ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

DESARROLLO DE UNA APLICACIÓN PARA EL APRENDIZAJE DE PROGRAMACION ESTRUCTURADA EN C

APLICACIONES DE SOFTWARE EDUCATIVO Y EMPRESARIAL

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

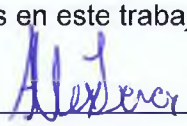
LUIS ALEXANDER LARA BEDON
luis.lara01@epn.edu.ec

DIRECTOR: DRA. ANA MARÍA ZAMBRANO VIZUETE
ana.zambrano@epn.edu.ec

DMQ, marzo 2023

CERTIFICACIONES

Yo, Luis Alexander Lara Bedón, afirmo que el trabajo de integración curricular descrito en este documento es original y ha sido elaborado por mí. Además, declaro que este trabajo no ha sido presentado previamente para obtener ningún grado o calificación en ninguna institución. Asimismo, he asegurado que he consultado todas las referencias bibliográficas mencionadas en este trabajo.



Luis Alexander Lara Bedón

Certifico que el presente trabajo de integración curricular fue desarrollado por Luis Alexander Lara Bedón, bajo mi supervisión.

Dra. Ana María Zambrano Vizuete

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

SR. LUIS ALEXANDER LARA BEDÓN

DRA. ANA MARÍA ZAMBRANO VIZUETE

DEDICATORIA

A mis padres Sonia y Luis, en especial dedico con todo mi corazón este trabajo a mi madre, pues sin ella no lo habría logrado. Sus enseñanzas y apoyo incondicional me han llevado a ser una mejor persona. A mi padre, por su entrega, sacrificio y amor por nuestra familia.

Luis Alexander Lara Bedón

AGRADECIMIENTO

Agradezco a Dios y a mi familia, quienes siempre han creído en mí, dándome ejemplo de superación, humildad y sacrificio; enseñándome a valorar todo lo que tengo, sin dejar de lado las ganas de mejorar.

Brindo un agradecimiento especial a la Dra. Ana María Zambrano, directora de este trabajo, por darme las directrices correctas, brindarme su apoyo y permitir que este trabajo se terminara a buen recaudo.

Luis Alexander Lara Bedón

ÍNDICE DE CONTENIDOS

RESUMEN

Este Trabajo de Integración Curricular tiene como propósito diseñar e implementar una aplicación web para facilitar el aprendizaje de la programación estructurada (lenguaje C/C++) a través de la programación por bloques. La aplicación usa la biblioteca *React Flow* para crear diagramas de flujo, que son la base de este proyecto. El usuario tendrá acceso a los bloques necesarios cuando sean requeridos. La aplicación también permite al usuario obtener el código en lenguaje C/C++, simplificando el proceso y mejorando la forma en que se escribe el código para evitar errores comunes al programar. Los bloques diseñados para el proyecto incluyen: *IN, DEFINE, FOR, OUT, IF y OPERATION*. Estos bloques se usan para controlar entradas y salidas, como sensores, actuadores y pantallas, así como para controlar la ejecución de un programa, como bucles y sentencias condicionales. Estos bloques se usan para crear programas complejos y ayudan a los principiantes a aprender a programar de manera intuitiva y sencilla.

En el Capítulo 1 se estudian las herramientas a utilizar en la aplicación que se construye usando el patrón Modelo Vista Controlador (*MVC*), *React* JS, *React Flow*, *Bootstrap, Child Proccess* y G ++. Se usa Visual Studio Code para conectar la aplicación con el procesador del equipo.

En el Capítulo 2 se lleva a cabo la obtención de los requerimientos que necesita cumplir la aplicación y como desarrollar la aplicación, tanto en su parte operativa como en la interfaz del usuario.

Y finalmente, en el Capítulo 3 se analiza que la aplicación cumpla con los objetivos y requerimientos propuestos en base a la utilización de esta, por medio de un grupo de personas de prueba y se analizan los resultados obtenidos para conseguir las conclusiones y recomendaciones acerca del proyecto desarrollado.

PALABRAS CLAVE: MVC, Framework, Microsoft Visual Studio, *React*, Node JS, Kanban.

ABSTRACT

This Curricular Integration Project aims to design and implement a web application to facilitate learning structured programming (C/C++ language) through block programming. The application uses the React Flow library to create flowcharts, which form the basis of this project. The user will have access to the necessary blocks when required. The application also allows the user to obtain code in C/C++ language, simplifying the process and improving the way code is written to avoid common programming errors. The blocks designed for the project include: IN, DEFINE, FOR, OUT, IF, and OPERATION. These blocks are used to control inputs and outputs, such as sensors, actuators, and displays, as well as to control program execution, such as loops and conditional statements. These blocks are used to create complex programs and help beginners learn to program intuitively and simply.

Chapter 1 studies the tools to be used in the application that is built using the Model-View-Controller (MVC) pattern, React JS, React Flow, Bootstrap, Child Process, and G++. Visual Studio Code is used to connect the application to the computer processor.

Chapter 2 carries out the acquisition of the requirements that the application needs to meet and how to develop the application, both in its operational part and in the user interface.

Finally, in Chapter 3, it is analyzed whether the application meets the objectives and requirements proposed based on its use by a group of test users, and the results obtained are analyzed to obtain conclusions and recommendations about the developed project.

KEYWORDS: MVC, Framework, Microsoft Visual Studio, *React*, Node JS, Kanban.

1. INTRODUCCIÓN

Con el paso de los años el uso de la programación se ha ido incrementando, siendo incluso que no solo las personas que trabajan en este medio y que utilizan diferentes lenguajes de programación en sus labores deban ser las únicas que necesitan aprender la lógica de un lenguaje de programación. Aprender a programar puede ser comparado con aprender un nuevo idioma, en el cual se estimula el pensamiento lógico, estructurado, y también la creatividad al momento de resolver un ejercicio planteado. Ejercicios que pueden poseer múltiples caminos de resolución y que, dependiendo de la lógica utilizada en el desarrollo, pueden convertirse en algo muy extenso y complicado, todo con base en la forma en que cada persona evalúa el problema y las ideas que surgen al momento de resolverlo. Además, aprender a programar aumenta el valor de cada individuo en el mundo laboral pues otorga la posibilidad de mejorar sus aptitudes, utilizando la lógica y conocimientos adquiridos.

Un programa es un conjunto de instrucciones para realizar una tarea, que utilizando la sintaxis de cada lenguaje de programación realiza una petición al procesador de la computadora, con el fin de obtener una respuesta; dependiendo de si la estructura está bien implementada, la respuesta será la esperada o no.

El escribir código puede resultar una tarea compleja y confusa para algunas personas, ya que se necesita conocer la sintaxis de cada comando, y con el fin de evitar esta confusión al momento de programar, se desarrolla una aplicación que permita en base a diagramas de flujo el poder programar estimulando la creatividad y lógica del programador; adicionalmente facultando al mismo el obtener el código en un lenguaje de programación, simplificando el proceso, mejorando la forma en que se debe escribir el código y evitando errores comunes que se cometen al momento de programar, como pueden ser: olvidar cerrar funciones, concluir de manera correcta una sentencia (es decir “;” al final de la misma), e identar de forma incorrecta el código[1].

Las personas que son principiantes en la programación pueden resultar beneficiadas por la programación en bloques, ya que la aplicación desarrollada permitirá que se programe de una forma intuitiva, teniendo a disposición bloques para su adición cuando sean necesarios según el ejercicio. Al igual que la programación “textual o formal” se tienen elementos como: variables, bucles, declaraciones, fórmulas, iteraciones, funciones del lenguaje, entre otras [1].

[1] La identación de código es la adición de espacios o tabulaciones al inicio de las líneas para mejorar su legibilidad y organización.

La programación por bloques es una forma de programación visual que utiliza bloques de construcción, cada uno con una función específica, para crear algoritmos. Estos bloques se enganchan juntos para representar la estructura del programa y son una forma simple y fácil de aprender para los principiantes de la programación. Los diagramas de flujo son una herramienta valiosa para apoyar esta metodología ya que permiten representar el flujo del programa de una forma sencilla e intuitiva.

Una de las principales ventajas de la programación por bloques es que es fácil de aprender para los principiantes. Es una forma simple y concreta de representar un algoritmo, lo que lo hace accesible para niños y adultos que no tienen experiencia previa en programación. El uso de bloques de construcción en lugar de líneas de código también reduce la posibilidad de cometer errores sintácticos, lo que facilita la corrección de problemas y el aprendizaje del lenguaje.

Además, la programación por bloques con diagramas de flujo es una herramienta poderosa para el aprendizaje de la lógica de programación. Los bloques de construcción permiten a los estudiantes aprender los conceptos básicos de la programación, como el uso de variables para la manipulación de datos, la estructura de condicionales y bucles, y la manipulación de datos, de una manera concreta y visual.

Un bloque de programación es exactamente una figura geométrica que va a contener código preestablecido en el cual solo se tendrá que rellenar con las propiedades necesarias tales como: nombre entidad, valor, su tipo y el conjunto de instrucciones matemáticas o lógicas.

Con el desarrollo de este prototipo se busca apoyar el proceso de enseñanza y aprendizaje de la asignatura de Programación (USFQ), a través del desarrollo de un entorno más amigable y sencillo de entender. Para lograr este objetivo, se ha optado por desarrollar una aplicación web basada en el *Framework React Js* [2] y se utilizarán librerías, como la biblioteca "*React Flow*" [3] [4] que permite crear diagramas de flujo, los cuales generarán el entorno de programación intuitivo en el que el usuario tiene acceso a los bloques necesarios en el momento que los requiere

1.1. OBJETIVO GENERAL

Diseñar e implementar una aplicación web para facilitar el aprendizaje de programación estructurada (lenguaje C/C++), mediante programación por bloques.

1.2. OBJETIVOS ESPECÍFICOS

- Analizar las herramientas necesarias y la metodología que se utilizarán en el desarrollo del prototipo.
- Diseñar la aplicación web con tres módulos: Aprendizaje, Programación por bloques, Manual de Usuario.
- Implementar el prototipo de acuerdo con el diseño realizado.
- Analizar la funcionalidad a través de la validación del cumplimiento de los requisitos funcionales y no funcionales.

1.3. ALCANCE

El presente Trabajo de Integración Curricular tiene como objetivo el desarrollo de un prototipo de aplicativo web que permita la enseñanza de manera más intuitiva y ágil de la asignatura de “Programación”, que a personas principiantes les puede resultar un poco molesta y confusa, incluso complicada, puesto que pueden olvidar pequeños detalles y características de notación que en un lenguaje de programación convencional son indispensables; errores que pueden ser: un punto y coma, la identación, el cierre o finalización de un comando, u otro carácter necesario para ciertos lenguajes de programación. Esto es algo que se evita al momento de utilizar programación por bloques, pues el bloque ya está preprogramado para que solo se necesite rellenar los campos necesarios con la lógica y las variables que se haya pensado para resolver el programa.

La aplicación contará con una página principal o índice donde puede elegir entre los siguientes módulos:

- Módulo de Programación.
- Módulo de Aprendizaje.
- Manual de Usuario.

Con este prototipo se conseguirá construir gráficamente un programa utilizando Diagramas de Flujo, así como obtener el código en lenguaje C/C++ que en el computador se va a ejecutar con el fin de visualizar si la lógica utilizada es correcta y se resuelve el problema de forma exitosa.

Para una correcta utilización de la programación por bloques el usuario deberá tener como conocimiento previo la construcción de un diagrama de flujo el cual describe gráficamente un proceso, en este caso un algoritmo informático. Un diagrama puede ser usado para documentar, estudiar, planificar, mejorar y comunicar un algoritmo, la

representación gráfica de estos procesos en los diagramas de flujo utiliza un conjunto específico de figuras geométricas que representan cada fase específica del proceso que se evalúa. Estas formas predefinidas están unidas entre sí por flechas y líneas que marcan la dirección del flujo y, como en un mapa, definen el curso del proceso. De esta manera se comprenderá de mejor manera la lógica con la que funciona un programa.

El funcionamiento del programa es simple, en los módulos de Aprendizaje y Manual de Usuario se tiene la información detallada de cómo realizar diagramas en el prototipo, cuáles son los campos permitidos, que hace cada uno de los bloques y como escribir en ellos, por ejemplo: Cuando sea necesario utilizar un bucle, ya sea "*FOR*" o "*WHILE*" es necesario conocer la sintaxis para evitar errores en la programación tradicional.

En el Módulo de Programación se tiene a disposición los bloques de programación, el área de trabajo (donde se desarrollan los diagramas de flujo) y al costado derecho los botones de "Generar Código" (botón encargado de extraer el código del diagrama creado); Además, el botón "Guardar Código" (encargado de guardar de forma local el código obtenido) y finalmente el botón "Ejecutar" (ejecuta el código obtenido y si es necesario aparece un recuadro donde se llenan los datos ingresados por teclado), también cuenta con un recuadro donde se puede visualizar el código extraído y el resultado de la ejecución.

A. Fase Teórica

Se estudiará el Modelo Vista Controlador (MVC) [5] [6] [7] con el cual se va a desarrollar la aplicación utilizando las 2 capas de este modelo (Controlador – Vista). Para el desarrollo de la programación de la aplicación web se estudia diferentes lenguajes de marcado y de programación como*: HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JavaScript, C/C++.* Se va a buscar información acerca del *Framework* que va a servir de apoyo en la mejora del proceso de desarrollo, como lo es la aplicación *React Js*, con su librería *React Flow*. Por último, se dará a breve rasgos una definición del Tablero Kanban [8] [9] y su metodología, así como si aplicación en al realizar este proyecto.

B. Fase de Diseño

La primera tarea consiste en actualizar el tablero Kanban, en el cual se detalla el estado de las actividades que se realizaron en la fase anterior y las tareas que se han añadido en la presente fase. Para seguir la metodología Kanban, se definirán las actividades que el sistema va a ejecutar, lo que permitirá establecer el tablero Kanban con cada una de las tareas a realizar.

Se analizará los requerimientos Funcionales y No Funcionales que necesita cumplir el sistema. Posteriormente se desarrollarán los diagramas *UML* (*Unified Modeling Processing*): Diagrama de clases y Diagrama de actividades.

Teniendo en cuenta los requerimientos Funcionales y No Funcionales, se elaboran los bosquejos primarios de cómo se verá la aplicación y se analiza cómo se irán desarrollando cada una de las funcionalidades del sistema, así como que tipo de programas se va a utilizar para lograr desarrollar e implementar el prototipo.

C. Fase de Implementación

Se procederá a la instalación de los diferentes programas que van a ser usados en el desarrollo de este sistema, para lo cual se iniciará con la actualización del tablero Kanban, dando por finalizadas algunas de las tareas que se han planteado en la Fase de Diseño como en la Fase Teórica. Para empezar, se realiza la instalación de *Visual Studio Code* y *React Js* y las librerías *Reactflow*, *Bootstrap* y *Child Process*. Una vez terminado las instalaciones, se procederá a la codificación en base al diseño definido. Teniendo el Diseño ya establecido y los módulos necesarios, se codifica cada uno de estos, siendo el principal y más complejo el Módulo de Programación, pues es donde podemos encontrar la interfaz para programar en base a desarrollo de bloques.

1.4. MARCO TEÓRICO

En el **Apartado 1.4.1**, se analizará el Estado del Arte en Sistemas Software de enseñanza de la materia de programación. En el **Apartado 1.4.2** se estudiará la Metodología Kanban para la gestión y organización del proyecto. Siguiente, en el **Apartado 1.4.3** se detallarán Herramientas y Tecnologías a utilizar dentro del Prototipo que utiliza bloques de código para escribir programas. Estos bloques de código se organizan en estructuras lógicas y sintácticas específicas, las cuales se utilizan para controlar el flujo de un programa y realizar tareas específicas, dentro del diseño elaborado.

1.4.1. ESTADO DEL ARTE EN SISTEMAS DE SOFTWARE DE ENSEÑANZA DE PROGRAMACIÓN.

Si bien es cierto existen algunos proyectos que tratan de enseñar a programar de una manera no convencional, con el fin de facilitar el aprendizaje de programación y salir de los paradigmas para llegar de mejor manera al estudiante, y así aprovechar las

destrezas de cada una de las personas. Aprender, tanto como enseñar a programar, es una tarea compleja, es por esta razón que existen distintas formas de enseñanza, que de forma no convencional (no únicamente utilizando una computadora y el compilador del lenguaje empleado) muestran varias metodologías que permitan de mejor manera aprender a programar, como se enumera a continuación en la Tabla 1.1:

Tabla 1.1 Programas y proyectos relacionados a la enseñanza de programación.

Proyecto	**Definición**
CODING DOJO [10]	Es una metodología de enseñanza de la programación que se centra en la práctica y la repetición para ayudar a los estudiantes a aprender y mejorar sus habilidades de programación. La metodología se basa en la idea de que la mejor manera de aprender a programar es a través de la práctica continua y la repetición de los conceptos fundamentales. Ofrece cursos de programación interactivos, utilizando gráficos para explicar el funcionamiento de cada uno de los comandos, sin embargo, no tiene un sistema de programación grafica como tal, sino es más bien una escuela de programación, en la que se reúnen un gran número de personas para trabajar de forma colectiva tanto para enseñar como para resolver e implementar soluciones a lo que alguno de sus miembros necesite. La metodología se divide en varios pasos: 1. Un líder del dojo presenta un desafío o problema a los estudiantes. 2. Los estudiantes trabajan en el desafío en equipos de 2 a 4 personas. 3. Cada equipo presenta su solución al desafío al final de un periodo de tiempo específico. 4. El líder del dojo revisa las soluciones y brinda retroalimentación. 5. Los estudiantes practican y refactorizan sus soluciones en base a la retroalimentación recibida. 6. El proceso se repite varias veces, con desafíos cada vez más difíciles. Algunas de las ventajas de *Coding Dojo* es que es una forma de enseñanza colaborativa, se fomenta la comunicación y el trabajo en equipo, se enseña a los estudiantes a pensar de manera

	lógica y a resolver problemas de manera efectiva, también al aprender a través de la repetición de los conceptos básicos, se establece una base sólida para el aprendizaje continuo.
PSEINT [11] [12] [13]	Es un programa de computadora que permite a los usuarios escribir, probar y depurar programas de computadora utilizando una interfaz visual y un lenguaje de programación sencillo y fácil de entender. Se basa en el lenguaje Pascal, con una sintaxis y estructuras simples. Su diseño está pensado para que los principiantes puedan aprender a programar de manera sencilla y amigable. Es una herramienta que apoya a un estudiante en sus primeros pasos en la programación. El uso de un pseudolenguaje[2] en español, que incluye un editor de diagramas de flujo sencillo e intuitivo, permite concentrarse en los conceptos fundamentales de los algoritmos computacionales. Al reducir las barreras del idioma, se proporciona un entorno de trabajo con recursos de aprendizaje y asistencia que facilitan el proceso de aprendizaje. El pseudocódigo[3] se suele emplear como una forma inicial de familiarizarse con los conceptos fundamentales de programación, como el manejo de estructuras de control, expresiones, variables, entre otros, sin necesidad de preocuparse por las reglas sintácticas específicas de un lenguaje de programación específico. Este software tiene como objetivo facilitar la escritura de algoritmos en este pseudolenguaje para los principiantes al presentar una serie de consejos y trucos y proporcionar algunas herramientas adicionales para ayudarlo a encontrar errores y comprender la lógica de los algoritmos.

[2] Pseudolenguaje es un lenguaje de programación simplificado que se utiliza para facilitar el aprendizaje de la programación. Está diseñado para ser fácil de entender y utilizar por las personas que no tienen experiencia en programación.

[3] Pseudocódigo es una representación textual de un algoritmo o programa, que utiliza un lenguaje estructurado de alto nivel para describir las acciones que se deben realizar en un programa

TINKERCAD CODEBLOCKS - ORIENTADO A SISTEMAS EMBEBIDOS [14] [15]	Es una plataforma de programación visual basada en bloques, diseñada para ayudar a los principiantes a aprender a programar de manera intuitiva y sencilla. Se utiliza principalmente para programar dispositivos con Arduino y otros microcontroladores. La programación en *Tinkercad Codeblocks* se basa en el uso de bloques de construcción, que representan diferentes funciones y estructuras de programación. Los bloques se pueden conectar entre sí para crear programas complejos. La plataforma ofrece una variedad de bloques de construcción que se pueden utilizar para controlar entradas y salidas, como sensores, actuadores y pantallas. También proporciona bloques para controlar la ejecución de un programa, como bucles y condicionales. Es un gran sistema para emular la programación de sistemas embebidos, principalmente el uso de Arduino, con un apartado denominado *CodeBlocks* en el cual se utilizan elementos geométricos como si fueran bloques de construcción que apilándolos se va a tener de forma “compacta” una sección estructurada de bloque, en la cual se van a rellenar ciertos campos, dependiendo del bloque que se seleccione.
FLEXBOX FROGGY [16]	A pesar de que sea una metodología orientada a la programación en *HTML*, más específicamente a los estilos de una página web. *Flexbox Froggy* es un juego de aprendizaje de programación basado en el uso de *Flexbox*, un sistema de diseño de cajas en CSS que se utiliza para organizar y distribuir elementos en una página web. El juego se basa en el uso de comandos CSS para resolver desafíos y organizar los elementos de una página web en una forma específica. El jugador debe utilizar los comandos *CSS* adecuados para colocar los elementos en su posición correcta en la página web. Los desafíos van desde colocar un solo elemento hasta crear diseños complejos utilizando varios elementos y diferentes valores de propiedades de *Flexbox.* El juego es una buena manera de aprender los conceptos básicos de *Flexbox* de manera interactiva y divertida, ya que a medida que el jugador completa desafíos, va aprendiendo como

	utilizar las diferentes propiedades de *Flexbox*. Además, ayuda a entender como los elementos se relacionan entre sí en un diseño y cómo se pueden ajustar para lograr el diseño deseado.
MINECRAFT FOR EDUCATION - CRACK THE CODE [17]	Un ejemplo destacado del uso de un sistema de programación gráfico es el juego muy conocido llamado *Minecraft*, en el que se utilizan bloques para llevar a cabo diversas acciones, lo que ha sido muy valorado por sus usuarios. La empresa *Crack the Code* ha desarrollado una forma de enseñar programación a niños mientras juegan este popular videojuego. *Minecraft for Education* es una versión educativa del popular juego *Minecraft*, que se utiliza para enseñar programación y habilidades de pensamiento computacional de manera lúdica y divertida. La versión educativa de *Minecraft* incluye la capacidad de crear programas y scripts utilizando bloques de construcción, conocido como *Crack the Code*. *Crack the Code* es un recurso educativo dentro de *Minecraft for Education* que permite a los estudiantes programar en un ambiente virtual utilizando bloques de construcción para controlar los aspectos del juego. Los estudiantes pueden utilizar estos bloques para crear programas para automatizar tareas, mover entidades y crear objetos en el juego.

Destacando cada uno de los ejemplos revisados, el uso de una forma no convencional de aprender favorece de gran manera para el aprendizaje de un lenguaje de programación, aunque este enfoque de enseñanza se lo ha dejado únicamente a niños, resulta muy útil para todas las personas que son nuevas al momento de aprender a programar, y que mejor si se encuentra una manera de que todo el sistema de programación sea de forma gráfica facilitando el aprendizaje.

A pesar de que existen varios proyectos con la intención de tener una manera más fácil e intuitiva de enseñar a programar, el hecho de ser un sistema web, podría permitir que se pueda ingresar desde cualquier parte para poder realizar el aprendizaje con este sistema. Y que está desarrollado en herramientas nuevas que le pueden dar una nueva visión al desarrollo de aplicaciones de este estilo.

1.4.2. METODOLOGÍA KANBAN

Kanban es una metodología que ayuda a los equipos de desarrollo a alcanzar un equilibrio entre las tareas pendientes y la capacidad de trabajo de cada miembro del equipo. La metodología Kanban se basa en una filosofía basada en la mejora continua, en la que se extraen actividades de una lista de actividades pendientes en un flujo de trabajo continuo.

Se ha implementado el tablero Kanban en el proceso de programación, ya que su uso resulta beneficioso para los desarrolladores, quienes consiguen obtener un trabajo productivo, rápido y eficiente, donde la organización sea la que destaque al momento de desarrollar cualquier tipo de proyecto. El uso de esta metodología ha traído muy buenos resultados destacando empresas muy grandes que lo han adoptado en sus procesos, así como Nike que desde su implementación mejoró resultando ser una de las marcas favoritas en el mundo en la actualidad. Kanban es parte de una metodología de manufactura la cual se basa en la técnica JIT (*Just in Time*) que significa hacer solo lo que es necesario.

El objetivo principal de la técnica es garantizar tasas de producción estables para evitar excedentes de productos terminados, y retrasos en la entrega; Por lo tanto, el trabajo debe organizarse en función de la capacidad de los centros y equipos de procesamiento. Los tableros Kanban son herramientas flexibles de gestión de proyectos diseñadas para visualizar el trabajo, eliminar retrasos y aumentar la eficiencia (o el flujo). Esto ayuda a los equipos a determinar el orden del trabajo diario. Los tableros Kanban usan pestañas, columnas y mejoras continuas para ayudar a los equipos técnicos y de servicio a realizar la cantidad correcta de trabajo y, por supuesto, hacerlo.

Las columnas comúnmente utilizadas en un tablero Kanban incluyen:

- "Por hacer": Las tareas que aún no han sido asignadas a un miembro del equipo.
- "En progreso": Las tareas que están siendo trabajadas en este momento.
- "Completadas": Tareas que han sido finalizadas y aprobadas.

Las tareas individuales se representan mediante tarjetas o "Post-it" que se mueven a través de las columnas del tablero a medida que se completan. Esto permite a los miembros del equipo ver fácilmente el estado de una tarea y quién la está trabajando. El tablero también puede incluir información adicional como el propietario de la tarea, fecha de vencimiento, y prioridad. Además, también se pueden utilizar indicadores

visuales como gráficos de burbujas, para ver cómo se está desempeñando el equipo en tiempo real.

1.4.3. HERRAMIENTAS, TECNOLOGÍAS Y CONCEPTOS PARA LA IMPLEMENTACION DEL PROTOTIPO DE SOFTWARE.

Para el desarrollo de esta aplicación web se han utilizado varios programas y librerías, los cuales han permitido que el despliegue de este prototipo sea ágil y efectivo, logrando así cumplir con los requerimientos de este proyecto.

Las distintas herramientas, librerías y conceptos utilizados para la codificación de esta aplicación, han sido seleccionadas con base en cuál es la mejor opción para lograr un resultado óptimo y cumplir con todos los objetivos trazados. Por tanto, en el desarrollo de esta aplicación se han usado las siguientes herramientas, librerías y conceptos mostrados en la Tabla 1.2:

Tabla 1.2 Herramientas y Tecnologías a utilizar.

Elementos utilizados	
Modelo Vista Controlador (MVC) [5] [6] [7]	*MVC* dicta un patrón a seguir para que una aplicación sea desarrollada de manera organizada y estructurada. La base para esto es dividir el código en tres niveles distintos, delimitados por sus responsabilidades, llamados modelos, vistas y controladores. *MVC* se usa principalmente en sistemas que requieren una interfaz de usuario, aunque en la práctica se puede usar el mismo modelo arquitectónico para muchos tipos diferentes de aplicaciones. Esto surge de la necesidad de crear un software más confiable y con un ciclo de vida más adecuado, en el que se mejore la mantenibilidad, la reutilización del código y la separación de conceptos. En la capa Vista, donde se encuentran cada una de las interfaces que el estudiante va a visualizar y con las que va a interactuar cuando utilice el programa, su objetivo es claro pues busca facilitar el uso de los mecanismos usados para que se pueda dar la programación. En la capa Controlador se tienen todos los procesos que realiza la página web, cómo lo son la construcción de los bloques, así como para obtener tanto el código como la llamada a la ejecución del programa realizado por el usuario.

	En la capa Modelo es donde se encuentran las entidades que van a servir para almacenar datos, tanto los que van por defecto como los datos generados por la aplicación. Cabe recalcar que, al no necesitar almacenar datos, este proyecto no cuenta con la Capa de Modelo, todas las distintas funcionalidades se realizan a nivel de la Capa Controlador, tanto como en la Capa Vista.
React JS	*Framework React Js*, *React* es una librería de *JavaScript* centrado en el desarrollo de la interfaz de usuario. Así se define la propia biblioteca, y por supuesto esta es su principal actividad. Pero el caso es que en *React* se encontró un gran aliado para crear cualquier tipo de aplicación web, *SPA* (S*ingle Page Application*) o aplicación móvil. Para lograr esto, existe un ecosistema completo de módulos, herramientas y componentes que rodean a *React*, lo que permite a los desarrolladores lograr objetivos avanzados con un mínimo esfuerzo. Entonces, *React* es una base sólida para construir sistemas web en conjunto con *JavaScript*. También es fácil de desarrollar ya que proporciona contenido prefabricado y lo único necesario es instalar las librerías y su posterior llamado, siendo las librerías utilizadas en el desarrollo de este prototipo un ejemplo de ello.
React Flow	Es una biblioteca para crear aplicaciones basadas en nodos. Estos pueden ser diagramas estáticos simples o editores complejos, en los cuales se pueden ir creando varios nodos, preestablecidos y agregarlos a la pantalla de programación. Puede implementar tipos de nodos y tipos de bordes personalizados y viene con componentes como un mini mapa y controles de gráfico. Tal como *Panning and zooming*[4], que es el modelo que permite realizar el desplazamiento en el “mapa” de la aplicación, así como realizar acercamientos y alejarse si es necesario para poder tener una vista mejor del módulo de programación, además de una de las características poderosa de *React Flow* es la capacidad de agregar

[4] Panning and zooming es una técnica utilizada en interfaces gráficas de usuario para navegar por contenidos visuales, permitiendo al usuario desplazarse a través de un espacio gráfico más grande o reducir o ampliar la vista de un objeto gráfico.

	nodos personalizados. Puede mostrar lo que quiera en nodos personalizados. Se pueden definir múltiples *ID* de origen y destino, por ejemplo, para mostrar elementos de formulario o gráficos. En esta sección se implementará un nodo con un campo de entrada que actualizará el texto en otras partes de la aplicación. En la librería *ReactFlow*, los nodos se utilizan para crear diagramas de flujo visuales y para representar procesos y operaciones en una aplicación. Algunos ejemplos de nodos en *ReactFlow* incluyen: • Entrada y salida de datos: nodos que se utilizan para recibir y enviar información a otros nodos. • Procesamiento de datos: nodos que realizan operaciones en los datos de entrada y producen resultados. • Decisión: nodos que toman decisiones basadas en los datos de entrada y dirigen el flujo de trabajo a diferentes rutas. Los nodos en *ReactFlow* son altamente personalizables y se pueden configurar para adaptarse a las necesidades específicas de una aplicación. Los desarrolladores pueden crear nodos personalizados o utilizar nodos preconstruidos en la librería.
Bootstrap [18] [19] [20]	*React*-Bootstrap es un *Framework* dedicado al front-end, es decir se dedica específicamente a la parte visual de una aplicación, es un conjunto de archivos *Cascading Style Sheets* (CSS) y *JavaScript*, como una de las principales funciones es que la página web sea *responsive*[5] y que se adapte a cambios de hardware. Se utiliza para crear interfaces de usuario atractivas y responsivas que se ajustan a diferentes tamaños de pantalla. *Bootstrap* ofrece una amplia variedad de componentes y herramientas de diseño, como formularios, menús de navegación, tablas, botones, modales y mucho más. *Bootstrap* y *React* pueden utilizarse juntos para crear aplicaciones web y móviles de alta calidad. Para hacer esto, los desarrolladores pueden utilizar componentes de *Bootstrap* en *React* y crear componentes personalizados que se ajusten a las necesidades de su aplicación.

[5] Una página web responsive es una página que se adapta automáticamente al tamaño y resolución de la pantalla del dispositivo utilizado para visualizarla.

Child Proccess [21]	Este módulo permite manejar, crear, manipular, compilar algunos procesos del procesador. Se utiliza *Child Process*, el cual permite crear subprocesos, es decir que va a realizar consultas al computador, las cuales se ejecutan como multitareas en al mismo tiempo de la ejecución del programa, para así obtener la respuesta a la compilación realizada. Es decir, permite a los desarrolladores lanzar aplicaciones y comandos externos desde una aplicación *Node.js.* Los desarrolladores pueden utilizar *Child Process* para realizar tareas pesadas en segundo plano, como la descarga de archivos, el procesamiento de datos y la ejecución de comandos externos. Esto ayuda a mejorar el rendimiento y la eficiencia de la aplicación. *Child Process* ofrece diversas opciones para controlar y configurar los procesos secundarios, como la asignación de prioridades, la gestión de errores y la terminación de procesos.
G++ [22]	Es un compilador de Línea de Ordenes para programas de C/C/C++, utilizando la consola de la PC. El uso de G++ permite al desarrollador escribir programas en el lenguaje C/C++ y luego traducirlos a código máquina para que puedan ser ejecutados en una computadora. El compilador toma el código fuente escrito en C/C++ y genera un archivo ejecutable que contiene el código objeto y el código de enlace necesario para crear el programa final. Es necesaria su instalación para mediante una *API* conectar a la aplicación desarrollada y que esta pueda resolver los ejercicios que se plantean y envíe el resultado a la interfaz de la aplicación.
Visual Studio Code [23] [24]	*Visual Studio Code* (*VS Code*) es el editor de código de *Microsoft*. Es un software gratuito y multiplataforma disponible para *Windows*, *GNU/Linux y macOS. VS Code* tiene una buena integración con *Git*, compatibilidad con la depuración de código y varias extensiones que básicamente le permiten escribir y ejecutar código en cualquier lenguaje de programación, gracias a que se pueden instalar plugin, que permiten desarrollar un prototipo de manera correcta al poder observar si es que existe un error en la codificación.

API [25]	*API* (*Application Programming Interfaces*), son protocolos y conceptos que permiten desarrollar e integrar software de varias aplicaciones para que puedan interactuar entre ellas siguiendo reglas, las que son establecidas en el mismo. El uso de una *API* permite a los desarrolladores crear aplicaciones más complejas y robustas, aprovechando las funcionalidades de otros sistemas y servicios, y ofrece mayor flexibilidad para construir soluciones personalizadas y escalables. Dado a que la aplicación, necesita resolver programación, es decir, enviar consultas al procesador del computador, se necesita de la implementación de una *API* la cual permite la interacción entre aplicaciones, en este caso se van a conecta *G++* y la aplicación desarrollada.
Diagramas de flujo [26] [27]	Los diagramas de flujo son una herramienta visual que representa gráficamente el flujo de un programa. Son útiles para representar de manera clara y concisa el funcionamiento de un programa y para ayudar a los programadores a entender el algoritmo subyacente. En el desarrollo por bloques los diagramas de flujo se unan como una forma de hacer la programación más accesible y fácil de entender para los principiantes. En lugar de escribir código directamente, los usuarios pueden crear diagramas de flujo que representen el flujo de un programa. Luego, el software traduce automáticamente el diagrama de flujo en código ejecutable. Algunas de las características de los diagramas de flujo en la programación por bloques incluyen: • Representación gráfica clara y fácil de entender del flujo de un programa • Herramientas para crear y modificar diagramas de flujo fácilmente • Integración con el código ejecutable, permitiendo a los usuarios ver y editar el código directamente a partir del diagrama de flujo.

2. METODOLOGÍA

En este capítulo se mostrará la construcción de la aplicación web paso a paso, empezando con la planificación de las actividades realizadas, los bosquejos iniciales y los detalles con los cuales se llegó a la aplicación final.

La metodología utilizada en la ejecución del proyecto es de tipo aplicada[6] resolviendo el problema planteado con la propuesta del aplicativo *Flowchart*.

A lo largo del **Apartado 2.1**, correspondiente al Diseño, se definirán las principales pautas para el desarrollo de la aplicación de escritorio; por otra parte, en el **Apartado 2.2**, Implementación, se explorarán las decisiones de desarrollo tomadas.

2.1. DISEÑO

Esta etapa se enfoca en establecer cómo se organizarán los componentes de la aplicación y cómo interactuarán entre sí.

Se observa todos los detalles en cuenta para desarrollar el prototipo denominado *Flowchart*, teniendo en cuenta los requerimientos Funcionales y No Funcionales, las capas de la Arquitectura MVC y el diseño final de la aplicación web.

2.1.1. PLANTEAMIENTO DEL TABLERO KANBAN

Para la realización del Tablero Kanban se ha utilizado *Lucid Chart*, donde se registran cada una de las tareas y se ubican según corresponda en las distintas tablas del tablero mencionadas en el **Apartado 1.4.2.** y se identifica a que fase pertenecen las tareas según el color establecido para cada fase como se muestra en la **Figura 2.1.**
Para la fase de Diseño se encontraron cinco tareas, las cuales son:

- Recopilar Requerimientos Funcionales y No Funcionales
- Diseñar Diagramas de Casos de Uso y de Actividades
- Diseñar bosquejos de la aplicación.

[6] Metodología de tipo aplicada: Es un conjunto de pasos y procedimientos que se siguen para alcanzar los objetivos establecidos.

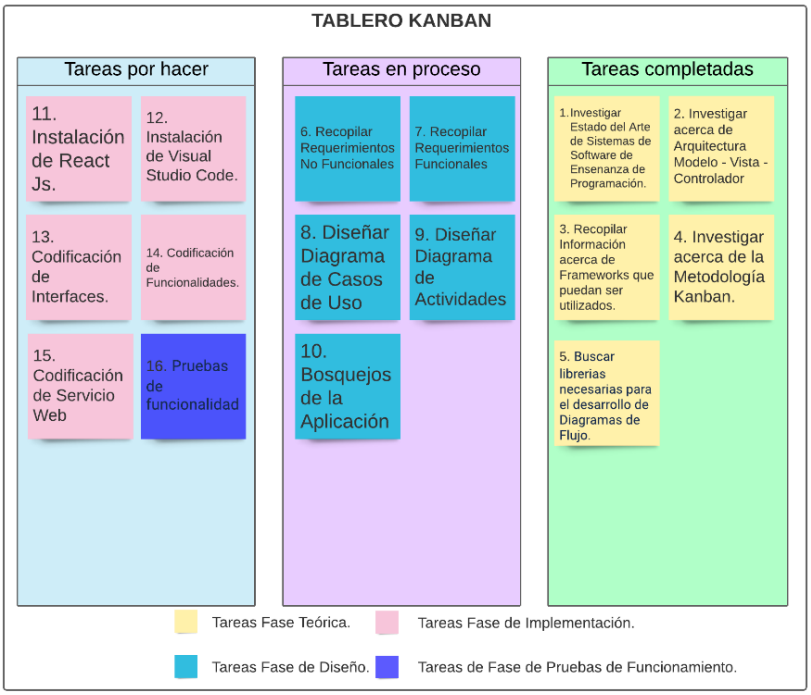


Figura 2.1 Tablero Kanban en Etapa de Diseño.

2.1.2. REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales son una lista de características y funcionalidades específicas que un sistema, producto o servicio debe tener para cumplir con las necesidades de los usuarios. Estos requisitos son esenciales para el funcionamiento correcto y la utilidad del sistema y describen las funciones que el sistema debe cumplir.

Con los requerimientos funcionales se describe qué actividad necesita que realice la aplicación, también se describen como lo va a hacer. Para empezar, se detallan las Historias de Usuario que son referidas a cada código según corresponda a su Requisito Funcional.

Tabla 2.1 RF 01: Manual de Usuario.

RF 01	**Desarrollar el Manual de Usuario**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Se requiere de la construcción de un Manual de Usuario	
Criterios de Aceptación:	
• Se encuentran las formas de como unir las líneas, como rellenar los campos de los bloques, donde se encuentran los botones necesarios para obtener el código.	

Tabla 2.2 RF 02: Módulo de Aprendizaje.

RF 02	**Implementar Módulo de Aprendizaje**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Módulo de aprendizaje para la autoeducación del estudiante.	
Criterios de Aceptación:	
• Para garantizar que el estudiante pueda aprender solo, se necesita de guías en las cuales él pueda revisar cómo funciona el programa. • En el módulo de Aprendizaje, además, se encuentran videos demostrativos de cómo funcionan los bloques y como se rellena cada uno de ellos. Ejemplos de programas realizados.	

Tabla 2.3 RF 03: Agregar Bloques de Programación.

RF 03	**Agregar bloques en el Área de Programación**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Se requiere agregar bloques al área de programación.	
Criterios de Aceptación:	
• En el Módulo de Programación para comenzar a modelar un Diagrama se deben agregar los distintos bloques al área de programación, para ser unidos entre sí y crear un programa.	

Tabla 2.4 RF 04: Creación de Diagrama de Flujo.

RF 04	**Modelar diagrama de flujo en el área de programación**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Crear diagramas en el en el Módulo de Programación.	
Criterios de Aceptación:	
• Cada bloque de programación, por cualquier motivo va a tener una ayuda para que el estudiante pueda saber con qué tipo de dato puede rellenar cada campo. • Tendrá una señal de alerta cuando se esté ingresando de forma incorrecta un dato.	

- Cada uno de los bloques de programación tendrá de forma clara en que puntos va a unir con líneas los bloques.
- Dentro del módulo de programación se encuentra el boto de Obtener código, el cual al presionarlo se obtiene el código en Lenguaje C/C++.

Tabla 2.5 RF 05: Diseñar Bloques de Programación.

RF 05	**Diseñar bloques de programación**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Diseñar cada uno de los bloques y explicar cómo funciona cada uno de los bloques de programación	
Criterios de Aceptación:	

- Cada uno de los bloques de programación se encuentran a disposición del estudiante en un recuadro del módulo de programación.
- **Bloque IF:** para el bloque *if* se tiene su bloque complemento *"end_if*" es el que permite el cierre del condicional. en el bloque se pueden utilizar sentencias como símbolos matemáticos (<,>, ==, entre otros) y funciones del lenguaje de programación (*fmod, powf*, entre otros), así como utilizar operadores lógicos (&&, ||) que permiten realizar una condición más completa.
- **Bucles:** Se tienen dos funciones bucles: *For y while*, cada uno de estos bucles tiene su respectivo "*end*" el cual debe ser agregado para cerrar el bucle. cada uno de los bucles va a necesitar una variable con la cual va a crecer o decrecer, un punto final o donde pueda salir del bucle o hasta que se cumple cierta condición, además en el bucle *for* se ingresa el salto de la variable que depende como parámetro de inicio.
- **Proceso:** Bloque donde se asigna valores, se realizan las ecuaciones y se utilizan fórmulas matemáticas.
- **Resolución de casos:** *Switch* (bloques: *switch, end* - case (bloques: *case - end*), comúnmente utilizado para la creación de menús, donde en cada opción o *case* resuelve un tipo de proceso.
- **Ingreso de datos:** Bloque *in* permite el ingreso de los datos por teclado que se han asignado a una variable, para su correcta ejecución.
- **Bloque de Impresión:** Bloque *out* se utiliza para imprimir el valor de una variable o un escrito dentro del programa.

• **Bloque de asignación de variable:** Con este bloque a cada variable creada se le asigna una etiqueta (*int, char, string, boolean o double*) para su utilización dentro del sistema.

Tabla 2.6 RF 06: Obtener de forma ordenada el código.

RF 06	**Obtener de forma ordenada el código**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Obtención de código a partir del Diagrama de Flujo	
Criterios de Aceptación:	
• Después de crear un Diagrama de Flujo en la aplicación, para poder obtener el código, el estudiante debe dar *click* en “Generar código”, código aparece en la parte derecha de la pantalla.	

Tabla 2.7 RF 07: Compilar código.

RF 07	**Compilar código obtenido.**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Compilar el diseño construido	
Criterios de Aceptación:	
• Una vez creado el diagrama de flujo y que se haya obtenido el código, la aplicación será capaz de compilar el programa y ejecutar. • Obtener resultado según el diagrama de flujo diseñado y la lógica utilizada para la resolución de problema planteado.	

Tabla 2.8 RF 08: Descarga de código generado.

RF 08	**Descargar código generado**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Descargar el código que se obtiene a partir del Diagrama de Flujo	
Criterios de Aceptación:	

• Una vez generado el Diagrama al presionar el boto de generar código se observa el código en pantalla, el cual se lo puede descargar con el objetivo de ejecutarlo cuando se quiera y de “guardar” lo realizado, el código que se descarga es en lenguaje C/C++.

2.1.3. REQUERIMIENTOS NO FUNCIONALES

Los requisitos no funcionales de una aplicación de programación se refieren a las características o restricciones del sistema que no están relacionadas directamente con las funciones específicas que el sistema debe cumplir, sino más bien con cómo el sistema debe funcionar. Estos requisitos incluyen, entre otros, aspectos como rendimiento, seguridad, escalabilidad, disponibilidad, compatibilidad y accesibilidad.

Tabla 2.9 RNF 01: Interfaz intuitiva.

RNF 01	**Interfaz intuitiva**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Interfaz sencilla de entender y acceder fácilmente a los módulos	
Criterios de Aceptación:	
• Desarrollar la interfaz de la forma más sencilla y que únicamente estén los elementos necesarios para evitar confusiones y sea sencillo navegar entre los módulos.	

Tabla 2.10 RNF 02: Usabilidad del prototipo.

RNF 02	**Usabilidad del prototipo**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Interfaz intuitiva para facilitar el uso del aplicativo	
Criterios de Aceptación:	
• El prototipo contará con una página índice donde se podrá seleccionar el módulo de programación en el cual se van a tener los diferentes bloques de programación. • El sistema debe ser intuitivo, con el fin de que el estudiante sepa cómo llegar fácilmente al módulo de programación.	

• Teniendo en cuenta el criterio de desarrollo de Diagramas de Flujo, utilizar Figuras conocidas al momento del diseño de los bloques.

Tabla 2.11 RNF03: Disponibilidad.

RNF 03	**Asegurar Disponibilidad.**
Usuario:	Estudiante
Prioridad:	Alta
Descripción:	
Disponibilidad del prototipo desarrollado.	
Criterios de Aceptación:	
• Al ser un prototipo web, tiene un alto grado de disponibilidad y rapidez.	

2.1.4. MÓDULOS DEL PROTOTIPO

Los módulos creados para este aplicativo son: Módulo de Programación, Módulo de Aprendizaje y Manual de Usuario. El desarrollo de estos tres módulos permite que el sistema sea suficiente para que un estudiante pueda ingresar al mismo y comenzar a aprender por sí solo.

- **Módulo de aprendizaje**

Módulo que proporciona información detallada sobre cómo utilizar el prototipo de programación por bloque. El manual incluye descripciones detalladas de la interfaz de usuario, los distintos de bloques de programación, la ejecución de programas y la depuración, así como instrucciones para guardar y exportar código.

Además, incluye ejemplos y tutoriales paso a paso para ayudar a los usuarios a comprender cómo crear y editar programas de manera efectiva. También se incluyen explicaciones detalladas sobre cómo solucionar problemas comunes y cómo utilizar la herramienta de manera eficiente. Recurso valioso por proporcionar información clara y concisa, el manual ayuda a los usuarios a aprovechar al máximo la funcionalidad de la herramienta y a crear programas de manera efectiva.

- **Módulo de Programación**

En el Módulo de Programación se encuentra la parte más importante del aplicativo, donde se permite al usuario crear y editar programas de manera visual e intuitiva, utilizando bloques de programación en lugar de escribir código de texto. Este módulo

proporciona una interfaz gráfica fácil de usar y bloques de programación con sintaxis claras y fáciles de entender. Además, permite obtener y compilar el código y exportar el código. Esto hace que la programación sea más accesible y amigable para los usuarios, especialmente aquellos que son nuevos en la programación.

- **Módulo de Manual de Usuario**

El objetivo del manual es proporcionar instrucciones claras y detalladas para ayudar a los usuarios a comprender y utilizar todas las funcionalidades de la aplicación. Este manual es esencial para asegurar que los usuarios comprendan y aprovechen al máximo la funcionalidad de la herramienta.

Se van a identificar cada uno de los bloques, se muestra como unir bloques, que código preestablecido contiene cada una de las figuras y como utilizar los diferentes botones de la interfaz del módulo de programación. Además, como es la navegación dentro de la interfaz de la aplicación.

2.1.5. DISEÑO DE LA CAPA CONTROLADOR

La Capa de Controlador es la responsable de recibir las peticiones del usuario, procesarlas y determinar cómo deben ser manejadas. Su función principal es gestionar la ejecución de los programas creados por los usuarios y garantizar que los programas se ejecuten de manera eficiente y sin errores.

La Capa Controlador trabaja en conjunto con la interfaz de usuario y la biblioteca de bloques de programación para proporcionar una experiencia de programación visual e intuitiva para los usuarios. Por ejemplo, cuando un usuario agrega un bloque de programación a su proyecto, la capa controladora se encarga de verificar que el bloque se integre correctamente con el resto del programa.

Además, la Capa Controlador proporciona características avanzadas como la depuración de programas y la capacidad de identificar en base a las líneas de código generadas donde se encuentra un error en la programación, para que sea más sencillo encontrar el problema en el diagrama de flujo y corregirlo rápidamente, de esta manera el programa será compilado de forma eficiente.

2.1.5.1. **Diagramas de Casos de Uso**

El término “Caso de Uso” hace referencia a una acción de un proceso, que en conjunto forman un Diagrama de Casos de Uso, el cual es una secuencia ordenada de acciones para cumplir un proceso, realizada por uno o más actores en un sistema, en este caso únicamente por el estudiante.

Como se observa en la **Figura 2.2** se tiene como único Actor al Estudiante, el cual tendrá a disposición los distintos módulos de la aplicación y toda la información necesaria para aprender a utilizar los bloques de programación, ver tutoriales de ejemplos realizados, la teoría necesaria detrás de cada una de las funciones comprimidas dentro de los bloques.

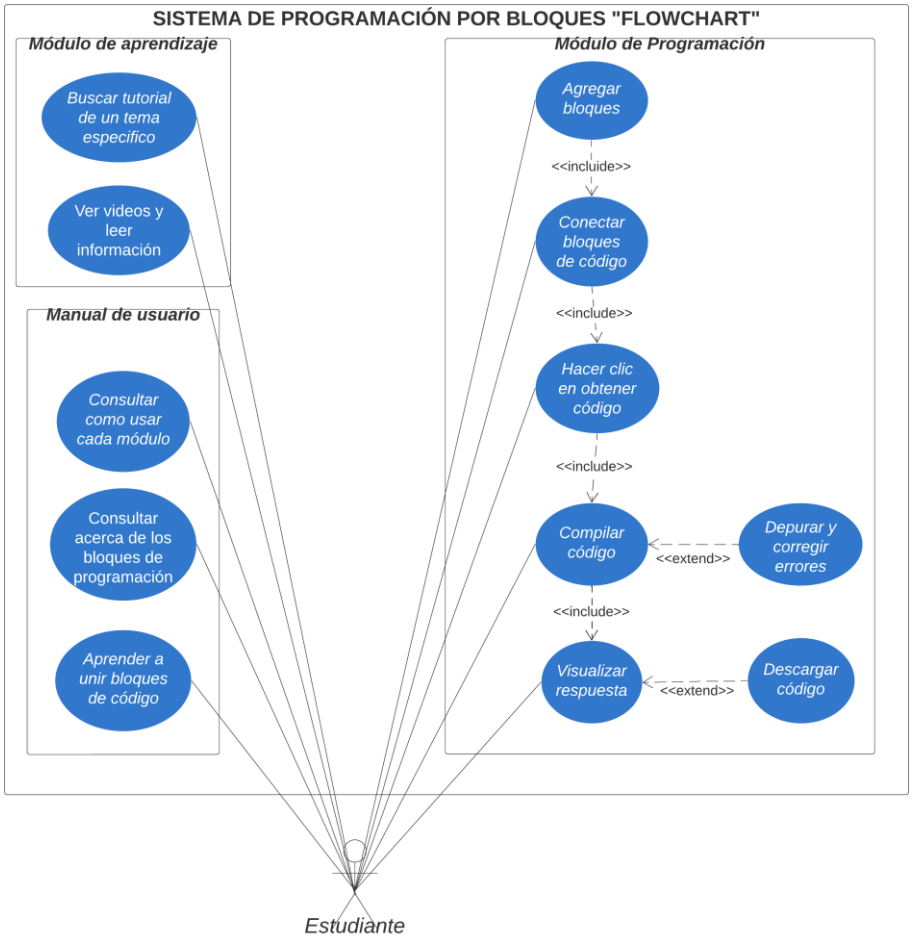


Figura 2.2 Casos de Uso del aplicativo "Flowchart".

2.1.5.2. DIAGRAMA DE ACTIVIDADES

Un Diagrama de Actividad, también conocido como diagrama de flujo de actividad, es una herramienta gráfica utilizada para representar el flujo de actividades que utiliza

símbolos para representar acciones, eventos, decisiones y flujos de control, en un sistema o proceso. Es una forma visual de representar el proceso desde el inicio hasta el final, mostrando los pasos que se siguen, las decisiones que se toman y los puntos de control o las excepciones [28] [29].

A continuación, en la **Figura 2.3**, se detallan los pasos a seguir para desarrollar un diagrama de Actividades, que empieza con la recepción de indicaciones para un determinado programa. De esa manera se ingresa al sistema y se empieza a desarrollar el ejercicio seleccionando los bloques que se van a utilizar, se asignan las variables y características dentro de cada bloque, según sea necesario, se unen los bloques dando forma así al diagrama. De esta manera se procede a generar el código y se ejecuta, si no existe error en el diagrama o las variables el código se compila de manera exitosa y si existe un error se debe revisar el diagrama y corregir los errores, una vez hayan sido corregidos se obtiene el código nuevamente y se ejecuta.

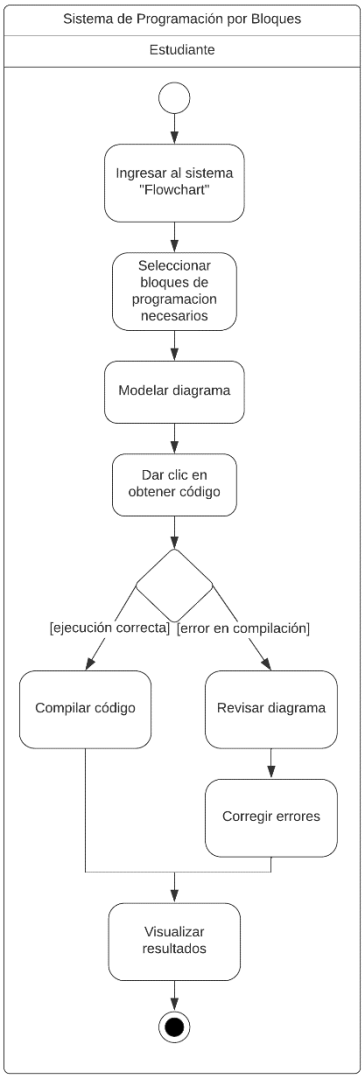


Figura 2.3 Diagrama de Actividades en el Módulo de Programación.

2.1.6. DISEÑO DE LA CAPA VISTA

La Capa Vista es responsable de presentar la información al usuario y recibir la entrada del usuario. En detalle, la Capa de Vista tiene las siguientes funciones en una aplicación web:

- **Generar interfaz de usuario**: La capa de vista es responsable de generar la interfaz de usuario, es decir, las páginas web que el usuario ve e interactúa. Esto puede incluir formularios, tablas, enlaces, etc. La vista puede utilizar lenguajes de marcas, como *HTML, CSS y JavaScript*, para generar la interfaz de usuario.
- **Recibir entrada del usuario**: La vista también es responsable de recibir la entrada del usuario, a través de formularios o cualquier otro mecanismo en la interfaz de usuario.
- **Procesar datos**: La vista también puede procesar los datos antes de presentarlos al usuario, por ejemplo, formateando una fecha, un número, etc.
- **Mostrar datos**: La capa de vista es responsable de mostrar los datos al usuario, sea en forma de texto, tablas, gráficos, etc.

Para el diseño de la Capa Vista, siguiendo con los Requerimientos No Funcionales, se plantean bosquejos iniciales del aplicativo como se observa en la **Figura 2.4**, siendo esta la página índice la primera interacción que tiene el Estudiante con el sistema y sirviendo para la navegación dentro de la aplicación.

2.1.6.1. Diseño de la Interfaz Gráfica

La interfaz gráfica de una aplicación tiene como principal objetivo facilitar la interacción de los usuarios con la misma, mejorando su eficiencia y productividad. Para ello, es importante contar con una interfaz intuitiva y fácil de usar, por ello se ha optado por una interfaz sencilla para la página índice, donde se muestran los distintos módulos de la aplicación de manera clara y concisa como se muestra en la **Figura 2.4**. Además, se ha trabajado en una interfaz gráfica amigable y eficiente en el área de programación, como se puede apreciar en la **Figura 2.5.** De esta manera, se busca mejorar la experiencia de los usuarios al utilizar la aplicación y aumentar la satisfacción con la misma.

Figura 2.4 Página índice.

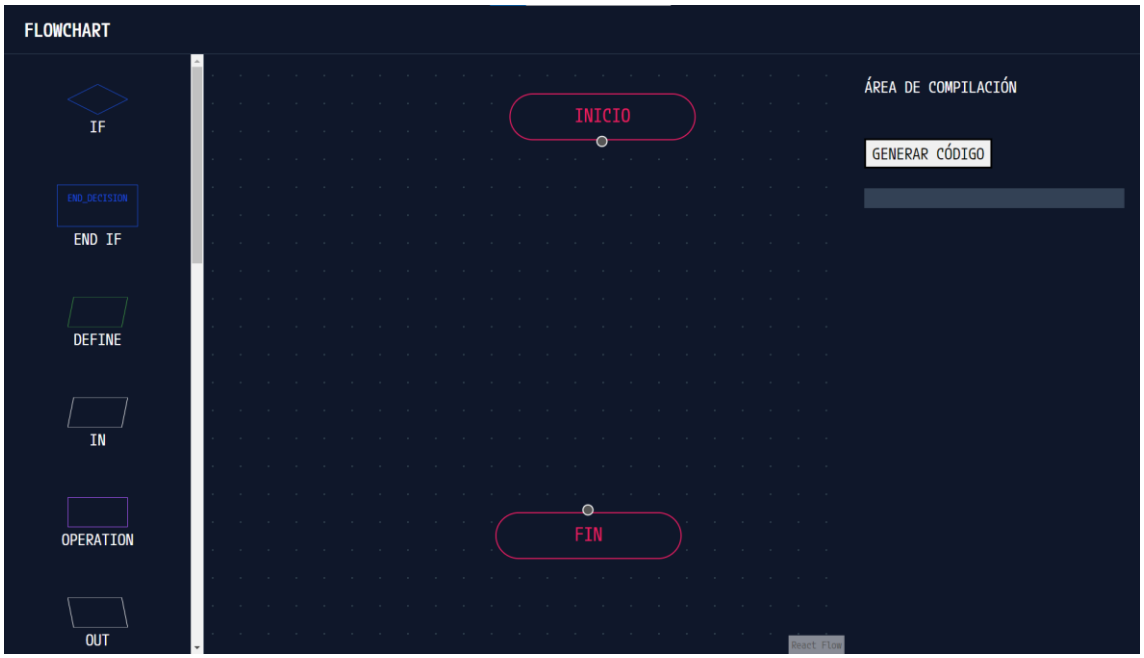


Figura 2.5 Módulo de Programación.

2.1.7. DISEÑO DE BLOQUES

Los bloques de programación están diseñados para ser intuitivos y fáciles de usar para los programadores de todos los niveles. Cada bloque representa una operación o una estructura de control y está diseñado para ser gráfico y visual, lo que lo hace fácil de identificar y usar.

En el proyecto programado en *React*, se han construido las funcionalidades y los bloques de programación de forma independiente, llamando al elemento cuando es

necesario. Para esto, se ha creado la clase Nodo[7], en la cual se definen las cualidades de los nodos a crear o ya creados, incluyendo: la posición de las aristas (puntos usados para unir los nodos) y la obtención del código según lo diseñado en el Diagrama.

La biblioteca *React Flow* se utiliza para construir cada uno de los bloques, permitiendo controlar la posición y el tamaño de los nodos a través de los efectos de construcción.

La construcción del bloque ***IF* o Decisión** se representa mediante un rombo con 3 nodos, como se ve en la **Figura 2.6**. El primer nodo se encuentra en la parte superior y está conectado con un bloque anterior, mientras que los nodos de los costados derecho e izquierdo representan los posibles caminos que puede tomar la sentencia ***IF***.

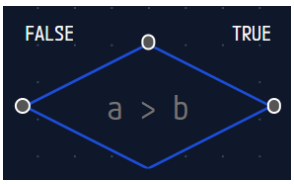


Figura 2.6 Bloque IF.

Como se puede ver en la **Figura 2.6**, la forma de completar datos en bloques comparado con una programación más tradicional es mucho más sencilla; además se tiene una ayuda, con la sintaxis de escritura dentro del bloque como se observa en la **Figura 2.7**.

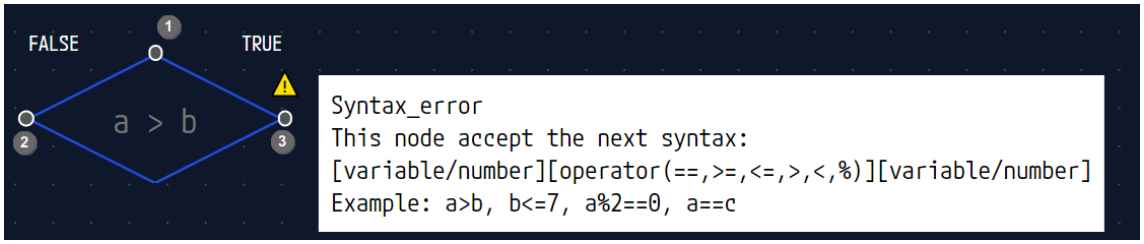


Figura 2.7 Nodo IF con instrucción de uso.

A continuación, en el **Código 2.1** se muestra la forma de codificación del bloque IF, en las líneas 1 a 3 se importan librerías necesarias y el estilo que llevará el bloque. Las líneas 4 a 13 se utiliza '*export default function'* que en *react* permite la exportación de una función específica (DecisionNode[8] en este caso) como el componente principal de un módulo, permitiendo su uso e importación en otras partes de la aplicación de manera sencilla y clara, facilitando la organización y reutilización de código en el desarrollo de

[7] En *ReactFlow*, la clase Nodo es un componente fundamental para la construcción de bloques y diagramas.
[8] DecisionNode función creada para permitir la extracción del código creado a partir del bloque IF.

aplicaciones de *React*, función que permite él envió de los datos que estén dentro del bloque, para extraer su código.

La función creada en las líneas 15 a la línea 20 del **Código 2.1** permite crear una ayuda visual para el estudiante, indicando que tipo de caracteres son aceptados, cual es la manera correcta de realizar una comparación y los operadores admitidos. De la línea 23 a la línea 30 se otorgan las características del polígono, como ID, *placeholder*[9] , nombre y también se establece target (es un concepto clave que se refiere al elemento específico del DOM (*Document Object Model*) sobre el cual se ha producido un evento. Su uso permite a los desarrolladores detectar y responder a interacciones del usuario en la aplicación de manera precisa y eficiente, permitiendo una mayor flexibilidad en la implementación de funcionalidades y mejorando la experiencia del usuario final).

```
1  import "./styles.css";
2  import { Handle, Position } from "React-Flow-renderer";
3  import { useEffect, useState } from "React";
4  export default function DecisionNode({ data }: any) {
5    const [error, setError] = useState(false);
6    function onExpressionChange(value: string) {
7      data.node._code = value;
8      checkSyntax(value);
9    }
10
11   useEffect(() => {
12     data.node._type = "decision";
13   });
14
15   function checkSyntax(value: string) {
16     const regex =
17       /([a-z])(>|<|==|<=|>=)([a-z]|\d)|([a-z])(%)([a-z])(==)([a-
18 z]|\d)/i;
19     setError(!regex.test(value));
20   }
21   return (
22     <div className="decision">
23       <input
24         id="input-data"
25         placeholder="a > b"
26         className="input-data-node"
27         onChange={({ target }) => {
28           onExpressionChange(target.value);
```

[9] Placeholder texto predeterminado en contenedores de tipo Input.

```
29          }}
30        />
31        <p className="decision-opt-l">FALSE</p>
32        <p className="decision-opt-r">TRUE</p>
33        <svg
34          style={{ position: "absolute", top: 0, left: 0, right: 0,
35  bottom: 0 }}
36          viewBox="0 25 100 50"
37          xmlns="http://www.w3.org/2000/svg"
38        >
39          <polygon
40            points="0,50 50,25 100,50 50,75"
41            fill="#0f172a"
42            stroke="#1D4ED8"
43          />
44        </svg>
45        {error && (
46          <div className="syntax-error">
47            <div className="error-hint">
48              <p>Syntax_error</p>
49              <p>This node accept the next syntax:</p>
50              <p>
51
52  {"[variable/number][operator(==,>=,<=,>,<,%)][variable/number]"}
53              </p>
54              <p>{"Example: a>b, b<=7, a%2==0, a==c"}</p>
55            </div>
56          </div>
57        )}
58        <Handle type="target" position={Position.Top} />
59        <Handle type="source" id="a" position={Position.Right} />
60        <Handle type="source" id="b" position={Position.Left} />
61      </div>
62    );
63  }
```

Código 2.1 Construcción del bloque IF.

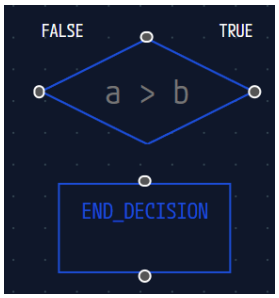
En la línea 31 y 32 del **Código 2.1** se definen los dos caminos que va a tomar el bloque y desde la línea 36 hasta la 43 se definen los puntos de los cuales se forma el polígono, para el caso de IF es un rombo.

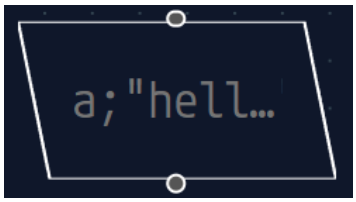
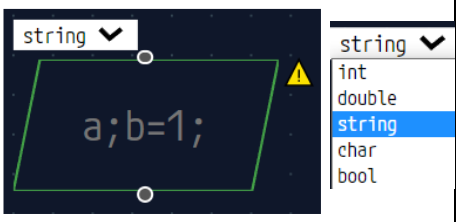
En la construcción de un polígono, este se utilizará en la sección de programación cuando sea necesario. Para la creación del polígono es fundamental que existan puntos definidos como nodos, los cuales serán conectados para formar el diagrama de flujo. Las aristas se generan en base a las líneas 58-60 del **Código 2.1.**

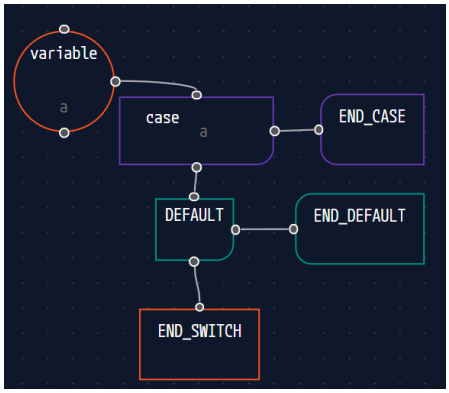
La programación del intérprete se realiza para que pueda analizar y mantenerse dentro del camino correspondiente del bloque IF. Esto se logra mediante dar un *click* en el bloque que se necesite, permitiendo insertar el bloque en el área de programación cada vez que sea necesario.

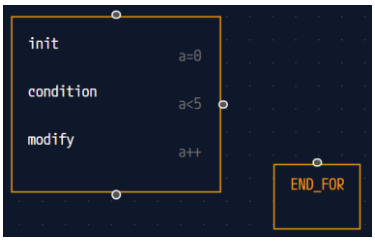
La forma en que se ha desarrollado este bloque de programación es similar a como se desarrollan los demás bloques, los cuales varían en número de nodos según la funcionalidad, y estos nodos son unidos entre sí, y cada uno de los bloques representan código en lenguaje C/C++, a continuación, en la **Tabla 2.12** se detalla cada uno de estos:

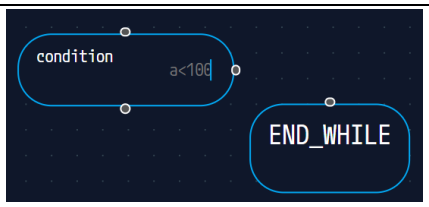
Tabla 2.12 Bloques diseñados.

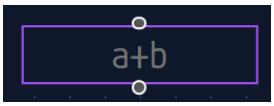
Bloque	Figura y código que representa	Descripción
INICIO y FIN	 1 #include <iostream> 2 #include <math.h> 3 using namespace std; 4 int main(){	Se muestran los bloques principales dentro del módulo de programación, los cuales son representados con un Rectángulo con esquinas redondeadas de color rojo. Una vez construido los bloques se asignan fragmentos de código se complementan uniendo más bloques.
IF		Para el uso del bloque IF se debe complementar con el bloque END IF. El bloque de IF permite el ingreso de condiciones que se analizan si se cumplen o no y elige un camino u otro.
IN		En este bloque que se observa que permite el ingreso de los datos por teclado necesario para un ejemplo, diseñado, dentro del bloque se tiene el código “std::cin>>a;”, el cual se refleja una vez se haya llenado algo dentro del bloque.

OUT		Este bloque *OUT* se lo puede utilizar como un bloque de impresión, es decir que muestra el valor que se escriba dentro del mismo, si está dentro de comillas se muestra el texto, si no es así se muestra el valor que tiene la variable escrita.
DEFINE		En este bloque se encuentra la parte de definición y asignación de las variables, es decir que se define una variable puede ser lo que se observa. Selecciona alguno de estos tipos de datos y se asigna el tipo de dato que se va a agregar.

SWITCH-CASE	
	`switch(){`
	`case :`
	
	`break;`
	`default:`
	
	`}`
		Bloque de SWITCH se complementa con los bloques tanto de CASE, DEFAULT y sus respectivos END, que finaliza el código, y obtiene el código como se observa.

FOR	
	`for(int i = valor inicial; i <= valor final; i = i + paso)`
	`{`
	`  Bloque de Instrucciones....`
	`}`
		Bucle de FOR, construido y representado con los siguientes bloques, en el bloque más grande se observa que únicamente es necesario agregar los parámetros necesarios (inicio, condición y salto) y unir los bloques, en los nodos que se observan, entre el bloque más grande y el de cierre, se unen bloques que de esta manera se encontrarían dentro del bucle. De esta manera se evita escribir todo el código mostrado y es más sencillo de entender.

WHILE	
	`while(condicion de finalizacion) //por ejemplo número == 100`
	`{  Bloque de Instrucciones....`
	`//ejemplo número++;`
	`}`
		Bucle While, en este bucle trabaja de la misma manera que FOR, con la diferencia que la variable crece dentro del bucle como se observa, que dentro de los parámetros del bucle no tiene un valor de salto para la variable de la que depende, sino que da la posibilidad de utilizar una condición para finalizar el bucle.

OPERACIO N		Es el bloque responsable de recibir las instrucciones, es decir, las operaciones matemáticas y lógicas dentro del sistema, representado con un rectángulo como se observa.

2.2. IMPLEMENTACIÓN

En esta fase se detallan todos los pasos realizados para la codificación del prototipo, en el cual se han utilizado además de programas, librerías que permitirán el desarrollo de cada una de las capas del programa, y la actualización del Tablero Kanban.

2.2.1. ACTUALIZACIÓN DEL TABLERO KANBAN.

La **Figura 2.17** muestra las tareas que ya se han completado con éxito y las que aún están en progreso. Una vez que se hayan completado todas las tareas, se pasará a la fase final del Trabajo de Integración Curricular, que incluye las pruebas del prototipo creado.

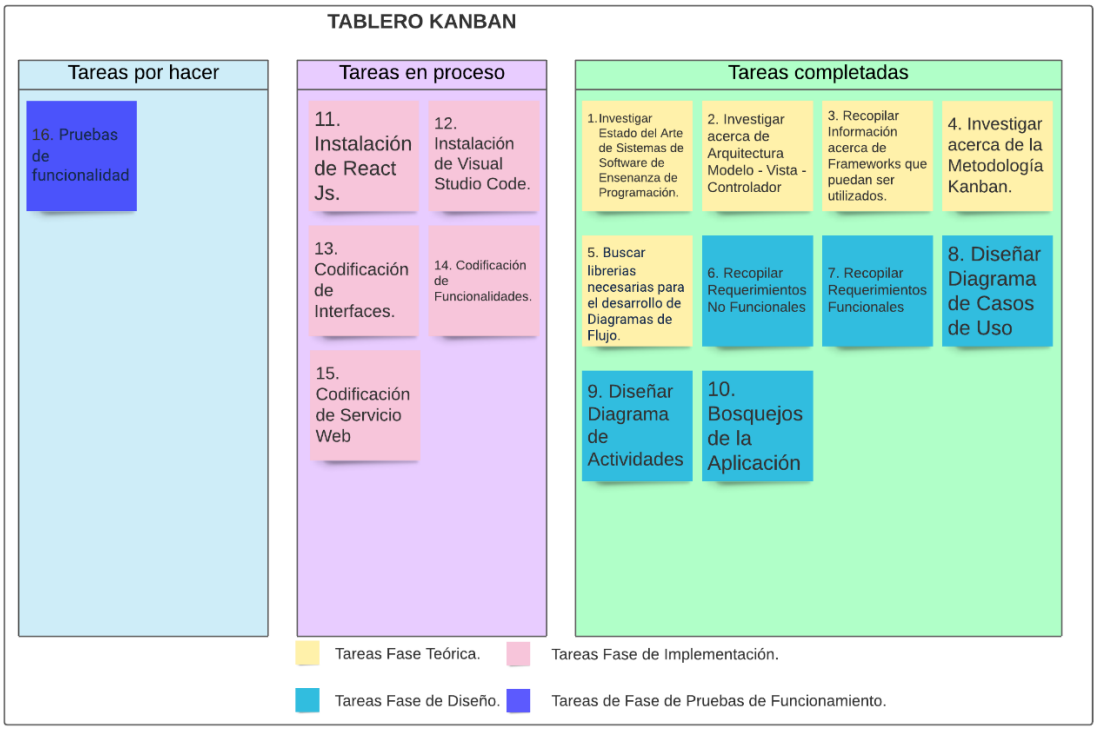


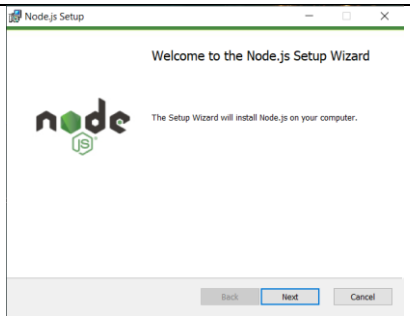
Figura 2.8 Actualización del Tablero Kanban en la Etapa de Implementación.

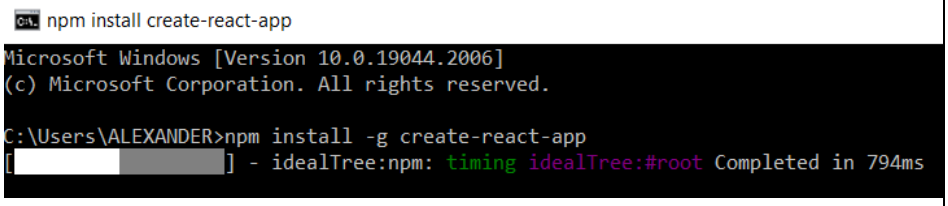
2.2.2. INSTALACIÓN DE HERRAMIENTAS NECESARIAS

Para la implementación se va a utilizar aplicaciones que serán instaladas de acuerdo con la **Tabla 2.13:**

Tabla 2.13 Pasos de instalación de Herramientas a usar.

Aplicación	Pasos de instalación.
Node.js	1. Descargar e instalar Node.js desde su página oficial: https://nodejs.org/en/download/ 2. Dar doble *click* sobre el archivo descargado y proceder con la instalación como se observa, dejando todos los parámetros por defecto de la instalación.

<table>
<tr><td></td><td> 

<ol start="3">
Verificar que la instalación fue exitosa abriendo una terminal y escribiendo "node -v" para obtener la versión de Node.js instalada.
También puede utilizar un administrador de paquetes de Node.js como nvm, navegando a https://github.com/nvm-sh/nvm</td></tr>
<tr><td>React.js</td><td>Una vez concluido el proceso de instalación de Node Js, se puede realizar la instalación de React Js, el cual necesita de Node Js como prerrequisito, la instalación de React Js se lo realiza con CMD, con el comando “npm install create-React-app” y procederá a la instalación como se muestra a continuación.
 

Esto creará una estructura de carpetas y archivos básicos para el proyecto React.
Dentro del directorio del proyecto escribir en la terminal npm start</td></tr>
<tr><td>Visual Studio Code</td><td>Descarga la última versión de Visual Studio Code desde la página oficial en https://code.visualstudio.com/download
Ejecutar el archivo de instalación descargado. En Windows, esto significa hacer doble click en el archivo .exe descargado.
Seguir las instrucciones que aparecen en el recuadro de instalación.
Una vez que la instalación esté completa, se podrá ejecutar Visual Studio Code haciendo click en el icono correspondiente en el menú Inicio (Windows).
La gran ventaja que se tiene por el uso de este editor de código es que se pueden instalar extensiones muy útiles para facilitar el uso.</td></tr>
</table>

G++	1. Descargar el instalador de GCC para Windows desde un sitio web oficial o fuente confiable. 2. Ejecutar el instalador y seguir las instrucciones del asistente de instalación. 3. Añadir la ruta de instalación de GCC a la variable de entorno PATH

.

2.2.3. IMPLEMENTACIÓN DE LA CAPA CONTROLADOR

La Capa Controlador en un modelo MVC, es responsable de recibir las solicitudes del usuario a través de la interfaz de usuario, procesarlas y luego enviar las respuestas apropiadas al Modelo o a la Vista. Recibiendo los datos de la interfaz gráfica mostrada en la **Figura 2.5;** recibe los valores escritos dentro de los bloques, obtiene el código y lo compila, mostrando el resultado de la programación.

La Capa Controlador de la aplicación se aplica principalmente en dos componentes importantes del prototipo, los cuales son: bloques de programación (cada uno de ellos) y servidor utilizado para la compilación de los ejemplos realizados en la interfaz.

2.2.3.1. Desarrollo del Módulo de Programación

Para empezar con la construcción del prototipo, se debe iniciar el servidor de la aplicación, como se observa en la **Figura 2.9**, en los cuales se van a enviar los datos necesarios para la ejecución de los datos.

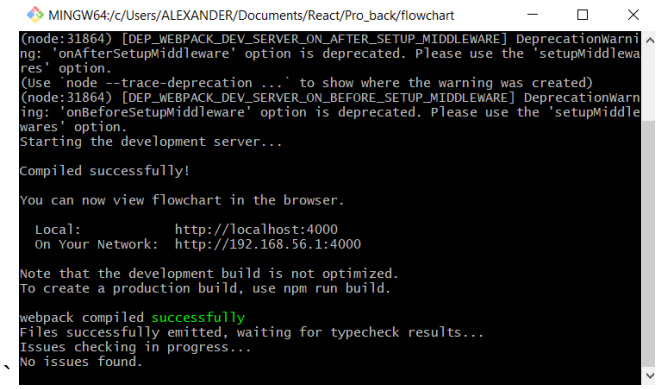


Figura 2.9 Encendido del Servidor de la aplicación.

Con los bloques ya construidos, se necesitan desarrollar las funciones que permitirán obtener el código, compilar el programa y su posterior ejecución, los cuales se describen en los siguientes apartados.

2.2.3.2. Desarrollo de API de Servidor Para el uso de G++ y Compilación del Prototipo.

Una vez construido el módulo de programación para permitir la “compilación” se comienza con la creación de la *API*. Utilizando *Child Process* recepta lo generado del Diagrama de Flujo diseñado en la aplicación web y envía el código generado a la aplicación G++ para proceder con la ejecución del código y obtener los resultados de la compilación de datos.

En el **Código 2.2** exporta dos funciones: **resolveRequest** y **saveFile**, y dos funciones adicionales **compileCode** y **executeCode** que se utilizan en **resolveRequest**.

La función **resolveReques**t es una función asíncrona que se utiliza para resolver una solicitud HTTP que contiene código fuente y, opcionalmente, datos de entrada. En primer lugar, la función llama a la función **saveFile** para guardar el código fuente en un archivo llamado _**prog.cpp**. Si se proporcionan datos de entrada, la función también llama a **saveFile** para guardar los datos de entrada en un archivo llamado **input.txt.**

A continuación, en la línea 14 la función **resolveRequest** llama a **compileCode** para compilar el código fuente utilizando el compilador g++. Si se produce un error al compilar, la función rechaza la promesa con una descripción del error.

Luego, en la línea 14 a la 18 la función llama a **executeCode** para ejecutar el programa compilado. Si se proporcionan datos de entrada, la función llama a **executeCode** con un indicador para leer los datos de entrada desde el archivo input.txt. Si no se proporcionan datos de entrada, la función llama a **executeCode** sin indicador.

Finalmente, la función responde a la solicitud HTTP con el resultado de la ejecución del programa compilado, que se envía como una respuesta JSON.

La función **saveFile** se utiliza para guardar el código fuente y los datos de entrada en archivos. Esta función utiliza la función **writeFile** del módulo **fs** de Node.js para escribir datos en un archivo.

La función **compileCode** utiliza la función **exec** del módulo **child_process** de Node.js para ejecutar un comando que compila el archivo _prog.cpp utilizando el compilador g++. Si se produce un error al compilar, la función rechaza la promesa con una descripción del error.

La función **executeCode** utiliza la función **exec** del módulo **child_process** de Node.js para ejecutar el archivo compilado _**prog**. Si se proporciona el indicador para leer los datos de entrada, la función redirige la entrada estándar desde el archivo input.txt. Si se

produce un error al ejecutar el archivo, la función rechaza la promesa con una descripción del error.

```
import { exec } from "node:child_process";
import fs from "fs";

export async function resolveRequest(req, res) {
  try {
    //saveCodeFile
    await saveFile("_prog.cpp", req.body.code);

    if (req.body.input) {
      await saveFile("input.txt", req.body.input);
    }

    //compile
    await compileCode();

    let result;

    if (req.body.input) {
      result = await executeCode(true);
    } else {
      result = await executeCode(false);
    }

    res.json(result);
  } catch (error) {
    console.log(error.err.toString());
    res
  .status(500)
  .json({ result: { message: "Internal Server error", details: error } });
  }
}

export async function saveFile(name, data) {
  return new Promise((resolve, reject) => {
    fs.writeFile(name, `${data}`, (err) => {
      if (err) {
        console.log(err);
        reject();
      }
      resolve();
    });
  });
}

async function compileCode() {
  return new Promise((resolve, reject) => {
    exec("g++ -o _prog _prog.cpp", (err, stdout, stderr) => {
      if (err) {
        console.log(err);
        reject({ err, stderr });
      }
      resolve();
    });
```

```
  });
}

async function executeCode(input) {
  return new Promise((resolve, reject) => {
exec(input ? "_prog < input.txt" : "_prog", (err, stdout, stderr) => {
      if (err) {
        console.log(stderr);
        reject({ err, stderr });
      }
      console.log(stdout);
      resolve({ result: stdout });
    });
  });
}
```

Código 2.2 Servidor para el llamado a G++.

Una vez terminada la implementación de la API se inicia el servidor con el comando "npm start" presentado en la **Figura 2.10.**

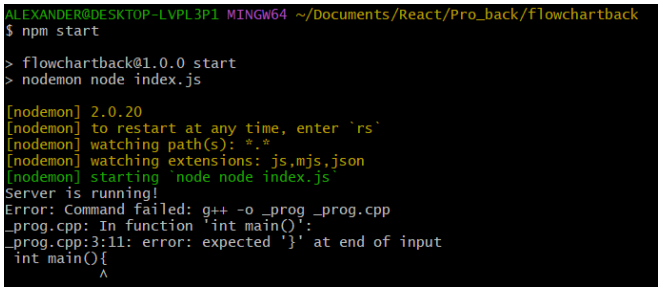


Figura 2.10. Llamado a la API del Prototipo.

Con el API construido se necesita poder extraer el código y posteriormente enviar el código para su compilación.

2.2.3.3. Función para generar código.

Para la obtención de código en el proyecto es necesario declarar una variable global la cual obtiene el código de los bloques, sin importar cuantos bloques sean o si están dentro de otros bloques, la función, crea un ID dependiendo como han sido unidos, sin importar cual bloque haya sido llamado primero al área de programación, como se muestra en el **Código 2.3.** es una expresión de mapeo (mapping[10]) de un array de nodos llamado **nodes** utilizando el método **map()**. El objetivo de esta expresión es encontrar los nodos de origen y destino de una conexión (connection) específica.

[10] Expresion de mapeo es una técnica útil para transformar los elementos de un array y generar un nuevo array con los resultados de la transformación.

Dentro de la función de mapeo, se comprueba si el **id** del nodo **_node** es igual al **source** o **target** de la conexión. Si **_node.id** coincide con **connection.source**, se establece el valor de **nodeSource** como **_node**. Si **_node.id** coincide con **connection.target**, se establece el valor de **nodeTarget** como **_node**.

```
    nodes.map((_node) => {
      if (_node.id === connection.source) {
        nodeSource = _node;
      }
      if (_node.id === connection.target) {
        nodeTarget = _node;
      }
      return null;
    });
```

Código 2.3 Identificador de Nodos

En el **Código 2.4** se obtienen las expresiones de los bloques, en conjunto al código de cada bloque, que se observan en el **apartado 2.1.7** la función asíncrona llamada **getExpressions()** que se utiliza para obtener una secuencia de expresiones a partir de un conjunto de nodos de una interfaz gráfica.

Primero, la función reinicia la variable global **myscript** que se utiliza para almacenar la secuencia de expresiones. Luego, la función recorre todos los nodos de la interfaz gráfica y, si encuentra un nodo de tipo "**enddecision**", establece su atributo **_decisionway** en "*true*".

A continuación, la función llama al método **getScript()** del primer nodo de la interfaz gráfica. Este método recorre recursivamente los nodos conectados al nodo actual y genera una secuencia de expresiones que se almacenan en la variable global **myscript**.

Después de obtener la secuencia de expresiones, la función la establece en un estado local de la página web llamado **outputScript**. La función también determina si hay nodos de entrada en la interfaz gráfica y establece un estado local llamado **hasInput** en consecuencia.

```
async function getExpressions() {
    global.myscript = "";

    //restart enddecision nodes way
    nodes.filter((node) => {
      if (node.type === "enddecision") {
        node.data.node._decisionway = "true";
      }
    });

```

```
    nodes[0].data.node.getScript();

    setOutputScript(global.myscript);

    const inputNodes = nodes.filter((node) => node.type === "in");

    setHasInput(inputNodes.length);
  }
```

Código 2.4 Función de obtención de código.

2.2.3.4. Función para compilar código.

Una vez se obtenido el código se procede con la compilación, para lo cual el código obtenido es enviado al servidor por medio de un método Post como se observa en el **Código 2.5**.

En el análisis del **Código 2.5**, es una función asíncrona llamada **resolveCode()** que se utiliza para ejecutar código en un servidor remoto utilizando una API. El código envía una solicitud *POST* a la URL "**http://localhost:3000/execute**" con el código que se va a ejecutar y, opcionalmente, los valores de entrada para el código. Después de recibir la respuesta del servidor, el resultado se convierte en un objeto JSON y se muestra en una ventana en la página web del prototipo.

La función maneja errores que pueden ocurrir durante la ejecución del código. Si se produce un error, el mensaje de error se mostrará en la ventana junto con un mensaje predeterminado que indica que hubo un error al compilar el código. Además, la función establece una variable de estado ***isExecuting*** que indica si la solicitud de ejecución está en progreso o no. Por otro lado, si el código es ejecutado de manera exitosa, se muestra el resultado de la ejecución, lo cual indica que el sistema está funcionando de manera adecuada y no existen problemas en la compilación y ejecución del código.

```
async function resolveCode() {
    try {
      setIsExecuting(true);

      let body = { code: global.myscript };

      if (hasInput) {
        let _ivalues = inputValue.split(";");

        let _stringValue = "";

        _ivalues = _ivalues.filter((item) => {
          if (item !== "") {
            _stringValue = `${_stringValue}${item}\n`;
```

```
            }
          });

          //@ts-ignore
          body["input"] = _stringValue;
        }

        const reponse = await fetch("http://localhost:3000/execute",
{
          headers: { "Content-Type": "application/json" },
          method: "POST",
          body: JSON.stringify(body),
        });
        const result = await reponse.json();

        if (result.result.details) {
          throw result.result.details.stderr;
        }

        let _regex = result.result.replace(/\n/g, "<br/>");

        _regex = _regex.replace(/ /g, " ");
        setExecuteResult(_regex);
      } catch (error) {
        setExecuteResult(
          `There was an error when trying to compile the
code.\n[Compiler error]:\n${error}`
        );
        console.log(error);
      } finally {
        setIsExecuting(false);
      }
    }
```

Código 2.5 Ejecución de código generado.

2.2.4. IMPLEMENTACIÓN DE LA CAPA VISTA

Gracias a que *React* es un *Framework* modular se construyen los nodos por separado y una vez terminados todos los elementos se unen en la aplicación final, para la creación de una página web final, dividida en áreas como el menú, programación, compilado y ejecución, como se ilustra en la **Figura 2.11.**

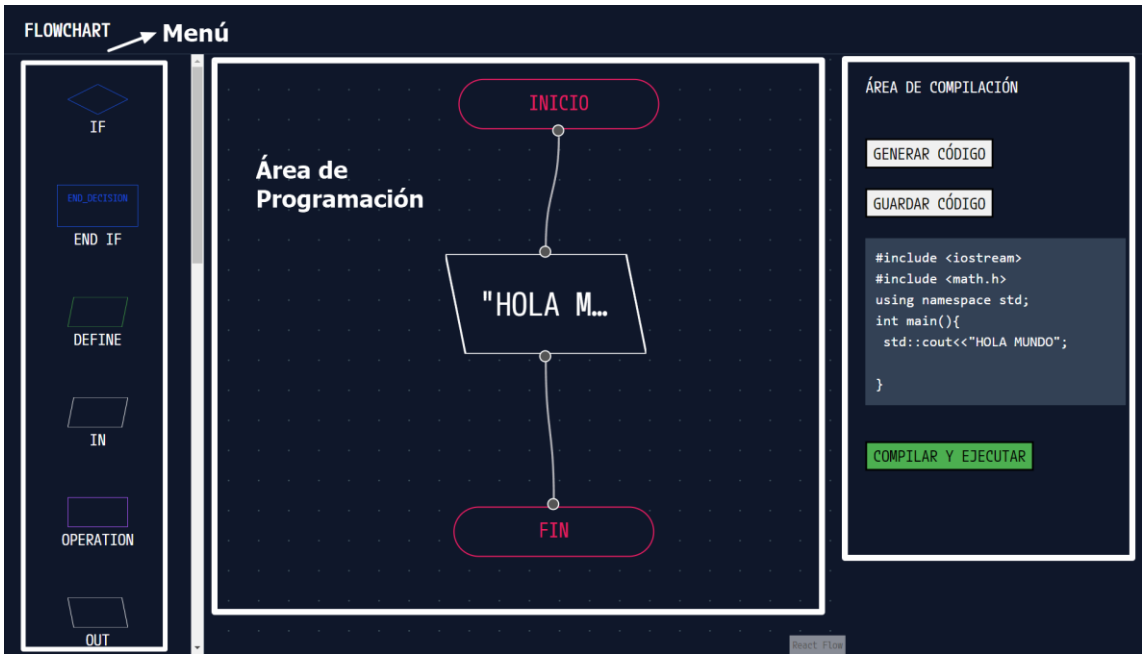


Figura 2.11 Áreas del Módulo de Programación.

En la **Figura 2.12** se encuentran todos los bloques de Programación que existen en el menú de la **Figura 2.11** que son llamados al área de programación al darle *click* sobre alguno de ellos.

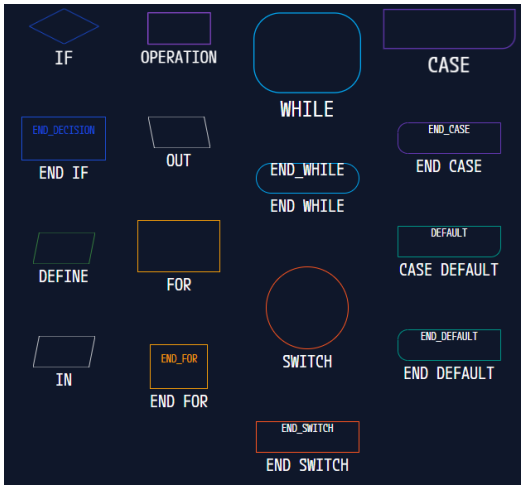


Figura 2.12 Bloques de Programación.

En la **Figura 2.13** muestra el espacio que comprende al Área de Programación, en este espacio van a aparecer los bloques cliqueados para posteriormente se unidos y llenados con los datos requeridos para el programa.

Figura 2.13 Área de Programación.

Cuando ya haya sido creado el Diagrama de Flujo se procede al Área de Compilación a generar el código que este genera, el cual será obtenido al darle *click* en “GENERAR CÓDIGO” botón que se observa en la **Figura 2.14** y posteriormente aparecerá la opción de ejecutar como se muestra en la **Figura 2.15.**

Figura 2.14. Generar de Código.

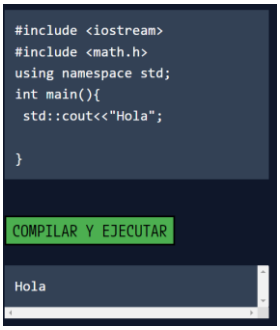


Figura 2.15. Compilación de Código

Como demostración del funcionamiento del prototipo, se ha resuelto un ejercicio que, ingresando dos variables por defecto, a=5 y b=6, encontrar la suma de a y b, como se ve en la **Figura 2.16**, se utiliza un bloque “IN” para ingresar las variables, uno de “*OPERATION*” para realizar la asignación matemática de la variable c y un bloque “*OUT*” para imprimir el valor obtenido.

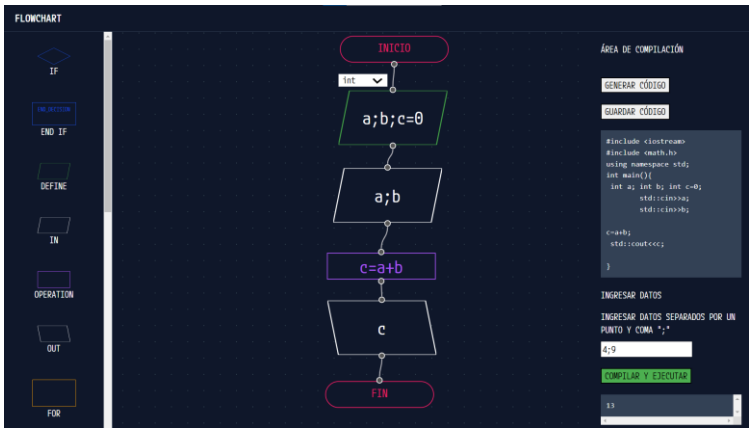


Figura 2.16 Resultado de ejecución.

Una vez realizadas todas las modificaciones pertinentes, se ha llegado al bosquejo final de la aplicación, teniendo como página índice la siguiente **Figura 2.17**, en la parte del Módulo de Programación se tiene la **Figura 2.18** su interfaz principal.

Figura 2.17 Página principal del Prototipo.

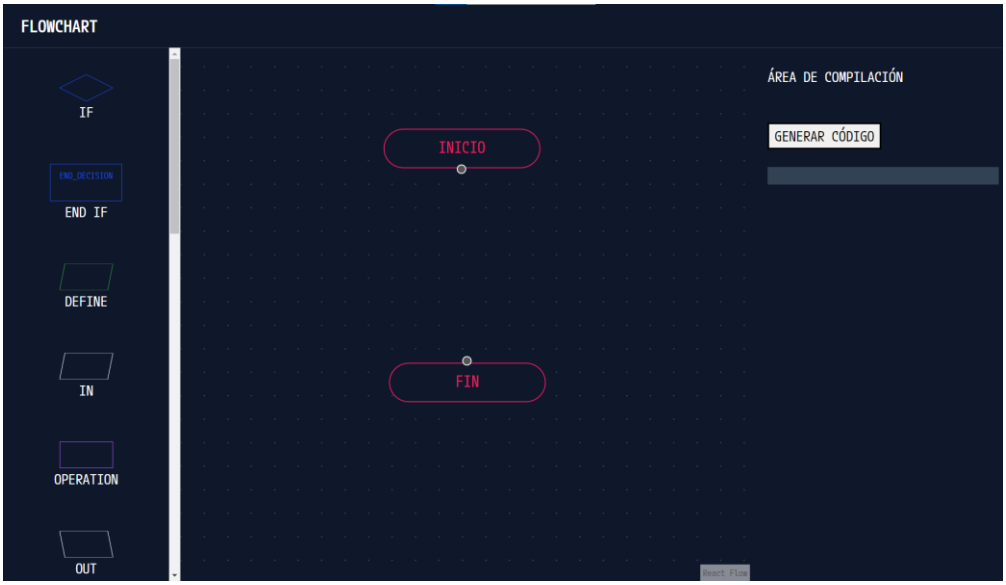


Figura 2.18 Módulo de Programación.

3. RESULTADOS Y DISCUSIÓN

En el presente Capítulo se mostrarán los pormenores de las pruebas realizadas en el prototipo y los resultados obtenidos tras finalizar la etapa de implementación del prototipo.

A lo largo del **Apartado 3.1**, correspondiente a Resultados se actualizará el tablero Kanban, se definirá el entorno de pruebas, la validación de Requerimientos Funcionales y No Funcionales. Siguiente, en el **Apartado 3.2** se definirá las conclusiones obtenidas del prototipo y las recomendaciones en base a eventualidades al momento del desarrollo del prototipo.

3.1. RESULTADOS

En esta fase se analizan los resultados de las pruebas ejecutadas en el entorno de pruebas previamente configurado. El objetivo es determinar si el código cumple con los requisitos y si se manejan adecuadamente todas las situaciones previstas.

3.1.1. ACTUALIZACIÓN DEL TABLERO KANBAN

El tablero Kanban se actualiza para indicar la necesidad de completar las pruebas del prototipo desarrollado como se muestra en la **Figura 3.1**. Una vez finalizada esta tarea, el tablero será dado por culminado.

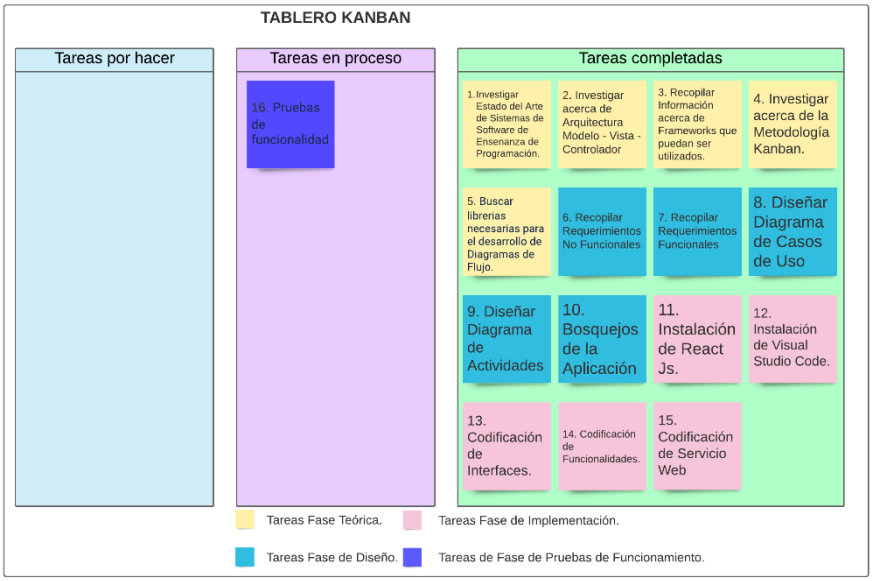


Figura 3.1 Tablero Kanban en etapa de Resultados.

3.1.2. DEFINICIÓN DEL ENTORNO DE PRUEBAS

Para determinar que el prototipo cumple con todos los requisitos, se realizaron pruebas controladas mediante la utilización del prototipo. Estas pruebas fueron diseñadas a lo largo de un mes en conjunto con 10 estudiantes y un profesor del área de Programación de la Universidad San Francisco de Quito (USFQ).

Se presentó 10 ejercicios de la asignatura en forma de código y se pidió a los estudiantes ingresar a la aplicación y tratar de replicar el ejemplo en el prototipo; navegando entre módulos de la aplicación para aprender sobre su funcionamiento. Antes de comenzar a realizar los ejercicios, se impartió una breve inducción sobre el funcionamiento del sistema. Se proporcionó una explicación sobre cómo cada estudiante puede ser autodidacta utilizando el sistema; a continuación, el procedimiento:

- Fue proporcionado al estudiante el sistema, en su página índice, con el fin que naveguen entre los módulos y encontrar una solución.
- A todos los estudiantes fueron entregados los ejercicios a realizar con instrucciones claras y detalladas para cada uno de ellos. Los ejercicios dados estaban organizados en orden de complejidad, comenzando con los ejercicios más sencillos y avanzando hacia los más difíciles.
- Fueron establecidos los plazos para cada uno de los ejercicios.
- Tras la finalización de cada ejemplo se ofrece una retroalimentación y se muestra una solución a la que llego el encargado de las pruebas.
- Después de la finalización de cada ejemplo, se proporciona retroalimentación y se exhibe una solución posible a la que arribó el responsable de las pruebas.
- El instructor evaluó el desempeño de los estudiantes de forma manual en cada uno de los ejercicios. Los criterios de evaluación solo fueron si lograron encontrar una solución sin necesitar de ayuda o solicitando al instructor resolver una duda.
- Con el propósito de determinar si la aplicación satisface su objetivo, se llevará a cabo la entrega de una encuesta al finalizar el proceso.

3.1.3. VALIDACIÓN DE REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

La validación de los requerimientos funcionales fue llevada a cabo a través de un enfoque basado en entrevistas personales con cada uno de los participantes, así como mediante la aplicación de una encuesta general. Se tuvieron en cuenta las observaciones y comentarios proporcionados por los usuarios después de la finalización

de las pruebas del prototipo con el fin de garantizar el cumplimiento adecuado de los requerimientos establecidos y asegurar una experiencia satisfactoria para el usuario final. Estos resultados se detallan en la **Tabla 3.1.**

Tabla 3.1 Validación de los Requerimientos Funcionales.

RF	Descripción	Observación	
RF01	Desarrollar Manual de Usuario		Los estudiantes comentaron que gracias al Manual de Usuario les resultó mucho más simple el aprender sobre los bloques del programa y cómo unirlos con otros bloques.
RF02	Implementar Módulo de Aprendizaje		Se constató que los usuarios fueron capaces de comprender de manera clara la forma en que se deben crear y organizar los diagramas en el módulo de programación. A través de la interfaz intuitiva y las herramientas disponibles, los usuarios pudieron identificar de manera efectiva la ubicación correcta de los bloques, lo cual contribuyó a mejorar la comprensión y legibilidad de los diagramas generados.
RF03	Agregar Bloques de Programación.		Agregar bloques en el área de Programación es un proceso sencillo y que gustó mucho a los estudiantes, pues solo se necesita dar un *click* sobre el bloque que se quiere agregar.
RF04	Modelar Diagrama de Flujo.		Se observó que los usuarios encontraron sencillo el proceso de añadir y unir nuevos bloques en el módulo de programación debido a la amplia variedad de bloques disponibles. Sin embargo, también se identificó que los estudiantes necesitaban ordenar el diagrama para mejorar su presentación visual, aunque esto no afectaba la funcionalidad de este.
RF05	Diseñar Bloques de Programación		Se observó que los estudiantes encontraron que el uso de figuras geométricas en el módulo de programación les permitió comprender de manera sencilla y rápida las funciones y características del lenguaje de programación.

			La representación gráfica de los elementos y conceptos del lenguaje permitió a los estudiantes visualizar de manera clara y comprensible las funciones de cada bloque, mejorando así su comprensión y aprendizaje.
RF06	Obtener de forma ordenada el código		Al terminar los diagramas los estudiantes entendieron de manera satisfactoria como obtener el código y cómo el código se observa de manera correcta y ordenada en la parte derecha del proyecto.
RF07	Compilar código		Se determinó que los estudiantes encontraron el proceso de compilación del programa en el módulo de programación cumple con la función. A través del uso de herramientas y recursos específicos, los estudiantes pudieron obtener el código generado en primer lugar, lo cual les permitió detectar y corregir errores de programación de manera temprana y efectiva.
RF08	Descargar código generado		Una vez finalizado la ejecución del programa creado, se tiene la posibilidad de descargar el código generado en la aplicación y es útil para entender sobre la estructura del lenguaje C/C++.

Para la validación de los requerimientos no funcionales, se llevó a cabo una encuesta de validación por parte de los usuarios de prueba. Además, esta técnica de recopilación de información permitió detectar y corregir problemas o inconvenientes en cuanto a la interacción del usuario con la aplicación, garantizando así una calidad mínima en la interfaz de usuario y el manejo del prototipo como se observa en la **Tabla 3.2.**

Tabla 3.2 Validación de los Requerimientos No Funcionales.

RNF	**Descripción**	**Observación**	
RNF01	Interfaz intuitiva.		Los usuarios manifestaron que, al ser una interfaz sencilla, navegar entre módulos fue una tarea fácil e intuitiva, al igual que el uso del módulo de programación.

RNF02	Usabilidad del prototipo.		Los estudiantes comentaron que, gracias a la interfaz intuitiva, el módulo de programación fue sencillo de utilizar, tanto el agregar un nuevo bloque, como obtener el código y compilar.
RNF03	Asegurar Disponibilidad.		Al ser una aplicación web, puede ser publicada en Internet, teniendo la posibilidad de ser usada en cualquier momento.

3.1.3.1. Resolución de ejemplos tomados en evaluaciones de la materia de programación.

Se realizan varios ejercicios, con el fin de determinar el funcionamiento del Prototipo y determinar la funcionalidad de este. Se plantearon los siguientes ejercicios:

Enunciado 1: Realizar un programa que entregue n valores de la serie de Fibonacci,

Solución: La serie de Fibonacci es una secuencia matemática en la cual cada número es la suma de los dos números anteriores. El primer término de la serie es 0 y el segundo es 1, y los números siguientes son 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, etc. Para la resolución de este ejercicio se necesitan cuatro bloques *IN,* dos bloques *DEFINE,* un bucle *FOR* con su respectivo *END, y 3 bloques OPERATION.* A demás, se deben crear las variables creadas dentro de los bloques, los cuales serán modelados como se observa en la **Figura 3.2.**

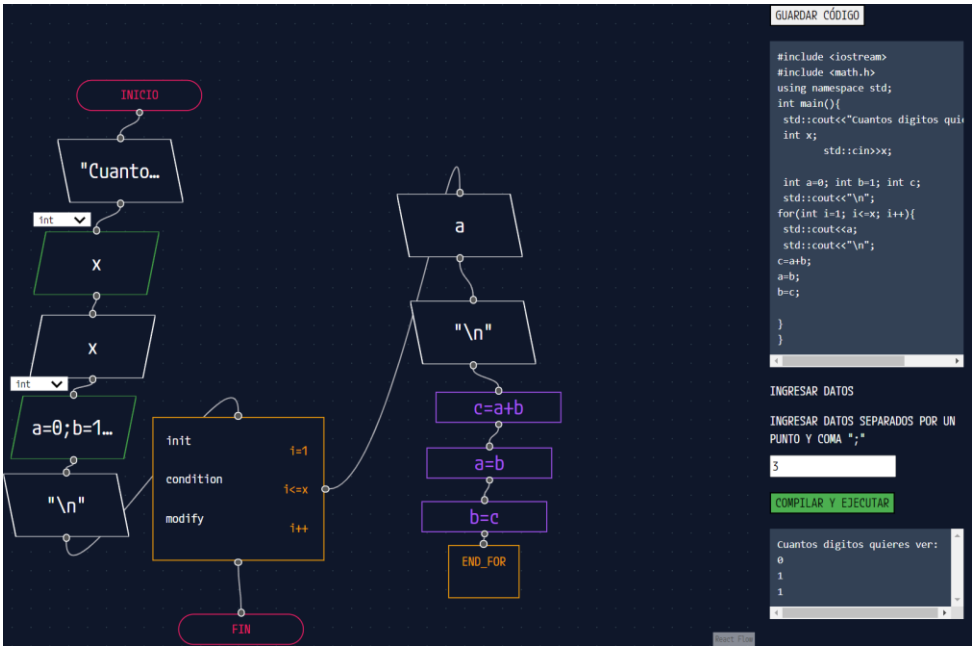


Figura 3.2 Encontrar n valores de la serie de Fibonacci.

Enunciado 2: Realizar un programa que entregue 10 valores de la serie de Fibonacci y descargue el código obtenido previo a la ejecución.

Solución: Para el desarrollo de este ejercicio se necesitan los siguientes bloques: 1 *DEFINE,* 1 *FOR,* 2 *OUT* y 3 *OPERATION.* Se modela el diagrama de la forma que se observa en la **Figura 3.3.** Utilizando las variables a=0, b=1 y c, El bloque de instrucciones (OPERATION), que se utiliza para realizar las operaciones del programa. En este caso, se inicializan los dos primeros valores de la serie de Fibonacci y luego se genera el resto de los valores utilizando un ciclo "*FOR*". En cada iteración del ciclo se calcula el siguiente valor de la serie, se muestra en pantalla y se actualizan los valores de las variables **a** y **b** para el siguiente cálculo, presentando en la variable “a” los valores de la serie. Y para finalizar se descarga el código generado, con el botón “GUARDAR CÓDIGO”

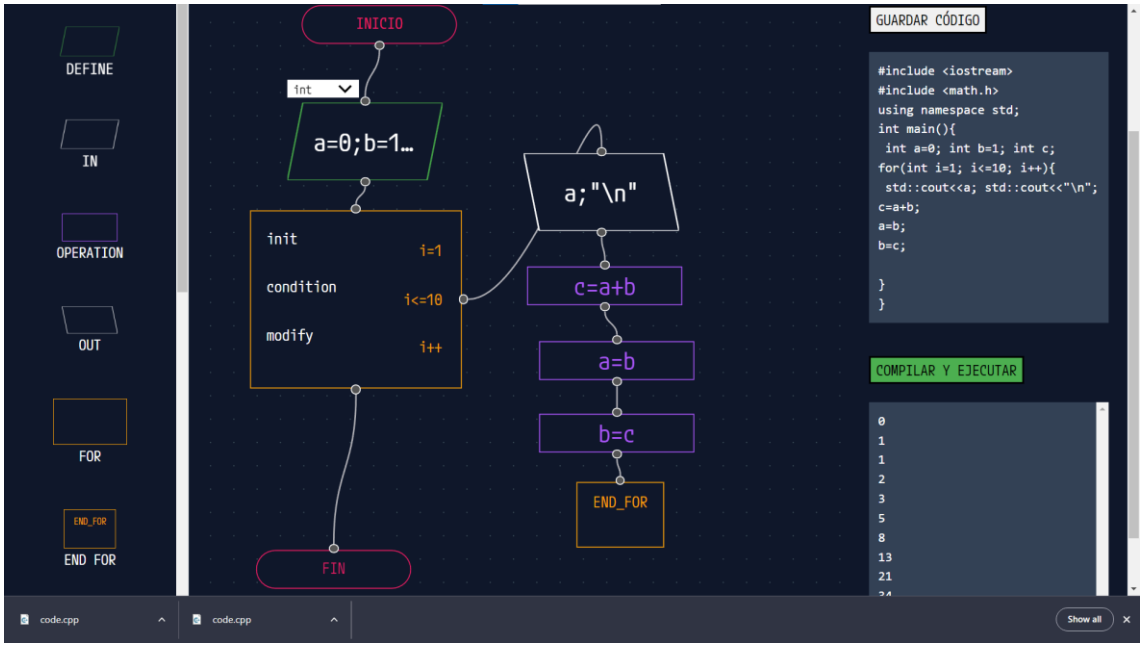


Figura 3.3 Descarga de código.

Enunciado 3: Determinar si un número es primo.

Solución: Para determinar si un número es primo, primero se evalúa que el número ingresado que sea mayor que cero. A continuación, mediante un bucle se determina si el número es primo al verificar si el número es divisible por 1 y por sí mismo. Es decir, solo tiene dos factores, 1 y el propio número. Los números primos son fundamentales en la teoría de números y tienen aplicaciones en criptografía, códigos correctores de errores y otros campos. La solución de este ejercicio se muestra **Figura 3.4.** Se utilizan los siguientes bloques: 3 *OUT, 1 DEFINE, 1 IN, 2 IF y 1 OPERATION,*

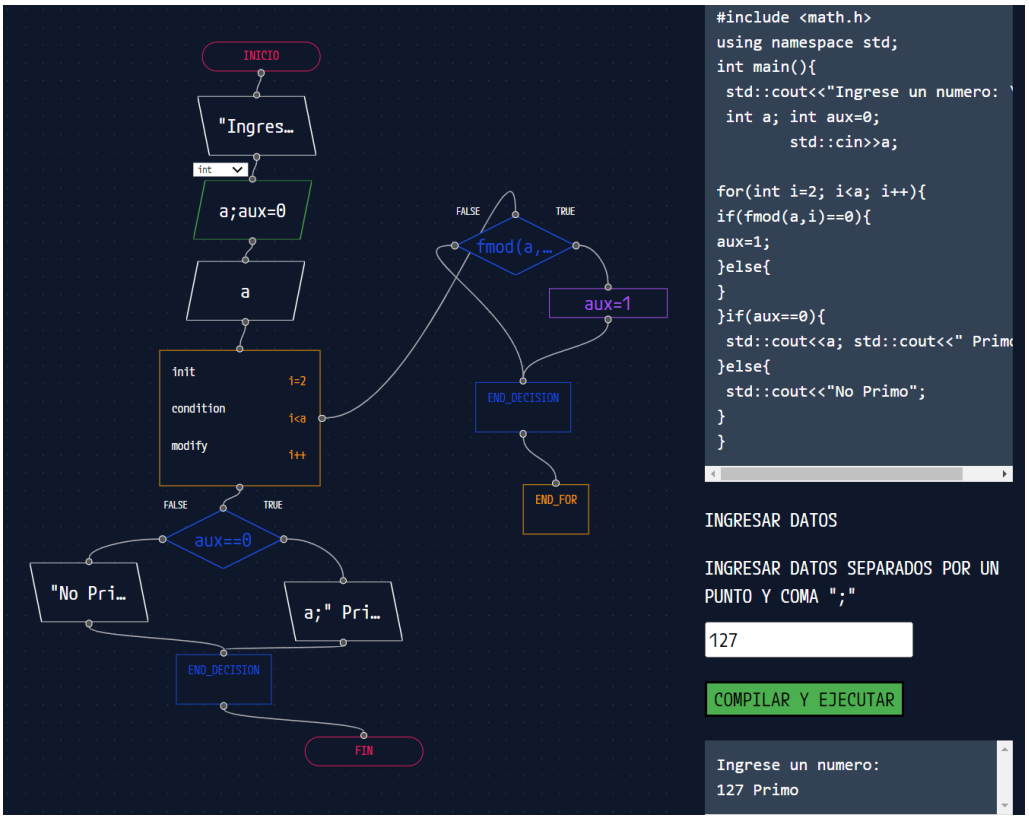


Figura 3.4 Determinar si un número es primo o no.

Enunciado 3: Realizar un programa que muestre la factorial de un numero entero.

Solución: En la **Figura 3.5** se muestra cómo se desarrolla un ejercicio que encuentre la factorial de un número. Para su resolución se van a necesitar los siguientes bloques: 3 *OUT*, 1 *DEFINE, 1 IN, 1 IF, 1 FOR y 1 OPERATION.*

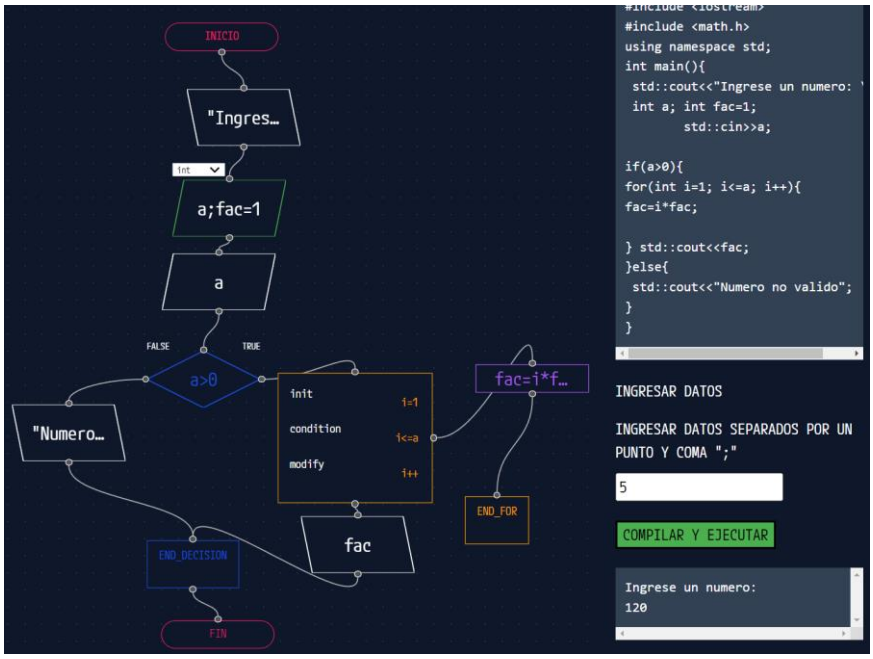


Figura 3.5 Ejemplo de la factorial de un número.

3.1.4. CIERRE DEL TABLERO KANBAN

Una vez que se han completado las pruebas del prototipo y se han solucionado los problemas encontrados durante su ejecución, el tablero Kanban que se muestra en la **Figura 3.6** se cierra, ya que se han cumplido todas las tareas planteadas.

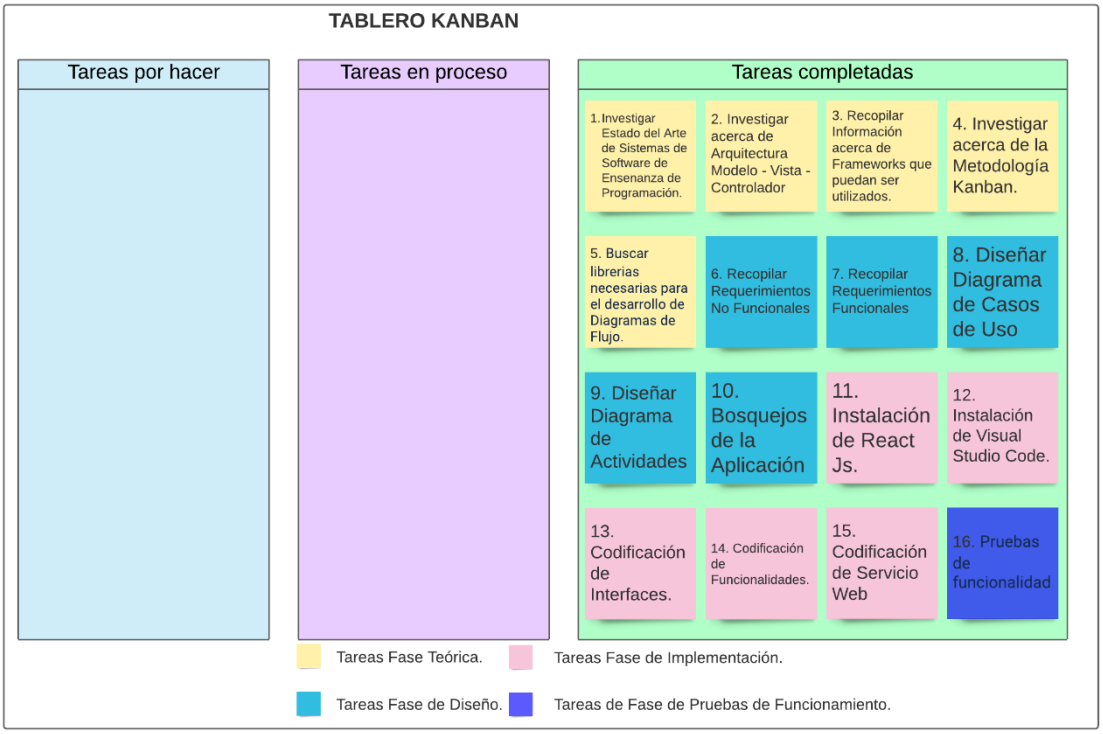


Figura 3.6 Finalización del tablero Kanban.

3.2. CONCLUSIONES Y RECOMENDACIONES.

En el presente apartado se mostrarán los por menores de las pruebas realizadas en el prototipo y los resultados obtenidos una vez concluidas todas las fases de desarrollo del aplicativo.

3.2.1. CONCLUSIONES

En el Trabajo de Integración Curricular, se llevó a cabo el desarrollo de un prototipo mediante un proceso estructurado en cuatro etapas: La Fase Teórica, Fase de Diseño, Fase de Implementación y Fase de Pruebas de Funcionamiento. Este proceso permitió una planificación y ejecución ordenada y controlada del proyecto, garantizando así la

calidad y el cumplimiento de los requerimientos establecidos llegando a las siguientes conclusiones:

- Se desarrolló una aplicación web que facilita el aprendizaje de los estudiantes de programación en lenguaje de C/C++, mediante bloques de código preprogramados y estructurando diagramas de flujo.
- Los requisitos funcionales del prototipo de programación por bloques incluyen el módulo de programación, el módulo de aprendizaje y el manual de usuario. Esto permitió a los usuarios tener una mejor experiencia al usar la aplicación.
- A partir del análisis de las diferentes herramientas, lenguajes de programación y entornos de desarrollo disponibles, se logró construir de manera satisfactoria un sistema que cumple con el objetivo principal de facilitar el aprendizaje de programación. Este sistema utiliza una metodología de desarrollo por bloques, que permite a los estudiantes construir y programar soluciones de manera progresiva y modular, facilitando su comprensión y aprendizaje.
- La construcción modular del prototipo utilizando React permitió una integración más sencilla y evitó el uso excesivo de código, unificando todo lo programado en una sola página. Se utilizaron varias herramientas y tecnologías, incluyendo el patrón Modelo Vista Controlador (*MVC*), *React JS, React Flow, Bootstrap, Child Proccess y G*++, para construir una interfaz intuitiva y fácil de usar.
- A partir de los resultados obtenidos, se llega a la conclusión de que la aplicación cumple con su propósito al facilitar la comprensión de un lenguaje de programación a los estudiantes y proporcionar una perspectiva distinta sobre el aprendizaje de la programación.

3.2.2. RECOMENDACIONES

En este proyecto, se llevó a cabo el desarrollo de un prototipo mediante un proceso estructurado, durante el cual se presentaron varios problemas y desafíos que requirieron soluciones de diversa índole. Estos problemas se presentaron en las distintas fases del desarrollo del prototipo, incluyendo la Fase Teórica, de Diseño, de Implementación y de Pruebas de Funcionamiento. La identificación de estos problemas y la aplicación de soluciones adecuadas permitió superar los obstáculos y garantizar el cumplimiento de los requerimientos establecidos. Además, estos problemas permitieron identificar áreas de mejora en el proceso de desarrollo y diseño del prototipo.

En base a las eventualidades encontradas durante el desarrollo del prototipo, se plantean las siguientes recomendaciones para mejorar el proceso y garantizar una mayor eficacia en el desarrollo de futuros prototipos:

- Con la finalidad de obtener mejores resultados al realizar las pruebas del prototipo es mejor dar una inducción completa del sistema, para que los estudiantes sepan donde se encuentra cada uno de los elementos que pueden utilizar y también para evitar que los estudiantes quieran probar el Módulo de Programación sin antes revisar el funcionamiento del sistema.
- Se recomienda diseñar todos los bloques, funciones y estilos de manera separada durante el desarrollo del prototipo, para facilitar su integración en un solo componente en una etapa posterior. Esto permitirá una mayor flexibilidad en el proceso de desarrollo y permitirá que cada elemento se pueda modificar o ajustar de manera independiente sin afectar el funcionamiento de los demás componentes.
- Se recomienda implementar una metodología efectiva para solucionar errores durante el desarrollo de programas, y utilizar herramientas que faciliten la identificación de dichos errores. Esto permitirá una gestión más eficiente de los problemas que puedan surgir durante la fase de desarrollo y asegurará una mayor calidad del prototipo final.

4. REFERENCIAS BIBLIOGRAFICAS

[1] "¿Qué es la programación por bloques? - Crack The Code". https://blog.crackthecode.la/programacion-en-bloques (consultado el 15 de enero de 2023).

[2] "Qué es *React*. Por qué usar *React*". https://desarrolloweb.com/articulos/que-es-*React*-motivos-uso.html (consultado el 16 de octubre de 2022).

[3] "Introduction to *React* Flow". https://*React*flow.dev/docs/introduction/ (consultado el 16 de octubre de 2022).

[4] "Custom Nodes Guide - *React* Flow". https://*React*flow.dev/docs/guides/custom-nodes/ (consultado el 16 de octubre de 2022).

[5] "Qué es MVC". https://desarrolloweb.com/articulos/que-es-mvc.html (consultado el 15 de enero de 2023).

[6] "Modelo vista controlador (MVC). Servicio de Informática ASP.NET MVC 3 Framework". https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html (consultado el 13 de octubre de 2022).

[7] "Elements of MVC in *React*. Let's discover the original MVC pattern… | by Daniel Dughila | The Startup | Medium". https://medium.com/swlh/elements-of-mvc-in-*React*-9382de427c09 (consultado el 16 de octubre de 2022).

[8] "¿Qué es Kanban? Principales características y funciones". https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban (consultado el 13 de octubre de 2022).

[9] "¿Qué es un tablero kanban? | Atlassian". https://www.atlassian.com/es/agile/kanban/boards (consultado el 13 de octubre de 2022).

[10] "WhatIsCodingDojo - Coding Dojo". https://codingdojo.org/practices/WhatIsCodingDojo/ (consultado el 16 de octubre de 2022).

[11] "¿Qué es PseInt? - Aprender a programar PRO!!" https://www.aprenderaprogramar.pro/2020/04/que-es-pseint.html (consultado el 19 de enero de 2023).

[12] "PSeInt: ¿Qué es? Descargar y usar | Tecnología + Informática". https://www.tecnologia-informatica.com/pseint/ (consultado el 19 de enero de 2023).

[13] J. E. Beúnes Cañete, A. Vargas Ricardo, J. E. Beúnes Cañete, y A. Vargas Ricardo, "La introducción de la herramienta didáctica PSeInt en el proceso de enseñanza aprendizaje: una propuesta para Álgebra Lineal", *Transformación*, vol. 15, núm. 1, pp. 147–157, 2019, Consultado: el 19 de enero de 2023. [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2077-29552019000100147&lng=es&nrm=iso&tlng=es

[14] "Diseño 2D y 3D: TinkerCAD Codeblocks". https://formacion.intef.es/catalogo/mod/book/view.php?id=70&chapterid=406 (consultado el 16 de octubre de 2022).

[15] “Más información sobre Codeblocks | Tinkercad”. https://www.tinkercad.com/learn/codeblocks (consultado el 16 de octubre de 2022).

[16] “Flexbox Froggy - Un juego para aprender CSS flexbox”. https://flexboxfroggy.com/#es (consultado el 16 de octubre de 2022).

[17] “Curso de Programacion con Minecraft | Crack The Code”. https://www.crackthecode.la/cursos/curso-de-minecraft-basico (consultado el 16 de octubre de 2022).

[18] “Bootstrap: ¿qué es, para qué sirve y cómo instalarlo?” https://rockcontent.com/es/blog/bootstrap/ (consultado el 8 de enero de 2023).

[19] “*React*-Bootstrap · *React*-Bootstrap Documentation”. https://*React*-bootstrap.github.io/getting-started/why-*React*-bootstrap/ (consultado el 15 de enero de 2023).

[20] “Ratios · Bootstrap v5.0”. https://getbootstrap.com/docs/5.0/helpers/ratio/ (consultado el 15 de enero de 2023).

[21] “Procesos secundarios en Node.js (Child process) | by Diego Esteban | Medium”. https://medium.com/@dcortes.net/child-process-en-node-js-69e537c6a50f (consultado el 8 de enero de 2023).

[22] “MinGW G++”. https://www.fdi.ucm.es/profesor/luis/fp/devtools/MinGWUso.html (consultado el 8 de enero de 2023).

[23] “Qué es Visual Studio Code y qué ventajas ofrece | OpenWebinars.net”. https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/ (consultado el 16 de octubre de 2022).

[24] “Visual Studio: IDE y Editor de código para desarrolladores de software y Teams”. https://visualstudio.microsoft.com/es/ (consultado el 16 de octubre de 2022).

[25] “API: qué es y para qué sirve”. https://www.xataka.com/basics/api-que-sirve (consultado el 15 de enero de 2023).

[26] “Software de Diagramas Online | Lucidchart”. https://www.lucidchart.com/pages/es (consultado el 5 de diciembre de 2022).

[27] “¿Qué es un diagrama de flujo? | Lucidchart”. https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-flujo (consultado el 15 de enero de 2023).

[28] “▷ Diagrama de actividades”. https://diagramasuml.com/actividades/ (consultado el 19 de enero de 2023).

[29] “Diagramas de Actividades | LENGUAJE DE MODELADO UNIFICADO UML”. https://unadzsurlab.com/UML/U1/diagramas_de_actividades.html (consultado el 19 de enero de 2023).

[30] “Placeholder - Semantic UI *React*”. https://*React*.semantic-ui.com/elements/placeholder/ (consultado el 19 de enero de 2023).

5. ANEXOS

La inclusión de anexos como parte complementaria al desarrollo del presente Trabajo de Integración Curricular es esencial para garantizar una comprensión completa y detallada del mismo. Los anexos proporcionan información adicional y detalles técnicos que son necesarios para comprender y evaluar completamente el trabajo.

ANEXO I. Código del prototipo Flowchart.

ANEXO II. Código del API de conexión entre Flowchart y G++.

ANEXO III. Documentos de Encuesta y Resultados.

ANEXO I. Código del prototipo Flowchart.

El código para ejecutar la aplicación Flowchart está disponible para descargar en el enlace proporcionado. A continuación, se presenta el archivo correspondiente

FlowchartMain

ANEXO II. Código del API de conexión entre Flowchart y G++.

El código para la API de conexión entre la aplicación desarrollada y G++, que permite la compilación, está disponible para descargar en el enlace proporcionado. A continuación, se presenta el archivo correspondiente.

API_FlowChart

ANEXO III. Documentos de Encuesta y Resultados.

Se encuentran los documentos utilizados para validar los requerimientos y llegar a las conclusiones mediante su análisis.

Documentos