menos la mayoría), ordenador de escritorio o a una versión web. Para el uso adecuado del sistema se requiere que el usuario tenga conexión a internet durante el procesamiento de la ecuación. Podemos plantear la problemática que se pretende resolver por medio de un desarrollar emprendedor con los siguientes puntos: A) Muchos estudiantes de carreras de ingeniería o ciencias exactas suelen verse en la necesidad de resolver ecuaciones diferenciales para una gran parte de sus tareas, lo cual es una carga adicional de estrés en la vida diaria de los estudiantes. B) Los estudiantes se ven frustrados porque ante ciertos problemas que involucran la resolución de ecuaciones diferenciales se sientes incapaces de dar con una solución correcta. C) Actualmente solo algunas plataformas y sistemas complejos son capaces de resolver una gran gama de ecuaciones diferenciales que podrían presentarse en la carrera de los estudiantes de ingeniería o ciencias exactas, así como algunos investigadores. El acceso a estas plataformas y sistemas suele llegar a representar una barrera para dar con la solución de la ecuación diferencial.

En base a los resultados que tienen las aplicaciones similares al sistema planteado, así como el poco gasto que se requiere para su realización son indicadores de que el proyecto es viable. Además, la existencia de un equipo con personas capaces y dispuestas a desarrollarlo junto con un tiempo de entrega considerable favorecen aún más a la viabilidad del proyecto. El proyecto es aplicable porque puede ser utilizado en el día a día de los estudiantes e investigadores cada que sea necesario resolver una ecuación diferencial (que pasa a menudo en el campo académico superior). El proyecto es accesible por ofrecer flexibilidad a los usuarios en la manera de ingresar la ecuación diferencial, así como la plataforma en donde interactúa con el sistema. Los múltiples medios de entrada del sistema pueden sortear algunas de las condiciones especiales de los usuarios del sistema. Por otro lado, las diferentes presentaciones del sistema en función del conocimiento en el área por parte de los usuarios generan que el proyecto sea accesible de una mejor manera para cada tipo de usuario. La intención primordial del proyecto es precisamente que sea usable por parte de estudiantes e investigadores para resolver ecuaciones diferenciales de manera eficiente en su día a día. La solución planteada ofrece una opción sencilla de usar, pero también que es capaz de atacar específicamente la necesidad de resolver ecuaciones diferenciales de manera práctica.

La solución es en parte más innovadora que las otras por la amplia gama de tecnologías que implementa, entre las cuales podemos rescatar: Cliente: Diseño del sistema: Con Java-XML para Android; Kivy-Python para iOS y otros dispositivos móviles; JavaScript-HTML-CSS para una versión web; C#-ASP para una versión de escritorio. Se puede utilizar un framework como Angular o React para facilitar el diseño de la versión web. Módulo de recepción: Cámara y sensores de los dispositivos móviles; periféricos de equipos de escritorio. La gestión sería por medio de Java-XML para Android; Kivy-Python para iOS y otros dispositivos móviles; y JavaScript-HTML-CSS para una versión web; C# para una versión de escritorio. Módulo de interpretación: API de reconocimiento de imagen (OCR). Google Vision es la opción más viable. Módulo de clasificación: Java para Android, Python para iOS y otros dispositivos móviles, JavaScript para una versión web, C# para una versión de escritorio. Módulo de despacho: JSON (Para dar formato estándar a la salida) y Conectores con el servidor según sea el caso. Servidor: Librerías de cálculo avanzado y manejo simbólico de expresiones de Python (SymPy), en concreto el módulo de ODE a manera de referencia y módulo de integración, derivación y manejo de expresiones simbólicas.

La integración de todas estas tecnologías tiene vuelven al proyecto nuevo en el mercado y es competitivo debido a su enfoque completamente Open Source, es decir, de código abierto, por lo que los costes de financiamiento y de distribución serían nulos. Es interesante analizar de manera financiera lo que ocurre con la **financiación** de un proyecto de naturaleza Open Source, ya que se basa en una cultura de bazar y no de catedral (los bienes son comunes y es la misma gente la que participa en su mejoramiento y valoración, y no solo un grupo selecto de personas). Para plantear la idea del financiamiento, podemos estudiar lo que menciona Eric Raymond en su obra “El Caldero Mágico”: Habiendo echado un ojo escéptico en un modelo prevalente, veamos si podemos construir una explicación económica afinada acerca de lo que hace a la cooperación Open Source sostenible. Esta es una cuestión que requiere ser examinada en un par de niveles diferentes. En un nivel, necesitamos explicar el comportamiento de los individuos que contribuyen a los proyectos Open Source; en otro, necesitamos entender las fuerzas económicas que sostienen la cooperación en proyectos Open Source como Linux o Apache. De nuevo, primero debemos demoler un modelo popular que interfiere con nuestra comprensión. Sobre cada intento de explicar el comportamiento cooperativo se cierne la sombra de la “tragedia de los comunes” de Garret Hardin. Hardin nos pide que imaginemos un campo que es propiedad común de un pueblo de campesinos, que llevan a su ganado a pastar en él. Pero el pastar degrada el campo, arrancando la hierba y dejando trozos de barro, donde la hierba tarda en volver a crecer. Si no hay ninguna política acordada (e impuesta) para asignar derechos de pasto que eviten la sobreexplotación, todos los incentivos empujan a llevar tanto ganado y tan rápidamente como sea posible, intentando extraer el máximo valor antes de que el campo común se degrade hasta convertirse en un mar de barro. La mayoría de la gente tiene un modelo intuitivo del comportamiento cooperativo que funciona de manera muy semejante a este. La tragedia de los comunes en realidad surge de dos problemas ligados, uno de sobreexplotación y otro de infra provisión. Por el lado de la demanda, la situación de los comunes anima a una carrera hasta el fin hacia la sobreexplotación: lo que los economistas llaman un problema de bien público congestionado. Por el lado de la oferta, los comunes recompensan el comportamiento egoísta, eliminando o disminuyendo los incentivos para que los actores individuales inviertan en desarrollar más pastos. La tragedia de los comunes predice sólo tres posibles resultados. Uno es el mar de barro. Otro es que algún actor con poder coercitivo imponga una política de asignaciones por el bien del pueblo (la solución comunista). El tercero es que el campo común se deshaga cuando los habitantes del pueblo vallen parcelas que pueden defender y gestionar de manera sostenible (la solución de los derechos de propiedad).Cuando la gente aplica este modelo a la cooperación Open Source, esperan que sea inestable y que tenga una vida corta. Como no hay una manera evidente de imponer una política de asignación para el tiempo de programador en Internet, este modelo conduce directamente a la predicción de que el común se romperá, varios fragmentos de software se volverán cerrados y decrecerá rápidamente la cantidad de trabajo dedicada al fondo común.

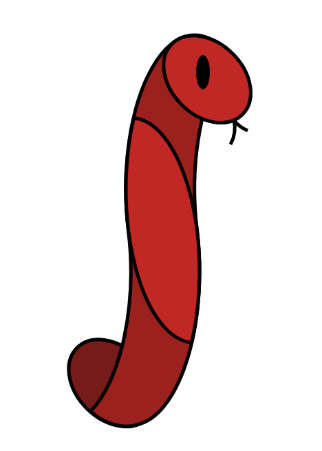
En realidad, es empíricamente claro que la tendencia es opuesta a esto. La tendencia en amplitud y volumen de desarrollo Open Source puede medirse por las remisiones diarias a Metalab y SourceForge (los sitios líderes en código para Linux) o los anuncios por día en freshmeat.net (un sitio dedicado a anunciar nuevas versiones de software). El volumen en ambos está creciendo constante y rápidamente. Claramente hay un aspecto crítico en el que el modelo de la “tragedia de los comunes” falla a la hora de interpretar lo que está pasando en realidad. Parte de la respuesta, por supuesto, yace en el hecho de que el uso del software no decrementa su valor. Sin duda, el uso extendido del software Open Source tiende a incrementar su valor, cuando los usuarios aportan sus propias mejoras (parches en el código). En este común inverso, la hierba crece más alta cuando se pasta. Que este bien público no pueda degradarse por el sobreuso da cuenta de la mitad de la tragedia de Hardin, el problema de los bienes públicos congestionados. No explica por qué el Open Source no sufre infra provisión. ¿Por qué la gente que sabe que la comunidad Open Source existe no muestra universalmente un comportamiento egoísta, esperando a que otros hagan el trabajo que necesitan o(si hacen el trabajo por sí mismos) sin molestarse en contribuir con su trabajo al común? Parte de la respuesta radical en el hecho de que la gente no necesita simplemente soluciones, necesita soluciones a tiempo. Raramente es posible predecir cuándo otro terminará una pieza dada de trabajo necesario. Si el pago por arreglar un bug o añadir una funcionalidad es suficiente para cualquier contribuyente potencial, esa persona lo hará (y en ese punto el hecho de que los demás sean egoístas se hace irrelevante).

Un hecho clave que la distinción entre uso y valor nos permite observar es que sólo el valor de venta está amenazado por el cambio de cerrado a abierto; el valor de uso no. Si el valor de uso más que el valor de venta es en realidad el mayor conductor del desarrollo de software, y (como se argumentó en La Catedral y el Bazar) el desarrollo Open Source es en realidad más eficaz y eficiente que el desarrollo cerrado, entonces debemos esperar encontrar circunstanciasen las que sólo el valor de uso esperado baste para financiar el desarrollo open Source. Y de hecho no es difícil identificar al menos dos modelos importantes en los que los salarios de desarrolladores a tiempo completo para proyectos open Source están financiados estrictamente a partir del valor de uso. El Open Source hace bastante difícil capturar el valor de venta directo del software. La dificultad noes técnica; el código fuente no es ni más ni menos fácil de copiar que el binario, y la imposición de las leyes de copyright y licencia para capturar el valor de venta no tiene que ser necesariamente más difícil para los productos open Source que para los cerrados. La dificultad radica más bien en la naturaleza del contrato social que soporta el desarrollo open Source. Por tres razones que se refuerzan entre sí, las principales licencias open Source prohíben la mayoría de tipos de restricciones en el uso, redistribución y modificación que podrían facilitar la captura de beneficios por la venta directa. Para entender estas razones, debemos examinar el contexto social en el que las licencias evolucionaron: la cultura hacker de Internet.

A pesar de los mitos acerca de la cultura hacker que todavía están muy extendidos fuera de él, ninguna de estas razones tiene que ver con la hostilidad hacia el mercado. Mientras que una minoría de hackers sin duda permanece hostil a la motivación del beneficio, la disposición general de la comunidad para cooperar con paquetes comerciales Linux como Red Hat, SuSE y Caldera demuestra que la mayoría de los hackers trabajarán felizmente con el mundo corporativo cuando sirva a sus intereses. Las verdaderas razones de los hackers para recelar de las licencias de captura directa del beneficio son más sutiles e interesantes. Una razón tiene que ver con la simetría. Mientras que la mayoría de los desarrolladores open Source no tiene intrínsecamente nada que objetar a que otros se beneficien de sus regalos, también demanda que nadie (con la posible excepción del creador de una pieza de código) esté en una posición privilegiada para extraer beneficios. J Cualquier Hacker está dispuesto a que Una SA se beneficie vendiendo su software o parches, pero sólo si el mismo JCH puede también hacerlo. Otra tiene que ver con las consecuencias no deseadas. Los hackers han observado que las licencias que incluyen restricciones y tarifas para uso comercial o venta (la manera más normal de intentar recapturar el valor directo de venta, y que en principio parece razonable) tienen serios efectos de congelación. Uno específico es arrojar una sombra legal sobre actividades como la redistribución en recopilaciones baratas en CD-ROM, que idealmente tenderíamos a fomentar. Más generalmente, las restricciones en el uso/venta/modificación/distribución (y otras complicaciones en el licenciamiento) imponen un sobrecosto para el seguimiento de la conformidad y (según aumenta el número de paquetes que maneja una persona) una explosión combinatoria de la incertidumbre percibida y los riesgos legales potenciales. Este resultado se considera perjudicial, y por tanto hay una fuerte presión social para mantener las licencias simples y libres de restricciones.

La razón última y más crítica tiene que ver con la preservación de la dinámica de revisión entre pares y cultura del regalo descrita en Colonizando la Noosfera del mismo autor. Las restricciones de licencia diseñadas para proteger la propiedad intelectual o capturar el valor directo de venta a menudo tienen el efecto de hacer legalmente imposible ramificar el proyecto. Este es el caso, por ejemplo, con las licencias de Sun llamadas “Fuentes de la Comunidad” para Jini y Java. Aunque la bifurcación se ve con recelo y se considera el último recurso (por razones discutidas en profundidad en Colonizando la Noosfera), se considera críticamente importante que ese último recurso esté presente en caso de incompetencia o defección del mantenedor (por ejemplo, para pasar a una licencia cerrada).

En cuanto a la **mercadotecnia** que se requiere para el proyecto, el diseño de marca está fuertemente relacionado con el sector al que se destina el proyecto y los objetivos que tiene, los cuales son: A) Estudiantes de ingenierías y ciencias exactas, B) Ingenieros egresados e investigadores en ciencias exactas. Y el objetivo: Evaluar el desempeño de una aplicación, que resuelve ecuaciones diferenciales por medio de métodos numéricos y que implementa redes neuronales para generar un lector óptico, como reductor del estrés académico del universitario; con objetivos específicos: • Concretar un algoritmo numérico capaz de resolver ecuaciones diferenciales por medio de una consulta bibliográfica • Definir un algoritmo que utilice redes neuronales para convertir la información contenida en una imagen en texto procesable • Traducir el texto obtenido en elementos matemáticos por medio de un algoritmo para identificar el tipo de ecuación diferencial al que se hace referencia • Diseñar una interfaz gráfica por medio de Java para que el usuario y el sistema intercambien información de manera clara y natural. De este modo, se busca un diseño de marca flexible y que sea amigable para la comunidad juvenil, así como para los diferentes grupos de investigadores que puedan realizar. Se adjunto un boceto del principal referente visual de la marca a desarrollar con la intención de verificar que se cumplen estos puntos de flexibilidad y al mismo tiempo estar acorde al desarrollo que se busca del proyecto:

El diseño es de una serpiente debido a que la principal tecnología a utilizar será el lenguaje de programación Python (Serpiente Pitón en inglés). Además, la forma de la serpiente hace alusión a uno de los símbolos matemáticos más importantes de nuestra aplicación: el símbolo integral, operador matemático con el cual es posible llegar a la solución de ecuaciones diferenciales ordinarias, tal como se planteó en un primer lugar del proyecto. Sobre la difusión que se pretende llevar a cabo para que sea utilizado y comenzar a generar ingresos mediante el modelo Open Source presentado previamente se utilizará una propagación por medio de una campaña publicitaria en redes sociales y por medio de una serie de vídeos en la plataforma de YouTube para generar contenido complementario para el uso y desarrollo de la aplicación. En dicha propaganda se busca la interacción de más personas con la aplicación para lograr un mayor volumen de desarrollo y de uso de la aplicación, lo necesario para el modelo Open Source planteado previamente.

Finalmente, para la implementación **ergonómica** del proyecto se realizó una calendarización por medio de una Carta de Gantt para mostrar una correcta organización y distribución de como se llevará a cabo el desarrollo del proyecto a lo largo del período Marzo hasta junio. Además, se definieron los elementos específicos que componen al proyecto mediante la redacción de un listado de requerimientos funcionales y no funcionales que describen las diferentes características técnicas que componen al proyecto y se establece una asignación en el desarrollo de cada uno de estos requerimientos entre los integrantes del equipo.

Con el planteamiento de estos elementos del proyecto del desarrollo de la PWA para resolver EDO queda claro que el proyecto entra de unos parámetros que garantizan su viabilidad, ya que cuenta con los elementos mínimos que, al ser combinados, generan una base sólida para garantizar el éxito parcial del proyecto, (pues un éxito completo depende de más factores que están lejos de elementos que puedan controlarse por los integrantes del equipo. Es de suma importancia establecer cada uno de estos elementos para el desarrollo del proyecto, debido a que forman los cimientos para nutrirlo y garantizar su avance y desarrollo. Los diferentes elementos que componen esta primera etapa del proyecto son fundamentales para comprender la naturaleza financiera y administrativa del proyecto, así como sus bases en los principios de innovación tecnológica y de desarrollo emprendedor. Es importante aclarar que es gracias a estos elementos que el proyecto entra de unos parámetros que garantizan su viabilidad, ya que cuenta con los elementos mínimos que, al ser combinados, generan una base sólida para garantizar el éxito parcial del proyecto, (pues un éxito completo depende de más factores que están lejos de elementos que puedan controlarse por los integrantes del equipo. Además, podemos recalcar que tener la planeación de estos elementos en una etapa temprana del proyecto permite considerar algunas adversidades antes de afrontarlas; es decir, nos sirve como un mapa de pronóstico de riesgos que puede llegar a tener el proyecto en algún momento dado, por lo que representan una herramienta más allá de los elementos indicadores que contienen de la viabilidad, si no para prevención y reajuste de la orientación que puede tomar el proyecto para ajustarse dentro de los parámetros ergonómicos y financieros que eso representa, así como garantizar una respuesta positiva mediante un buen estudio del mercado Open Source.